ТРУДЫ МОЛОДЫХ УЧЕНЫХ ФАКУЛЬТЕТА КОМПЬЮТЕРНЫХ НАУК ВГУ

ВЫПУСК 5

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет компьютерных наук

ТРУДЫ МОЛОДЫХ УЧЁНЫХ ФАКУЛЬТЕТА КОМПЬЮТЕРНЫХ НАУК ВГУ

В ы п у с к 5

Под общей редакцией
кандидата технических наук Д. Н. Борисова

Воронеж
2025

УДК 004.65 + 004.438.5
ББК Ч481(2)22
Т78

Рекомендовано к опубликованию
Ученым советом факультета компьютерных наук ВГУ

Т78 **Труды молодых учёных факультета компьютерных наук ВГУ.** Выпуск 5 / под общ. ред. Д. Н. Борисова ; Воронежский государственный университет. – Воронеж, 2025. – 719 с.

В сборник включены научные работы студентов бакалавриата и магистратуры факультета компьютерных наук ВГУ, выполненные в 2024-2025 гг. под руководством преподавателей факультета, представленные в виде докладов и рекомендованные к опубликованию оргкомитетами студенческих научных конференций.

УДК 004.65 + 004.438.5
ББК Ч481(2)22

© Воронежский государственный университет, 2025

Статьи студентов бакалавриата

Тестирование безопасности веб-приложений методом черного ящика

В. Ю. Акинин

Студент бакалавр

А. В. Акимов

Старший преподаватель

Введение

Современные веб-приложения являются ключевыми элементами цифровой инфраструктуры, которые обеспечивают широкий спектр услуг, начиная от электронной коммерции, заканчивая государственными сервисами. В связи с этим увеличивается количество кибератак, направленных на кражу персональных данных пользователей и нарушение функционирования систем. Так по данным компании «RED Security» за 2024 год совокупное число инцидентов информационной безопасности выросло в 2,5 раза по сравнению с 2023 годом [1, 2]. Вследствие чего безопасность программных продуктов становится критически важным аспектом разработки и эксплуатации. Особое значение в контексте обеспечения безопасности и качества ПО приобретает тестирование.

Одной из проблем при тестировании безопасности веб-приложений является интерпретация результатов. Многие инструменты предоставляют результаты сканирования в «сыром» виде, что неудобно для пользователей без глубоких знаний в области уязвимостей.

Данная работа посвящена исследованию наиболее распространенных видов уязвимостей, эксплуатируемых для атак на веб-приложения, таких как: SQL-инъекции, межсайтовый скриптинг, межсайтовая подделка запросов, а также разработке инструмента сканирования безопасности веб-приложений с графическим интерфейсом и автоматической интерпретацией результатов.

1. SQL-инъекции

SQL-инъекции представляет собой атаку, при которой злоумышленник внедряет вредоносный код в запросы, отправляемые на сервер системы управления базами данных (СУБД). Основной причиной возникновения уязвимости является использование динамических

© Акинин В. Ю., Акимов А. В., 2025

запросов к БД при проектировании приложения [3]. Такие запросы формируются путём конкатенации пользовательских данных, например, данных из формы или URL, с текстом запроса. Сервер базы данных выполняет запрос с внедрёнными данными так, как если бы они были частью легитимного SQL-кода. Пример эксплуатации уязвимости представлен на рис. 1.

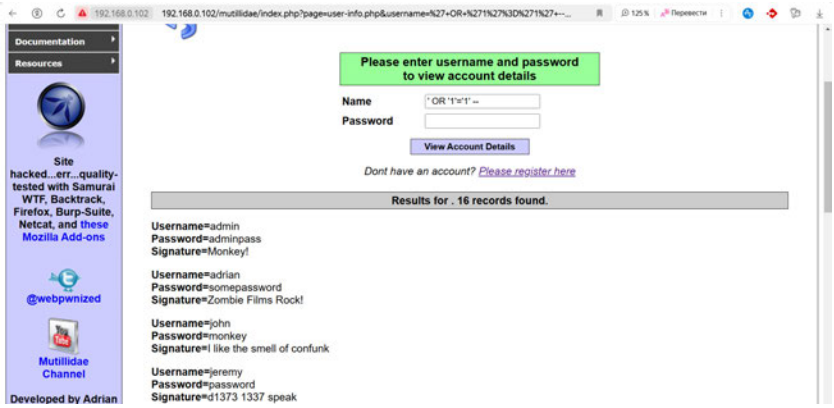


Рис. 1. Эксплуатация уязвимости типа SQL-инъекция в веб-приложении

Обнаружить уязвимость типа SQL-инъекция можно как ручным, так и автоматизированным тестированием. SQLMap – популярный инструмент с открытым исходным кодом, предназначенный для автоматизации процесса выявления и эксплуатации SQL-инъекций [4]. Основные возможности инструмента включают: автоматическое определение типа БД, проведение атак на основе различных техник SQL-инъекций таких как: «Error-based», «Union-based», «Blind» и других. Пример отчета о результатах сканирования представлен на рис. 2.

2. Межсайтовый скриптинг

Межсайтовый скриптинг (XSS) – это тип уязвимости в веб-приложениях, который позволяет злоумышленникам внедрять вредоносный код на веб-страницы, просматриваемые другими пользователями. Основной причиной возникновения уязвимости XSS является недостаточная обработка или фильтрация вводимых пользователями данных [5].

Пример эксплуатации уязвимости представлен на рис. 3. Атака заключается в передаче вредоносного JavaScript-кода в поле ввода

имени пользователя. Так как пользовательские данные не фильтруются, браузер исполняет внедренный код, который представляет собой вызов функции alert().

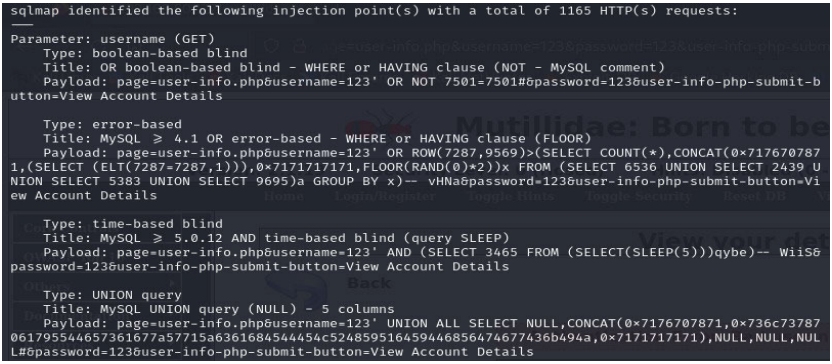


Рис. 2. Отчет о результатах сканирования инструмента SQLMap

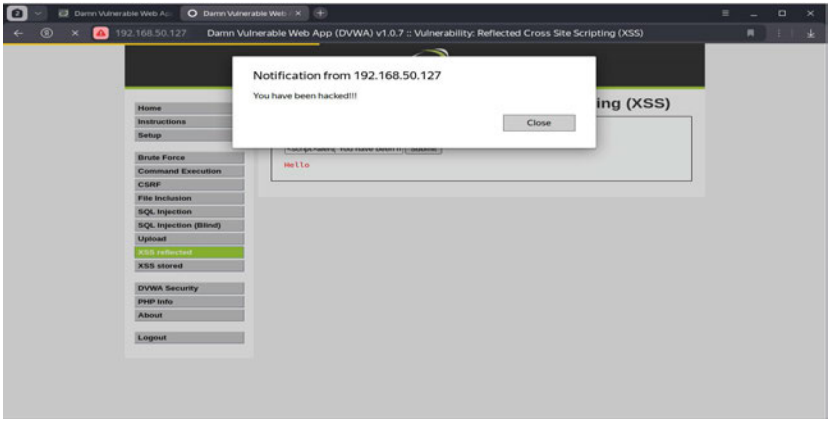


Рис. 3. Пример эксплуатации уязвимости типа XSS-инъекция

XSStrike – продвинутый инструмент для выявления уязвимостей, связанных с межсайтовым скриптингом. Основные функции включают: сканирование параметров веб-страниц, генерация скриптов на основе контекстного анализа, поддержка GET и POST запросов [6]. Пример отчета о результатах сканирования представлен на рис. 4.

Рис. 4. Отчет о результатах сканирования XSStrike

3. Межсайтовая подделка запроса

Межсайтовая подделка запроса (CSRF) – атака, которая обманом заставляет авторизованного пользователя выполнять нежелательные действия, например, смена пароля, отправляя вредоносные запросы без его ведома. Основными причинами возникновения уязвимости являются: отсутствие проверки веб-приложением адреса источника запроса, автоматическая отправка куки браузером, нарушение принципов архитектурного стиля REST и HTTP-идемпотентности [7].

В качестве примера эксплуатации уязвимости рассмотрим реализацию вредоносной страницы, при переходе на которую автоматически запускается скрипт смены пароля к аккаунту пользователя в уязвимом веб-приложении. Код страницы представлен на рис. 5.

```
<form action="http://192.168.50.127/dvwa/vulnerabilities/csrf/" method="GET">
    <input type="111111" type="hidden" autocomplete="off" name="password_new">
    <input type="111111" type="hidden" autocomplete="off" name="password_conf">
    <input type="hidden" value="Change" name="Change">
</form>

<script>document.getElementById("form7").submit();</script>
```

Рис. 5. Эксплуатация CSRF-уязвимости через вредоносную страницу для смены пароля

OWASP ZAP (Zed Attack Proxy) – инструмент с открытым исходным кодом, основанный на методике тестирования черного ящика. Продукт позволяет сканировать веб-приложения на наличие широкого спектра уязвимостей, включая CSRF [8]. Пример отчета о сканировании CSRF представлен на рис. 6.

Рис. 6. Отчет о результатах сканирования OWASP ZAP

Рассмотренные инструменты предоставляют результаты сканирования в достаточно сложном для восприятия виде, что создает барьер для специалистов, не обладающих глубокими знаниями в сфере уязвимостей и информационной безопасности, в частности для тестировщиков программного обеспечения. В связи с этим возникает необходимость разработки инструмента, который предоставляет результаты в удобном и наглядном виде, облегчая их интерпретацию.

4. Разработанные инструмент сканирования

Инструмент основан на модульной архитектуре и включает модули сканирования уязвимостей типа SQL-инъекция, XSS-инъекция, CSRF, а также модуль полного сканирования, который запускает одновременное сканирование на вышеупомянутые типы уязвимостей. Инструмент реализован с помощью языка программирования Python и фреймворка Flask [9]. Для сканирования на уязвимость типа SQL-инъекция в решение интегрирован SQLMap. Реализованы собственные алгоритмы сканирования XSS и CSRF с помощью библиотек requests – для работы с HTTP-запросами и BeautifuilSoup – для парсинга HTML-страниц с целью идентификации полей ввода и других элементов.

Пользователь задает целевой URL, выбирает режим сканирования и запускает сканирование через веб-интерфейс. Результаты сканирования автоматически обрабатываются и отображаются в виде таблицы с указанием параметров, типов уязвимостей, примеров атак и рекомендаций по защите. Дополнительно реализована возможность экспорта отчета в формате PDF. Ключевым преимуществом является интуитивный интерфейс, позволяющий использовать инструмент даже специалистам с базовыми знаниями в области информационной безопасности. Автоматизация процесса сканирования и формирования отчётов значительно сокращает время аудита веб-приложений. Для наглядной демонстрации логики работы инструмента построена

диаграмма деятельности. Она отражает основные этапы работы решения, начиная с ввода пользователем данных, заканчивая формированием отчета о результатах сканирования. Диаграмма представлена на рис. 7.

Рис. 7. Диаграмма деятельности

Предварительное тестирование разработанного инструмента производилось на уязвимых веб-приложениях multidae и DVWA, развернутых на базе виртуальной машины metasploitable2 [10]. Результаты сканирования на уязвимость типа SQL-инъекция представлены на рис. 8.

Пример отчета для экспорта в PDF-формате представлен на рис. 9.

Заключение

Данная статья посвящена разработке и реализации инструмента сканирования безопасности веб-приложений с графическим интерфейсом, автоматической интерпретацией результатов и возможностью экспорта отчетов о сканировании. В итоге данной работы получился полностью рабочий программный инструмент, реализованный в виде веб-приложения. Использование данного решения позволяет обнаружить уязвимости в веб-приложениях такие как: SQL-инъекции, XSS-инъекции и CSRF. Инструмент может быть полезен как разработчикам программного обеспечения, так и

пользователям, которые не обладают глубокими знаниями в области информационной безопасности.

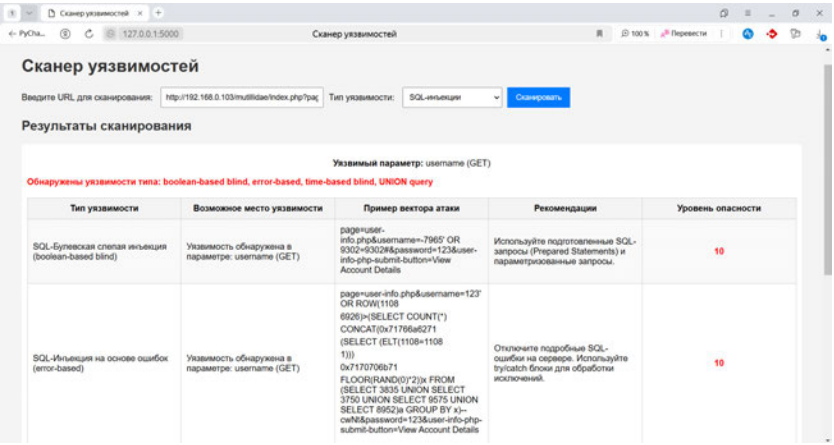


Рис. 8. Результаты сканирования на уязвимость типа SQL-инъекция

Отчёт о сканировании уязвимостей

Уязвимый параметр: username (GET)

Обнаружены уязвимости типа: boolean-based blind, error-based, time-based blind, UNION query

Тип уязвимости	Возможное место уязвимости	Пример вектора атаки	Рекомендации	Уровень опасности
SQL-Булевская слепая инъекция (boolean-based blind)	Уязвимость обнаружена в параметре: username (GET)	page=user-info.php&username=-9482' OR 4953=4953#&password=123&user-info-php-submit-button=View Account Details	Используйте подготовленные SQL-запросы (Prepared Statements) и параметризованные запросы.	10
SQL-Инъекция на основе ошибок (error-based)	Уязвимость обнаружена в параметре: username (GET)	page=user-info.php&username=123' OR ROW(8475 6959)>(SELECT COUNT(*) CONCAT(0x7171627671 (SELECT (ELT(8475=8475 1))) 0x7178717171 FLOOR(RAND(0)*2))x FROM (SELECT 2721 UNION SELECT 2578 UNION SELECT 2012 UNION SELECT 2705)a GROUP BY x)-- gLUY&password=123&user-info-php-submit-button=View Account Details	Отключите подробные SQL-ошибки на сервере. Используйте try/catch блоки для обработки исключений.	10

Рис. 9. Пример отчета о сканировании в PDF-формате

Список литературы

1. Число кибератак в России и в мире – Текст: электронный// tadviser: [Электронный ресурс]. – Режим доступа: https://www.tadviser.ru/index.php/Статья:Число_кибератак_в_России_и_в_мире
2. Статистика атак на веб-приложения: II квартал 2017 года – positive technologies [Электронный ресурс]. – Режим доступа: https://www.ptsecurity.com/ru-ru/research/analytics/web-application-attacks-2017-q2/
3. Clarke-Salt, J. SQL Injection Attacks and Defense / Justin Clarke-Salt, Justin Clarke. – Elsevier, 2012. – 576 p. – ISBN 1597499633, 9781597499637.
4. Stamparm, sqlmap: Automatic SQL injection and database takeover tool. GitHub. [Электронный ресурс]. – Режим доступа: https://github.com/sqlmapproject/sqlmap
5. Grossman, J. XSS attacks: cross-site scripting exploits and defense/ J. Grossman, Syngress, Burlington. – Mass, 2007. – 448 p. – ISBN 1597491543, 9781597491549.
6. XSStrike: Most advanced XSS scanner [Электронный ресурс]. – Режим доступа: https://github.com/s0md3v/XSStrike
7. Hoffman, A. Web Application Security Exploitation and Countermeasures for Modern Web Applications/ A. Hoffman. – O`Reilly Media, 2020. – 327 p. – ISBN 1492053112, 978-1492053118.
8. OWASP Zed Attack Proxy project landing page [Электронный ресурс]. – Режим доступа: https://github.com/OWASP/www-project-zap
9. Flask [Электронный ресурс]. – Режим доступа: https://flask.palletsprojects.com/en/stable/
10. Metasploitable: intentionally vulnerable Linux virtual machine [Электронный ресурс]. – Режим доступа: https://sourceforge.net/projects/metasploitable/

Интеграция WebRTC с серверной логикой

Д.Н. Бородин

Студент бакалавр

Н.К. Самойлов

Старший преподаватель

Введение

WebRTC (Web Real-Time Communication) — это открытая технология, предоставляющая браузерам и мобильным приложениям возможность организовать обмен аудио, видео и данными в режиме реального времени без установки дополнительных плагинов. Появившись как часть инициативы по стандартизации W3C и IETF, WebRTC быстро стала ключевым инструментом в разработке современных коммуникационных сервисов: от видеочатов до многопользовательских приложений и совместных рабочих пространств.

Несмотря на то, что WebRTC ориентирована на прямую P2P-коммуникацию, практически все реальные сценарии ее использования предполагают наличие серверной логики. Сервер необходим как минимум для организации сигналинга — обмена данными, необходимыми для установления соединения (SDP, ICE), но часто его функции гораздо шире. Особенно это актуально в архитектуре тонкого клиента, где вся бизнес-логика сосредоточена на сервере.

Цель данной научной статьи — рассмотреть технические аспекты интеграции WebRTC с серверной частью приложения, определить архитектурные подходы и проанализировать практические проблемы, возникающие при реализации P2P-взаимодействия через тонкий клиент.

1. Архитектура WebRTC: компоненты и принципы работы

WebRTC использует три ключевых компонента API [1]:

- getUserMedia() — это функция (метод), предоставляющая доступ к камере и микрофону;
- RTCPeerConnection — это интерфейс (класс) для установки соединения между участниками;
- RTCDataChannel — это интерфейс (класс) для обмена произвольными данными.

© Бородин Д.Н., Самойлов Н.К., 2025

Но WebRTC по умолчанию не умеет устанавливать соединения «самостоятельно». Для этого необходим процесс сигналинга — обмен информацией о сеансе (SDP) и параметрах подключения (ICE-кандидаты) [2]. Именно он требует интеграции с сервером.

Также в архитектуру WebRTC входят:

– STUN-серверы, помогающие определить внешний IP и порт клиента;
– TURN-серверы, которые обеспечивают ретрансляцию трафика, если соединение напрямую невозможно [3];
– Signaling-сервер, реализующий обмен SDP и ICE.

На рис. 1 представлена архитектура WebRTC с разделением по уровням: от пользовательского интерфейса до серверных компонентов. Такая структура облегчает реализацию сигналинга и интеграцию с STUN/TURN серверами.

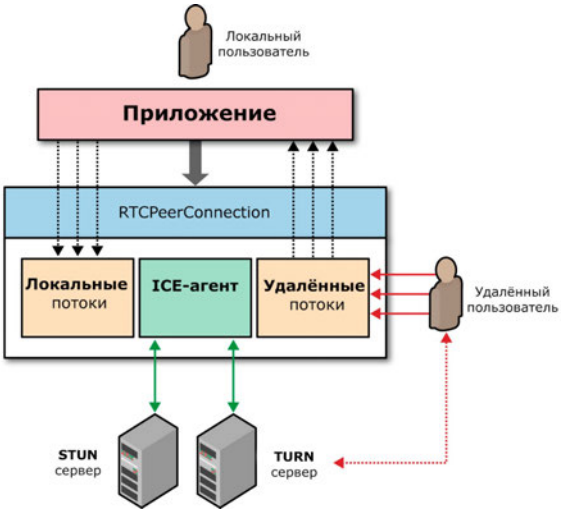


Рис. 1. Архитектура WebRTC

2. Интеграция WebRTC в архитектуру с тонким клиентом

Тонкий клиент (thin client) — это интерфейс, в котором отсутствует внутренняя логика работы приложения: он отображает данные и передает действия пользователя серверу. Это упрощает обслуживание и защищает бизнес-логику.

В случае с WebRTC тонкий клиент реализует:

– инициализацию камеры/микрофона;
– создание RTCPeerConnection;

– отправку SDP/ICE на сервер (через WebSocket);
– отображение входящего потока.

Вся логика управления комнатами, проверка прав доступа, логирование, маршрутизация осуществляется на сервере.

Можно выделить следующие шаги взаимодействия:

– подключение клиента к сигналинг-серверу;
– инициализация getUserMedia() и получение локального медиапотока;
– создание SDP-предложения;
– отправка SDP и ICE-кандидатов на сервер;
– сервер проверяет пользователя, добавляет его в комнату;
– сервер пересылает SDP и ICE другим участникам;
– клиенты устанавливают P2P-соединение через NAT с использованием STUN/TURN.

На рис. 2 изображен процесс инициализации peer-to-peer соединения между двумя WebRTC-клиентами. Сначала оба клиента обращаются к STUN-серверу для определения своих публичных IP-адресов. Затем с помощью сигнального сервера обмениваются предложениями SDP и ICE-кандидатами. После согласования параметров начинается передача медиаданных по прямому каналу.

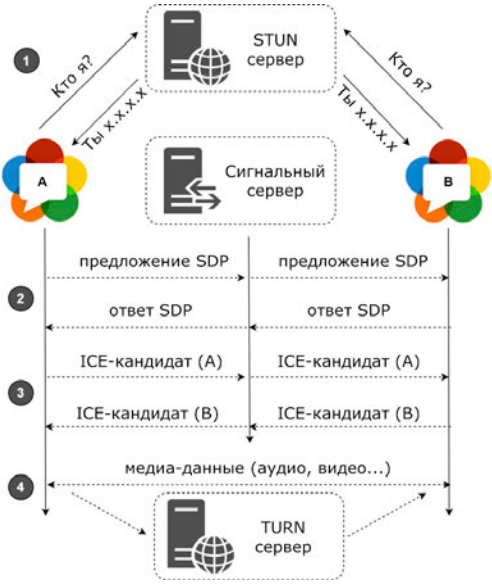


Рис. 2. Signaling между клиентами

3. Реализация сигналинга и логики взаимодействия

Чаще всего сигналинг реализуется через WebSocket, что обеспечивает низкую задержку и постоянное соединение. На серверной стороне используется карта подключений пользователей, позволяющая направлять сообщения между ними. Пример реализации обработки WebRTC-соединений на сервере с использованием WebSocket приведён в листинге [4].

Листинг

Обработка WebRTC-соединения на стороне сервера

```
const users = new Map();
wsServer.on('connection', (socket) => {
  socket.on('message', (msg) => {
    const data = JSON.parse(msg);
    if (data.type === 'offer') {
      users.get(data.to).send(JSON.stringify({type: 'offer',
data: data.data }));
    }
  });
});
```

Параллельно сервер выполняет:

1аутентификацию и авторизацию;

2проверку лимитов (например, макс. участников в комнате);

3запись метаданных сессий (для отладки и статистики).

4. Безопасность при работе с WebRTC

WebRTC обеспечивает высокий уровень безопасности «из коробки» [5], используя:

- DTLS (Datagram Transport Layer Security) для шифрования сигнальных сообщений;
- SRTP (Secure Real-Time Protocol) для медиа;
- шифрование всех данных и медиа от клиента к клиенту, включая TURN-трафик.

Однако слабым звеном может быть сигналинг, особенно если он реализован без HTTPS или без проверки подлинности клиента.

Рекомендации по безопасности:

- использовать HTTPS + WSS (WebSocket Secure);
- аутентифицировать пользователей с помощью JWT или OAuth;
- хранить временные ключи шифрования на сервере и ограничивать срок их действия;
- ограничивать доступ к TURN-серверу по токенам.

5. Проблемы при внедрении

На практике разработчики сталкиваются с рядом проблем:

– сложности NAT traversal: в сетях с симметричным NAT TURN-сервер становится обязательным;
– медленная настройка соединения: большое количество ICE-кандидатов может замедлить handshaking;
– проблемы с браузерами: поведение API немного отличается в Chrome, Firefox и Safari;
– сложность отладки: из-за P2P-связи проблемы могут проявляться не у всех пользователей сразу.

Рекомендуется использовать инструменты:
– chrome://webrtc-internals/ — отладка в Chrome;
– логирование всех стадий SDP и ICE;
– wireshark и tcpdump для анализа трафика.

6. Реальные сценарии применения WebRTC с тонким клиентом

Среди сценариев можно выделить следующие:

1. Видеочаты и конференции: наиболее распространенный случай — чат-комнаты (например, Jitsi, Zoom), где вся логика управления комнатами на сервере, а клиенты — это просто медиапотоки и интерфейс.

2. Онлайн-обучение: платформы дистанционного образования (например, Stepik, Skyeng) используют WebRTC для видеосвязи между учеником и преподавателем, а также для обмена интерактивными данными.

3. Телемедицина: врачебные консультации и наблюдения через WebRTC позволяют интегрировать шифрованную видеосвязь в медицинские ИС, соблюдая при этом HIPAA и GDPR.

4. Службы поддержки: операторы могут принимать вызовы от клиентов прямо в браузере, получая дополнительно доступ к истории, транскрипциям и записям — все это хранится на сервере.

7. Масштабирование и отказоустойчивость WebRTC-систем

WebRTC изначально предполагает P2P-связь, но при большом числе пользователей этого недостаточно. Для масштабирования и обеспечения отказоустойчивости применяются дополнительные серверы и архитектурные решения.

Наиболее распространенные подходы масштабирования — медиасерверы типа SFU и MCU. SFU (Selective Forwarding Unit) пересылает медиа-потоки между участниками без перекодирования, снижая нагрузку на сервер и обеспечивая меньшие задержки. Однако при этом клиенту нужно обрабатывать несколько потоков. MCU (Multipoint Control Unit), наоборот, агрегирует и перекодирует потоки в

один, снижая нагрузку на клиента, но увеличивая ее на сервер и добавляя задержки.

Сравнение данных подходов приведено в таблице. Схематическое различие между ними показано на рис. 3.

Таблица

Сравнение подходов MCU и SFU

Подход	Преимущества	Недостатки
SFU	Меньше задержка, экономия ресурсов	Нагрузка на клиент (обработка нескольких потоков)
MCU	Удобство, меньше потоков на клиенте	Высокая нагрузка на сервер, задержки

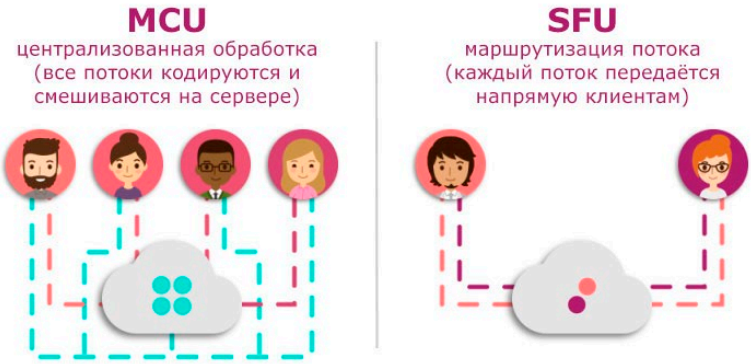


Рис. 3. Сравнение MCU и SFU

Для масштабирования используют балансировку нагрузки и георасределение. Сигналинг может идти через балансировщики, например, NGINX и WebSocket-серверы. TURN-серверы размещают в разных регионах, а медиасерверы (Janus, Jitsi Videobridge, mediasoup) поддерживают автоматическое масштабирование по нагрузке. Также внедряются механизмы мониторинга: проверка состояния серверов, архитектура с резервными инстансами и системы обмена сообщениями (RabbitMQ, Kafka) для синхронизации серверов.

Масштабируемые WebRTC-системы часто строят на облачных решениях и с использованием CDN. Облачные TURN-серверы (Twilio, Xirsys), виртуализация и контейнеризация (Docker, Kubernetes) позволяют быстро развертывать медиасерверы. CDN обеспечивает

стабильную доставку контента. Все это позволяет обслуживать тысячи параллельных видеосессий и использовать WebRTC в крупных корпоративных, образовательных и конференционных платформах.

Заключение

Интеграция WebRTC в архитектуру с тонким клиентом требует детального проектирования взаимодействия между клиентом и сервером. Несмотря на то, что WebRTC обеспечивает прямое P2P-соединение, в реальной разработке нельзя обойтись без серверной логики, которая управляет сигналингом, авторизацией, статистикой и безопасностью.

Понимание архитектуры, грамотная реализация сигналинга, использование STUN и TURN, а также защита каналов связи являются ключевыми условиями для создания надежной и масштабируемой системы реального времени [6]. Благодаря WebRTC разработчики могут строить сложные коммуникационные решения, не прибегая к установке дополнительного ПО, что делает его незаменимым инструментом для современных веб-приложений.

Список литературы

1. WebRTC API Reference [Электронный ресурс]. – MDN Docs. – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API

2. Signaling and WebRTC [Электронный ресурс]. – HTML5 Rocks. – Режим доступа: https://www.html5rocks.com/en/tutorials/webrtc/basics/

3. Traversing NAT with STUN and TURN [Электронный ресурс]. – WebRTC.org. – Режим доступа: https://webrtc.org/

4. GitHub: Simple Peer Signaling Server [Электронный ресурс]: исходный код. – Режим доступа: https://github.com/feross/simple-peer

5. WebRTC Security [Электронный ресурс]: исходный код. – Режим доступа: https://webrtc-security.github.io/

6. RFC 8825: Overview: Real-Time Protocols for Browser-Based Applications [Электронный ресурс]. – Режим доступа: https://www.rfc-editor.org/info/rfc8825

Значимость интегрирования документооборота в бизнес-процессы посредством инструментария 1С

Н. А. Бортник

Студент бакалавр

С. В. Бебнева

Ст. преподаватель

Введение

С развитием цифровых технологий организации стали все больше нуждаться в автоматизации бизнес-процессов, что связано с совершенствованием документооборота, используемого в системе 1С.Предприятие. Именно быстрая обработка документальной информации позволила успешным фирмам занять выгодные позиции на конкурирующем рынке, снизить издержки, повысить качество обслуживания клиентов.

1. Законодательная база ЭДО и роль 1С

На основании законодательных документов, таких как Федеральный закон №63-ФЗ «Об электронной подписи», Налоговый кодекс РФ, Федеральный закон №402-ФЗ «О бухгалтерском учёте», Федеральный закон №149-ФЗ «Об информации, информационных технологиях и о защите информации» установлены правила и обязательства электронного документооборота, что стало поворотным моментом как для разработчиков, так и для организаций. Внедренный электронный документооборот в 1С позволил заменить бумажные документы на более эффективные и быстро обрабатываемые – электронные. Наиболее популярной платформой обработки электронных документов, используемой в деятельности хозяйствующего субъекта, является система 1С:Предприятие, с помощью которой обрабатываются различные типы документов, касающиеся как деятельности, так и бухгалтерии.

2. Универсальность 1С:Документооборота

Важно отметить, что превосходство применения 1С Предприятия в электронном документообороте состоит в автоматизации формирования

© Бортник Н. А., Бебнева С. В., 2025

пакетов документов и быстрой передачи данных до проверяющих органов, а также в ускорении действий обрабатывания.

Система 1С является универсальной платформой для предприятий, включающей различные конфигурации, автоматизирующие бизнес-процессы, которая способствует эффективному управлению бухгалтерией, кадрами, документооборотом, складами, и т.д.

Конфигурация 1С:Документооборот 8.3 является наиболее популярной среди российских компаний, способной не только автоматизировать процессы документооборота, но обрабатывать и хранить их. Данная конфигурация включает в себя такие важные инструменты, как договора, счета-фактуры, акты и прочие необходимые документы для ведения бизнеса.

3. Автоматизация бухгалтерского документооборота

Преимущества использования конфигурации 1С: Документооборот, предназначенной для обеспечения эффективной передачи документов заключаются:

– в возможности сокращения времени при обработке документов, что приводит к оптимизации бюджета, то есть автоматизация рутинных бумажных процессов позволит снизить финансовые затраты, ускорить производительность и исключить ошибки;

– в уменьшении опасности потери данных при передаче и увеличении достоверности, что упрощает операцию архивирования и обеспечивает сохранность документов;

– в повышении конфиденциальности информации, которая кодируется и имеет ограниченный доступ нежелательных пользователей;

– в долговременном хранении документов в виде архивов (1С, с конфигурацией «Архив»).

В условиях стремительной цифровизации бизнеса автоматизация бухгалтерского документооборота становится не просто желательной, а необходимой мерой для эффективного управления финансовыми потоками и соответствия законодательным требованиям.

Главным стимулирующим фактором в автоматизации бухгалтерского документооборота является жесткая конкуренция на рынке, в которой лидерами являются организации, обладающие новейшими технологиями, обеспечивающими быстрый и защищенный процесс передачи данных.

На конкурирующем рынке современные предприятия сталкиваются с возрастающим объемом документов, сложностью налогового регулирования и необходимостью оперативного принятия управленческих решений.

4. Современные ИТ-решения и интеграция с 1С

Российские разработчики предлагают большое количество современных систем, обеспечивающих электронный документооборот (система «ДЕЛО» с возможностью синхронизации с бухгалтерией, платформа «ЕФРАТ», интегрируемая с 1С, система ТЕЗИС возможностью интегрироваться с учетной, кадровой и бухгалтерской системой 1С и прочие), которые можно использовать в совместной работе с 1С.

В этом контексте система 1С играет ключевую роль, обеспечивая полнофункциональный электронный документооборот, комплексные решения для бухгалтерского учета, налогообложения и финансового контроля. В комплексе информационные технологии позволяют автоматизировать процессы, снизить затраты на ручной труд и повысить прозрачность ведения бухгалтерского учета.

5. Ключевые преимущества использования 1С

Определяя преимущества использования 1С для электронного документооборота, можно выделить следующее:

1. Соответствие законодательству. Одним из важнейших аспектов использования 1С в бухгалтерском учете является его соответствие законодательным нормам. Государственное регулирование постоянно изменяется, а 1С обеспечивает оперативное обновление нормативных требований в своих продуктах. Основные преимущества, которые получает предприятие благодаря использованию 1С:

– обязательный электронный документооборот (ЭДО) с налоговой службой и контрагентами, что позволяет ускорить процесс документооборота и минимизировать вероятность потерь документов;

– применение новых форм отчетности и счетов-фактур согласно актуальным нормативным требованиям;

– автоматический расчет налогов и взносов, что уменьшает вероятность ошибок и делает процесс налогообложения более прозрачным;

– снижение рисков штрафов и санкций со стороны контролирующих органов за счет точного соответствия требованиям законодательства;

– интеграция с государственными системами («Честный Знак», «ГИС МТ», «ГИС ЖКХ»), обеспечивающая бесперебойный документооборот и контроль за товарами;

– обеспечение юридической значимости документов за счет электронной цифровой подписи (ЭЦП), что позволяет отказаться от бумажного документооборота.

2. Оптимизация бизнес-процессов. Автоматизация документооборота с помощью 1С позволяет не только соответствовать законодательным нормам, но и значительно повысить эффективность внутренних процессов компании. Благодаря внедрению системы 1С можно:

– сократить рутинные операции, связанные с обработкой первичных документов;

– минимизировать ошибки, возникающие при ручном вводе данных;

– ускорить обработку информации, что позволяет оперативно формировать управленческую отчетность;

– автоматически загружать и обрабатывать банковские выписки, что сокращает время работы бухгалтерии;

– осуществлять контроль взаиморасчетов и анализ дебиторской или кредиторской задолженности;

– настроить систему напоминаний и уведомлений, что помогает своевременно реагировать на задолженности и изменения в финансовом состоянии компании;

– автоматизировать расчет заработной платы и налогов, что упрощает работу кадровых и бухгалтерских служб.

3. Повышение прозрачности учета. Точность и прозрачность учета важный фактор для руководства компании. Использование 1С в документообороте позволяет обеспечить:

– централизованное хранение данных, исключающее дублирование информации и несоответствия в бухгалтерской отчетности;

– настройку аналитических отчетов по видам деятельности, что помогает более детально анализировать финансовые потоки;

– интеграцию с банковскими и налоговыми системами, позволяя вести учет в режиме реального времени;

– автоматическое формирование отчетности, что сокращает нагрузку на бухгалтерию и снижает вероятность ошибок;

– повышение достоверности финансовой отчетности, что критично для привлечения инвесторов и кредитных организаций.

4. Внедрение цифровых технологий. Современные технологии значительно упрощают работу с бухгалтерскими документами. В системе 1С предусмотрены интеграции с:

– криптографическими системами защиты данных (ЭЦП) для безопасного обмена документами;

– государственными платформами, обеспечивающими контроль за движением товаров и услуг;

– облачными сервисами, позволяющими сотрудникам удаленно работать с бухгалтерскими документами;

– мобильными приложениями, что расширяет возможности удаленного управления финансовыми потоками.

6. Недостатки системы 1С

Далее выделим недостатки современных конфигураций «1С:Бухгалтерия». Несмотря на функциональные возможности 1С, существуют определенные недостатки:

1. Неудобный интерфейс – сложность в освоении пользователями без технической подготовки.
2. Отсутствие автоматизации согласования платежей – требует ручной обработки данных.
3. Проблемы с консолидацией отчетности – при наличии нескольких юридических лиц.
4. Низкий уровень безопасности – уязвимости могут быть использованы злоумышленниками.
5. Сложность внедрения и обновления – требует квалифицированных специалистов.

7. Пути устранения недостатков

Рассмотрев преимущества и недостатки использования системы 1С в процессе документооборота, считаем важным предложить конкретные пути решения сложившихся недостатков.

1. Для улучшения интерфейса:

– внедрение кастомных обработок и расширений;

– использование веб-клиента «1С:Фреш» с адаптивным интерфейсом.

2. Для автоматизации согласования платежей следует:

– провести интеграцию с «1С:Документооборот», Контур. Диадок, СКБ Контур;

– разработать пользовательскую обработку для маршрутизации платежей.

3. Для оптимизации консолидации отчетности предлагается:

– внедрить «1С:Консолидацию»;

– использовать BI-систему (Power BI, Tableau) для проведения и демонстрации детального анализа;

– настроить обмен данными между базами через API.

4. Для повышения уровня безопасности следует:

– использовать аппаратные ключи (eToken, Рутокен);.

– включить аудит безопасности и разграничение прав доступа.

5. Для облегчения внедрения и обновлений важным является:

– процесс автоматизации обновлений через «1С:Обновление конфигураций»;

– использование Docker для 1С для минимизации проблем совместимости;

– внедрение DevOps-подходов для тестирования изменений.

Заключение

Итак, автоматизация бухгалтерского документооборота с помощью 1С позволяет значительно повысить качество и эффективность учета, защитить данные посредством их хранения в электронном виде, ускорить весь процесс документооборота, а сам процесс становится понятным и контролируемый. Однако успешное внедрение требует проработки возможных недостатков и использования дополнительных инструментов для их устранения. Внедрение новых технологий и методов управления данными делает 1С удобным и мощным инструментом для ведения бухгалтерского учета в компаниях с любым объемом документооборота.

Список литературы

1. Федеральный закон от 06.04.2011 № 63-ФЗ «Об электронной подписи» // Собрание законодательства РФ. – 2011. – № 15. – Ст. 2036.

2. Федеральный закон от 06.12.2011 № 402-ФЗ «О бухгалтерском учёте» // Собрание законодательства РФ. – 2011. – № 50. – Ст. 7344.

3. Налоговый кодекс Российской Федерации (часть первая) от 31.07.1998 № 146-ФЗ (ред. от 01.01.2024). – М. : Проспект, 2024.

4. Федеральный закон от 27.07.2006 № 149-ФЗ «Об информации, информационных технологиях и о защите информации» // Собрание законодательства РФ. – 2006. – № 31. – Ст. 3448.

5. Селифонов, С. В. Электронный документооборот в 1С: теория и практика / С. В. Селифонов. – М. : Инфра-М, 2020. – 216 с.

6. Смирнова, И. А. Автоматизация учета и документооборота: практическое руководство / И. А. Смирнова, Е. Н. Дьякова. – СПб. : Питер, 2022. – 304 с.

7. Гончаров, А. А. Внедрение электронного документооборота в организациях / А. А. Гончаров // Бухгалтерский учет. – 2023. – № 6. – С. 12-17.

8. Новикова, Е. С. Практика использования 1С: Документооборот / Е. С. Новикова // Главбух. – 2022. – № 14. – С. 48-53.

9. Казакова, И. Ю. Информационные технологии в экономике и управлении / И. Ю. Казакова. – М. : Юрайт, 2021. – 278 с.

10. 1С:Документооборот – эффективное управление корпоративной информацией [Электронный ресурс] : официальный сайт 1С. – Режим доступа: https://v8.1c.ru/document-management/

Разработка Telegram-бота для учёта личных финансов

Т. Д. Братышев

Студент бакалавр

Я. А. Воронцов

Доцент

Введение

Современный пользователь ежедневно сталкивается с необходимостью отслеживания и анализа личных финансовых расходов. При этом растёт интерес к средствам автоматизации, которые делают процесс учёта простым и удобным. Одним из перспективных решений является использование Telegram-ботов — программ, способных принимать и обрабатывать информацию через мессенджер.

В данной работе рассматривается разработка Telegram-бота, позволяющего вести учёт финансов, объединив данные из разных источников.

1. Постановка задачи

Для достижения поставленной цели необходимо решить следующие задачи:

– Реализовать обработку файлов банковских выгрузок в формате.xlsx;

– Добавить возможность распознавания QR-кодов чеков и извлечения информации из них;

– Обеспечить ручной ввод транзакций с сохранением в базе данных;

– Разработать модуль генерации аналитики в текстовом и табличном (.xlsx) формате.

2. Проектирование системы

В соответствии с требованиями была разработана User Flow диаграмма, показанная на рисунке 1, описывающая сценарий работы со стороны клиента.

© Братышев Т. Д., Воронцов Я. А., 2025

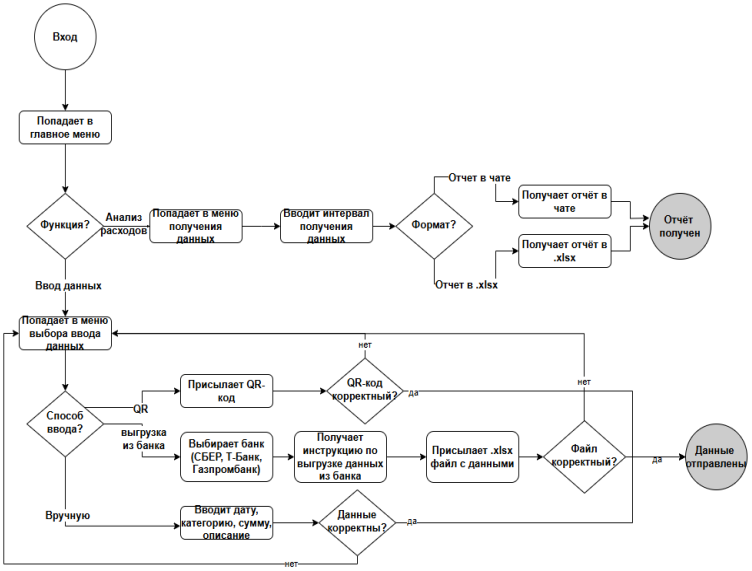


Рис. 1. User Flow диаграмма

Разработанное приложение построено по классической клиент-серверной архитектуре (рис. 2), в которой роли клиентов и сервера чётко разграничены.

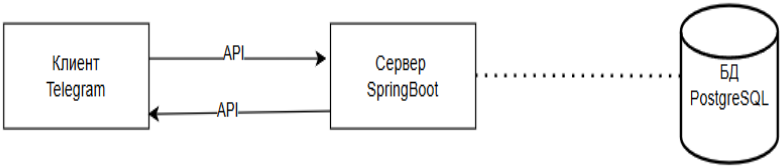


Рис. 2. Архитектура приложения

Клиентом в данной системе выступает пользователь Telegram, взаимодействующий с ботом через интерфейс мессенджера. Команды и сообщения, отправленные пользователем, поступают на серверную часть приложения через Telegram Bot API. Клиент не взаимодействует

напрямую с базой данных или внутренней бизнес-логикой, что обеспечивает высокий уровень абстракции и безопасности.

Сервером является Java-приложение [4], развернутое на базе Spring Boot [5]. Оно обрабатывает входящие запросы от Telegram через механизм Long Polling. При взаимодействии с пользователем бот получает сообщения (текст, файлы, изображения), обрабатывает их на сервере и сохраняет извлечённую информацию в базе данных.

3. Реализация логики

В ходе реализации проекта были выполнены следующие этапы:

– Выбор и настройка технологии Telegram Bot API [1] с использованием библиотеки TelegramBots (Java);

– Разработка механизмов обработки банковских выгрузок.xlsx с помощью библиотек Apache POI [2];

– Распознавание QR-кодов с чеков через библиотеку ZXing [3];

– Организация ручного ввода транзакций и сохранения в PostgreSQL [6] через Spring Data JPA [5];

– Реализация вывода аналитики по расходам в текстовом и табличном формате по заданному пользователем диапазону дат.

4. Реализация интерфейса

Была спроектирована информационная панель бота внутри приложения «Телеграм» (рис. 3).

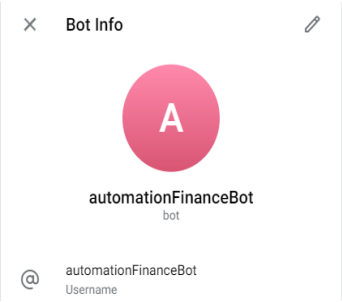


Рис. 3. Информационная панель бота

На рис. 4 продемонстрированы основные функции приложения.

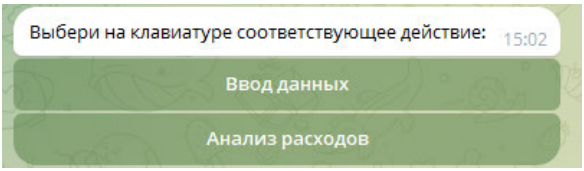


Рис. 4. Основные функции бота

На рис. 5 показаны способы ввода данных.

Рис. 5. Ввод данных

2. QR-код - пользователь отправляет изображение чека, содержащего QR-код, в котором содержатся следующие данные: дата и сумма операции. Помимо этого, останется ввести категорию и описание транзакции. Пример отправки QR-кода продемонстрирован на рис. 6.

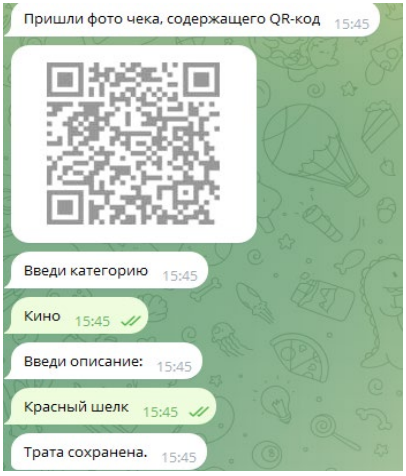


Рис. 6. Отправка QR кода

3. Выгрузка из банка - пользователь делает выгрузку из банка за определенный период (СБЕР, Т-Банк) в .xlsx формате и отправляет чат-боту (рис. 7).

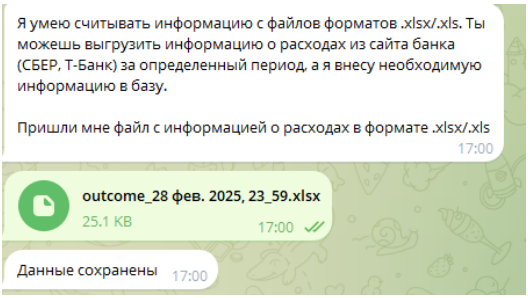


Рис. 7. Отправка выгрузки из банка

4. Ручной ввод - пользователь вводит следующие данные о транзакции: дата и время операции, сумма, категория и описание. На рис. 8 продемонстрирован ввод данных при помощи ручного ввода.

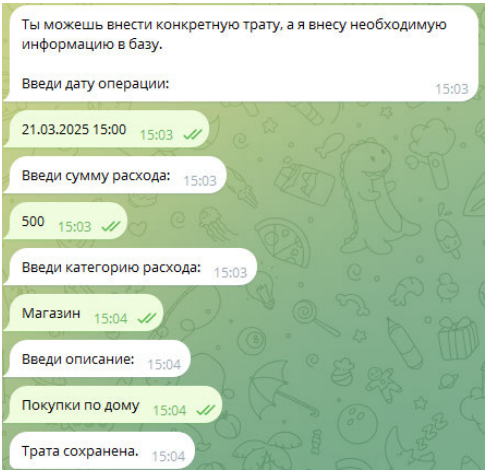


Рис. 8. Ручной ввод данных

Помимо ввода данных, пользователь может получить анализ расходов. Для этого следует отправить в чат интервал, по которому будут собраны данные.

На рис. 9 показан сформированный .xlsx файл, содержащий аналитику расходов за определенный период.

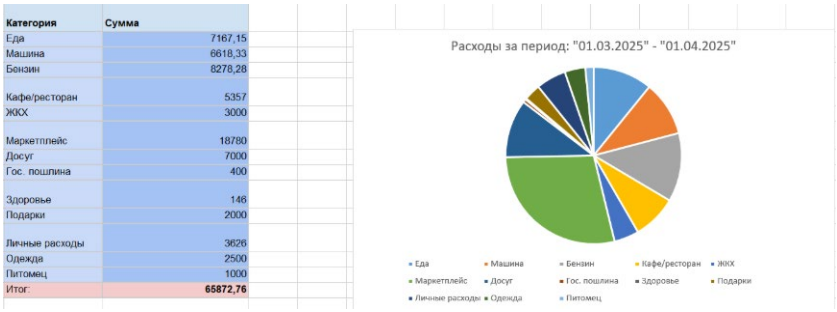


Рис. 9. Сформированный отчет за определенный период

Заключение

Разработка Telegram-бота для учёта личных финансов позволяет пользователю автоматизировать и упростить контроль расходов. Проект обладает гибкой архитектурой, лёгкой в сопровождении, и может быть расширен за счёт интеграции с банковскими API и ML-алгоритмами для автоматической категоризации.

Созданное решение может быть использовано как самостоятельное приложение или интегрировано в более сложные финансовые сервисы.

Список литературы

1. Telegram Bot API [Электронный ресурс]. – Режим доступа: https://core.telegram.org/bots/api
2. Apache POI – the Java API for Microsoft Documents [Электронный ресурс]. – Режим доступа: https://poi.apache.org/
3. ZXing – Barcode Image Processing Library [Электронный ресурс]. – Режим доступа: https://github.com/zxing/zxing
4. Документация Java Platform, Standard Edition [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/en/java/
5. Spring Boot Reference Documentation [Электронный ресурс]. – Режим доступа: https://docs.spring.io/spring-boot/docs/current/reference/html/
6. Документация PostgresSQL [Электронный ресурс]. – Режим доступа: https://www.postgresql.org/docs/
7. Савельев, С. Н. Современные средства автоматизации бухгалтерского учета / С. Н. Савельев. – М. : Финансы и статистика, 2020. – 192 с.
8. Клименко, И. В. Разработка клиент-серверных приложений на Java : учебное пособие / И. В. Клименко. – М. : ДМК Пресс, 2019. – 248 с.

Разработка методов генерации документов, сопутствующих процессу ГИА ФГБОУ ВО «Воронежского государственного университета»

С. И. Бутовецкая

Студент бакалавр

Е. А. Копытина

Доцент

Введение

Сотрудники образовательных учреждений тратят много времени на выполнение рутинных задач, как показано в [1-4]. В частности, процесс государственной итоговой аттестации (ГИА) сопровождается созданием и учетом множества документов, включая оценочные листы, протоколы заседаний государственной экзаменационной комиссии (ГЭК), результаты аттестации и другие. Каждый из документов имеет стандартный шаблон. Данные, необходимые для заполнения этих бумаг, секретарь ГЭК собирает вручную. Таким образом, подготовка документов, сопутствующих процессу ГИА, состоит в том, чтобы собрать нужную информацию по студентам и корректно вставить в шаблоны.

Такая работа является рутинной и занимает много времени у сотрудника образовательного учреждения. Поэтому автоматизация этого процесса является актуальной и значимой задачей.

Данная статья описывает методы генерации документов для ГИА ФГБОУ ВО «ВГУ», которые предоставят функционал генерации пакета документов по входным данным, тем самым продолжая разработку, описанную в [5]. Такая разработка позволит сократить время на формирование необходимых бумаг.

1. Постановка задачи

Постановка задачи в рамках данной статьи звучит следующим образом: требуется разработать методы, способные автоматизировать процесс заполнения документов для заседания ГЭК. Были выдвинуты следующие требования:

– Разработать функционал по сбору необходимой информации из excel-файла, поступающему на вход алгоритму генерации;

© Бутовецкая С. И., Копытина Е. А., 2025

– Разработать функционал генерации документа «Оценочный лист» с информацией по каждому студенту;
– Разработать функционал генерации документа «Протокол заседания ГЭК» для каждого заседания экзаменационной комиссии;
– Разработать функционал генерации документов «Приложение к протоколу ГЭК о защите» с информацией по каждому студенту на отдельном листе;
– Разработать функционал генерации документа «Приложение о присвоении квалификации выпускникам».

2. Средства реализации программного продукта

Методы генерации документов написаны на объектно-ориентированном языке Java с использованием следующих технологий:

– Gradle – фреймворк для автоматической сборки проекта с описанием зависимостей в файле build.gradle.
– PostgreSQL – объектно-реляционная система управления базами данных (СУБД).
– Spring – фреймворк для создания веб-приложений и Spring Boot – модуль Spring, который осуществляет автоматическую настройку веб-приложений и предоставляет встроенные HTTP-серверы (Tomcat). Spring Security – фреймворк для реализации регистрации и авторизации пользователей веб-приложения [6].
– Hibernate – библиотека для языка программирования Java, цель которой связать объектно-ориентированное программирование и реляционную базу данных, а также сопоставить типы данных Java с типами данных SQL. Hibernate является реализацией JPA [7].
– Aspose.Words – библиотека для работы с документами Microsoft Word, которая предоставляет API для создания, редактирования, конвертации и обработки документов. В частности, класс Cell библиотеки Aspose.Words позволяет программно манипулировать ячейками таблиц в Word-документах, включая их создание, форматирование, заполнение данными и объединение.

3. Реализация логики программного продукта

Разработка методов генерации документов может быть условно разделена на три части: чтение данных из документа Excel, обработка данных и заполнение необходимых документов по шаблонам.

На рис. 1 представлен пример заполненного документа «Защита ВКР». Таблица содержит информацию о теме ВКР, ФИО руководителя,

его оценке и типе темы по каждому студенту. Этот документ заполняется секретарем ГЭК и каждым студентом самостоятельно.

	A	B	C	D	E	F	G	H
1						Темы ВКР		
2	№	ФИО	Тема ВКР (строго как на титульном листе!)	ФИО руководителя	Оценка научного руководителя	выполнена по теме, предложенной студентом	выполнена по заявке предприятия/организации	относится к области фундаментальных и поисковых научных исследований
3	1	Киреева Мария Сергеевна	Разработка и внедрение системы управленческой отчетности на базе 1С: Бухгалтерия 8.3 для компании по производству корпусной мебели	Матвеев М.Г.	5	+		
4	2	Варлыгин Дмитрий Александрови	Повышение качества бизнес-процессов медучреждений с применением рекомендательной системы подбора лекарств на основе LLM с использованием дистилляции знаний	Илларионов Игорь Владимирович	4	+		
5	3	Налетова Елена Михайловна	Разработка системы анализа финансовой состоятельности предприятия малого бизнеса	Копытин Алексей Вячеславович	5	+		
6	4	Воронков Андрей Алексеевич	Управление проектом и распределение ресурсов с помощью моделей и методов сетевого планирования	Алейникова Наталья Александровна	5	+		
7	5	Слободчикова Мария Сергеевна	Разработка сервиса поддержки принятия решений по выбору телефона с использованием эконометрических оценок	Алейникова Наталья Александровна	4	+		
8	6	Черенкова Юлия Сергеевна	Интеграция SAP Hybris с SAP MDG	Илларионов Игорь Владимирович	5	+		
9	7	Долгих Тимофей Андреевич	Разработка аналитической системы вакансий на рынке труда	Илларионов Игорь Владимирович	5	+		
10	8	Лебеда Даниил Александрович	Разработка интернет-магазина на языке Python для компании «Shisha Life»	Алейникова Наталья Александровна	4		+	

Рис. 1. Фрагмент заполненного документа «Защита ВКР»

Документ «Защита ВКР» секретарь загружает в систему. Функция getStudentInfo() сканирует данный документ и собирает информацию по каждому студенту в список класса InfoStudent, который выглядит следующим образом:

```
public class InfoStudent {
    DateTime dateExam;
    String name;
    String themeName;
    String supervisorName;
    String supervisorMark;
    Theme themeType;
    List<Recommendation> recommendations;
}
```

Перед реализацией логики генерации пакета документов был проведен анализ документов, сопутствующих процессу ГИА ФГБОУ ВО «ВГУ». Для заседаний экзаменационной комиссии нужны следующие документы:

1. Оценочный лист;
2. Протокол заседания ГЭК по программе с приложением по защите ВКР по каждому студенту;
3. Протокол заседания ГЭК о присвоение квалификации выпускникам с приложением;

4. Результаты защиты ВКР по направлению.

При анализе внимание уделялось тем местам документа, которые возможно автоматизировать. Например, на рис. 2 представлены проанализированные документы.

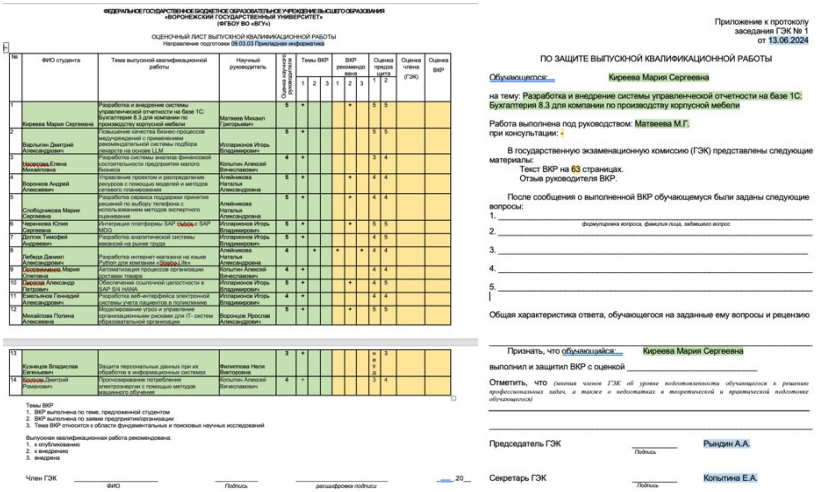


Рис. 2. Анализ документов

Зелёным выделена информация, которая заполняется данными из таблицы Excel «Защита ВКР», синим – информация, переданная на вход методам генерации, а желтым – вручную. Не выделенные части документа формируют его шаблон.

Методы генерации документов, упомянутых выше, похожи друг на друга и отличаются только логикой заполнения шаблона, поэтому рассмотреть функционал можно на примере одного документа.

Функция generation() класса MarkListGeneration реализовывает заполнение документа «Оценочный лист ВКР» по шаблону, который представлен на рис. 3.

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

ОЦЕНОЧНЫЙ ЛИСТ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
Направление подготовки {{direction}}

№	ФИО студента	Тема выпускной квалификационной работы	Научный руководитель	Оценка научного руководителя	Темы ВКР			ВКР рекомендована			Оценка предзащита		Оценка члена (ГЭК)	Оценка ВКР
					1	2	3	1	2	3	1	2		
{{count}}	{{fio}}	{{theme}}	{{fio_head}}	{{mark_head}}	{{theme_flag_1}}	{{theme_flag_2}}	{{theme_flag_3}}							

Темы ВКР
1. ВКР выполнена по теме, предложенной студентом
2. ВКР выполнена по заявке предприятия/организации
3. Тема ВКР относится к области фундаментальных и поисковых научных исследований

Выпускная квалификационная работа рекомендована:
1. к опубликованию
2. к внедрению
3. внедрена

Член ГЭК _____ _____ _____ ___.___.20__
ФИО Подпись расшифровка подписи

Рис. 3. Шаблон документа «Оценочный лист ВКР»

На вход функции передается шаблон документа documentTemplate и информация для заполнения documentInfo. Внутри функции реализовано создание нового документа на основе шаблона и замена плейсхолдеров на актуальные значения средствами библиотеки Aspose.Words:

– Плейсхолдер {{direction}} заменяется на значение infoDocument.direction;
– Плейсхолдеры в таблице заполняются в цикле, представленном ниже.

```
for (InfoStudent student :
info.infoStudents().stream().filter(is ->
is.getDateExam().equals(date)).toList()) {
    Row row = new Row(doc);
    count++;
    for (int i = 1; i < 16; i++) {
        Cell cell = new Cell(doc);
        Paragraph para = new Paragraph(doc);
        switch (i) {
            case 1 -> para.appendChild(new Run(doc,
Integer.toString(count)));
            case 2 -> para.appendChild(new Run(doc,
student.getName()));
            case 3 -> para.appendChild(new Run(doc,
student.getThemeName()));
            case 4 -> para.appendChild(new Run(doc,
student.getSupervisorName()));
            case 5 -> para.appendChild(new Run(doc,
student.getSupervisorMark()));
            case 6 -> para.appendChild(new Run(doc,
```

```
student.getThemeType().isStudentTheme() ? "+" : ""));
            case 7 -> para.appendChild(new Run(doc,
student.getThemeType().isEnterpriseTheme() ? "+" : ""));
            case 8 -> para.appendChild(new Run(doc,
student.getThemeType().isResearchTheme() ? "+" : ""));
            case 9 ->
                    para.appendChild(new Run(doc,
student.getRecommendations().contains(Recommendation.PUBLICATI
ON_RECOMMENDATION) ? "+" : ""));
            case 10 ->
                    para.appendChild(new Run(doc,
student.getRecommendations().contains(Recommendation.IMPLEMENT
ATION_RECOMMENDATION) ? "+" : ""));
            case 11 ->
                    para.appendChild(new Run(doc,
student.getRecommendations().contains(Recommendation.IMPLEMENT
ED) ? "+" : ""));
        }
        cell.appendChild(para);
        row.appendChild(cell);
    }
    table.appendChild(row);
}
```

После отработки описанной выше функции генерируется документ. Фрагмент полученного документа представлен на рис. 4.

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (ФГБОУ ВО «ВГУ»)

ОЦЕНОЧНЫЙ ЛИСТ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
Направление подготовки 09.03.03 Прикладная информатика

№	ФИО студента	Тема выпускной квалификационной работы	Научный руководитель	Оценка научного руководителя	Темы ВКР			ВКР рекомендована			Оценка предзащита		Оценка члена (ГЭК)	Оценка ВКР
					1	2	3	1	2	3	1	2		
1	Киреева Мария Сергеевна	Разработка и внедрение системы управленческой отчетности на базе 1С: Бухгалтерия 8.3 для компании по производству корпусной мебели	Матвеев М.Г.	5	+									
2	Варлыгин Дмитрий Александрович	Повышение качества бизнес-процессов медучреждений с применением рекомендательной системы подбора лекарств на основе LLM с использованием дистилляции знаний	Илларионов Игорь Владимирович	4	+									
3	Назарова Елена Михайловна	Разработка системы анализа финансовой состоятельности предприятия малого бизнеса	Колытин Алексей Вячеславович	5	+									
4	Воронков Андрей Алексеевич	Управление проектом и распределение ресурсов с помощью моделей и методов сетевого планирования	Алейникова Наталья Александровна	5	+									
5	Слободчикова Мария Сергеевна	Разработка сервиса поддержки принятия решений по выбору телефона с использованием эконометрических оценок	Алейникова Наталья Александровна	4	+									
6	Черенкова Юлия Сергеевна	Интеграция SAP Hybris с SAP MDG	Илларионов Игорь Владимирович	5	+									
7	Долгих Тимофей Андреевич	Разработка аналитической системы вакансий на рынке труда	Илларионов Игорь Владимирович	5	+									
8	Лебеда Даниил Александрович	Разработка интернет-магазина на языке Python для компании «Shisha Life»	Алейникова Наталья Александровна	4		+								
9	Пирагов Александр Петрович	Обеспечение ссылочной целостности в SAP S/4 HANA	Илларионов Игорь Владимирович	5	+									
10	Емельянов Геннадий Александрович	Разработка веб-интерфейса электронной системы учета пациентов в поликлинике	Илларионов Игорь Владимирович	5	+									
11	Михайлова Полина Алексеевна	Моделирование угроз и управление организационными рисками для IT-систем образовательной организации	Воронцов Ярослав Александрович	5	+									

Рис. 4. Сгенерированный документ «Оценочный лист ВКР»

Заполнение остальных документов реализовано по аналогии с описанным выше примером.

Заключение

В данной статье была рассмотрена реализация методов генерации документов, сопутствующих процессу ГИА ФГБОУ ВО «ВГУ». Разработанные функциональные возможности соответствует всем поставленным требованиям. В будущем планируется расширение функциональности данного программного продукта.

Список литературы

1. Бордюжа, Е. О. Разработка информационной системы, автоматизирующей генерацию шаблонов документов, сопутствующих прохождению учебных практик ФКН ФГБОУ ВО "ВГУ" / Е. О. Бордюжа, Е. А. Копытина, Н. С. Богданова // Труды молодых учёных факультета компьютерных наук ВГУ : Сборник статей. – Воронеж : Воронежский государственный университет, 2024. – С. 18-25. – EDN ITOFPW.
2. Свидетельство о государственной регистрации программы для ЭВМ № 2023661037 Российская Федерация. Приложение для автоматизации документооборота дополнительного образования : № 2023618813 : заявл. 03.05.2023 : опубл. 25.05.2023 / А. С. Голышева, Е. А. Копытина ; заявитель Федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный технический университет». – EDN NBUVMM.
3. Претков, А. И. Автоматизации формирования документов основных профессиональных образовательных программ высшего образования факультета компьютерных наук Воронежского государственного университета / А. И. Претков, Е. А. Копытина // Труды молодых учёных факультета компьютерных наук ВГУ : Сборник статей. – Воронеж : Воронежский государственный университет, 2023. – С. 94-107. – EDN HKXMIS.
4. Зобнин, Ю. А. Разработка и внедрение автоматизированных информационных систем на предприятии / Ю. А. Зобнин, А. С. Еропкина, О. В. Рындина. – Тюмень : Тюменский индустриальный университет, 2021. – 148 с. – ISBN 978-5-9961-2750-4. – EDN CHNGAD.
5. Бутовецкая, С. И. Разработка информационной системы факультета компьютерных наук ФГБОУ ВО "Воронежский государственный университет" / С. И. Бутовецкая, Е. А. Копытина // Труды молодых учёных факультета компьютерных наук ВГУ : Сборник статей. – Воронеж : Воронежский государственный университет, 2024. – С. 34-41. – EDN XPWOAR.

6. Ишкова, А. Д. Использование микросервисной архитектуры на основе Spring boot и Spring Cloud (Netflix) в реальных проектах / А. Д. Ишкова, Д. Я. Смирнов, О. Г. Яскевич // Интеллектуальные информационные системы : труды Международной научно-практической конференции, Воронеж, 08–10 декабря 2022 года. – Воронеж: Воронежский государственный технический университет, 2023. – С. 141-143.

7. Хемраджани, А. Гибкая разработка приложений на Java с помощью Spring, Hibernate и Eclipse / А. Хемраджани ; Анил Хемраджани ; [пер. с англ. и ред. В. А. Коваленко]. – Москва [и др.] : Вильямс, 2008. – 344 с. – ISBN 978-5-8459-1375-3. – EDN QMSDCD.

Основные направления развития средств анализа уязвимостей информационных систем

Д. М. Высоцкий

Студент бакалавр

С. А. Вялых

Доцент

Введение

Методология защиты современных информационных систем, определяемая нормативно-методическими документами ФСТЭК России [1-5], предполагает обязательное проведение процедур выявления, анализа и устранения уязвимостей как на этапе проектирования и ввода в эксплуатацию информационной системы, так и в процессе ее эксплуатации. При этом под уязвимостью понимается недостаток (слабость) программного (программно-технического) средства или системы и сети в целом, который(ая) может быть использован(а) для реализации угроз безопасности информации [5].

Однако на практике указанное определение часто толкуется, как ФСТЭК России [6], так и многими специалистами в области безопасности информации только в узком смысле – как уязвимость конкретного программного средства.

Кроме этого требования обязательной сертификации средств защиты информации в государственных информационных системах [1], приводят к абсурдному запрету использования иных средств для обнаружения и устранения уязвимостей.

Уязвимости нулевого дня, а также многие возможные уязвимости как системы в целом, так и при использовании конкретных технологий, могут использоваться для реализации различных способов атак и при этом не могут быть обнаружены только сертифицированными средствами сканирования уязвимостей или анализа защищенности.

До сих пор отсутствует системный анализ возможных уязвимостей информационных систем и реализуемых на их основе различных способов атак, реализация которых, в конечном счете, и должна подлежать оценке. Неполное описание предметной области необходимого мониторинга и оценки уязвимостей современных

© Высоцкий Д. М., Вялых С. А., 2025

информационных систем переводит важнейший этап обнаружения уязвимостей в формальную плоскость запуска сертифицированного средства сканирования без должного анализа реализуемых функций и конечному заключению об отсутствии уязвимостей и выполнении требований документов. Как пример - отсутствие описания предметной области необходимого мониторинга и оценки уязвимостей в Руководстве по организации процесса управления уязвимостями в органе (организации) [7], что также позволяет ограничить область анализа и формально выполнить требования документа.

Указанное позволяет утверждать, что выполнение требований нормативных документов ФСТЭК России (на бумаге) не позволяет обеспечить эффективный анализ и устранение уязвимостей и, как следствие, защиту современных информационных систем. На практике это подтверждается статистикой. Согласно отчету компании, Positive Technologies за 2022 – 2024 годы, количество зафиксированных инцидентов демонстрирует постоянный рост: 2021 г. – 2418 случаев, 2022 г. – 2921 случай, 2023 г. – 3286 случаев [8-9].

За первые три квартала 2024 года было зарегистрировано 2830 инцидентов, тогда как за аналогичный период 2023 года их число уже составляло 2414 [8-9]. Динамика роста наглядно отражена на рис. 1.

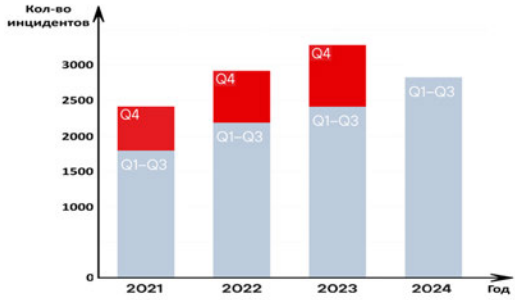


Рис. 1. Рост инцидентов информационной безопасности

1. Сертифицированные средства обнаружения уязвимостей

На практике для поиска уязвимостей, применяются сертифицированные сканеры.

Кроме этого, можно выделить отдельные классы программного обеспечения позволяющие регистрировать инциденты и на этой основе выявлять уязвимости. Это, например, системы обнаружения вторжений (СОВ) и/или системы управления событиями безопасности (SIEM).

Данные системы отличаются стоимостью и трудозатратами на обслуживание.

Любое средство защиты, регистрирующее события безопасности, в своем роде можно считать средством, позволяющим обнаружить инцидент и на его основе выявить уязвимость.

Среди сертифицированных средств обнаружения уязвимостей можно выделить следующие:

1. Программное изделие “Сетевой сканер безопасности XSpider 7.8.25”:

Разработан компанией “Positive Technologies”. Сертифицирован ФСТЭК на срок: 24.10.2014 – 24.10.2025. Выявляет компоненты сети, анализирует их на уязвимости и предоставляет рекомендации по устранению, поддерживает обнаружение уязвимостей OWASP Top 10 на веб-страницах. Также предусмотрен функционал по проверке качества паролей. Начальная цена лицензии составляет 22 477 ₽ и доходит до 6 692 400 ₽.

2. Средство анализа защищенности RedCheck:

Разработан компанией “Алтэкс-Софт”. Сертифицирован ФСТЭК на срок: 23.06.2014 – 23.06.2025. Обеспечивает сканирование сети и обнаружение сетевых уязвимостей, уязвимостей, содержащихся в открытом репозитории OVALdb, проверку безопасности систем управления базами данных, контроль конфигураций, управление обновлениями, оценку соответствия стандартам, формирует детальные отчеты. Начальная цена лицензии составляет 790 ₽ и доходит до 1 410 500 ₽.

3. Программный комплекс “Средство анализа защищенности “Сканер-ВС””:

Разработан компанией “Эшелон”. Сертифицирован ФСТЭК на срок: 24.10.2014 – 13.11.2029. Комплекс для тестирования защищенности информационных систем, включая сетевой аудит, поиск уязвимостей, анализ паролей, перехват трафика и аудит обновлений ОС. Начальная цена лицензии составляет 15 000 ₽ и доходит до 3 500 000 ₽.

4. Система анализа защищенности программного и аппаратного обеспечения TCP/IP сетей “сетевой сканер Ревизор Сети версия 3.0”:

Разработан компанией “Профиль Защиты”. Сертифицирован ФСТЭК на срок: 02.06.2015 – 02.06.2023. Техническая поддержка действует до 02.06.2026. Средство для анализа защищенности сетей, выявления уязвимостей и проверки соответствия стандартам безопасности. Включает функции сканирования сетевых узлов, аудита конфигураций и анализа трафика. Начальная цена лицензии составляет 1 800 ₽ и доходит до 138 862 ₽.

5. Программное изделие “Средство обеспечения безопасности информационных систем MaxPatrol” (SIEM):

Разработан компанией “Positive Technologies”. Сертифицирован ФСТЭК на срок: 12.04.2017 – 12.04.2025. Средство для мониторинга событий информационной безопасности, управления уязвимостями и аудита. Включает функции сканирования, анализа рисков и формирования отчетов.

6. Программный комплекс управления конфигурациями и анализа защищенности «Efros Config Inspector» v.4:

Разработан компанией “Газинформсервис”. Сертифицирован ФСТЭК на срок: 14.01.2021 – 14.01.2026. Средство для анализа конфигураций систем и сетевого оборудования на соответствие стандартам безопасности. Позволяет выявлять уязвимости.

7. Программное изделие “Система статического и динамического анализа для выявления вредоносных объектов Positive Technologies Sandbox”:

Разработан компанией “Positive Technologies”. Сертифицирован ФСТЭК на срок: 25.10.2022 – 25.10.2027. Средство для анализа подозрительных файлов и объектов на наличие вредоносного кода. Поддерживает поиск уязвимостей. Использует методы статического и динамического анализа в изолированной среде.

8. Программное изделие «Kaspersky Endpoint Security для Windows» (существуют также отдельные версии для Linux, Mac, Android):

Разработан компанией “Лаборатория Касперского”. Сертифицирован ФСТЭК на срок: 22.01.2019 – 22.01.2029. Комплексное решение для защиты конечных точек от вирусов и других угроз. Включает антивирус, защиту от сетевых атак и контроль устройств. Начальная цена лицензии составляет 15 610 ₽ и доходит до 442 233 ₽.

9. Средство защиты информации Secret Net Studio (СОВ):

Разработан компанией “Код Безопасности”. Сертифицирован ФСТЭК на срок: 16.05.2017 – 16.05.2025. Средство для защиты информации от несанкционированного доступа, включая контроль доступа, шифрование данных и аудит безопасности. Начальная цена лицензии составляет 5 671 ₽ и доходит до 18 281 ₽ за одно устройство.

10. Программный комплекс “Система управления событиями безопасности “RuSIEM”” (SIEM):

Разработан компанией “РуСИЕМ”. Сертифицирован ФСТЭК на срок: 12.05.2021 – 12.05.2026. Система для управления событиями безопасности, включающая функции мониторинга, анализа и реагирования на инциденты.

Основными недостатками перечисленных средств обнаружения уязвимостей является то, что они охватывают в основном только программные уязвимости и уязвимости сетевых протоколов, а также имеет высокую стоимость.

2. Возможные способы атак

Анализ возможных способов атак на современные информационные системы позволяет выделить следующие основные направления.

1. Социальная инженерия:

Любая атака, которая использует человеческую психологию для воздействия на цель, заставляя ее либо выполнить нужное действие, либо предоставить секретную информацию [10].

Данная группа включает в себя такие способы атак, как: различные виды фишинга, вишинг, способы, входящие в группу физической социальной инженерии (бейтинг, копание в мусоре, подглядывание через плечо) [10].

2. Эксплуатация программных уязвимостей:

Подразумевает под собой использование ошибок в программном обеспечении для получения несанкционированного доступа.

Способы атак в данной группе отличаются друг от друга по выбору типа программных уязвимостей для эксплуатации.

3. Сетевые атаки:

Направлены на получение доступа через сетевые подключения.

Данная группа включает в себя такие способы атак, как: атаки типа отказ в обслуживании, MITM-атаки, сканирование сети, сниффинг, атаки на сетевые устройства, атаки на VPN.

4. Физическое воздействие:

Предполагает непосредственный доступ нарушителя к оборудованию.

Данная группа включает в себя такие способы атак, как: несанкционированный доступ к оборудованию, перехват данных через физические каналы, подслушивание и визуальное наблюдение, использование периферийных устройств.

Подробная классификация возможных способов атак на информационные системы приведена на рис. 2. Красным цветом выделены возможные атаки, с использованием уязвимостей, которые могут обнаруживаться сертифицированными средствами защиты, оранжевым – те, для которых необходимо проводить дополнительные, специальные настройки и эксперименты средств сканирования.

Результаты проведенного анализа показывают возможности проведения атак с использованием уязвимостей, не обнаруживаемых

сертифицированными средствами защиты, включая широкие возможности их комбинирования.

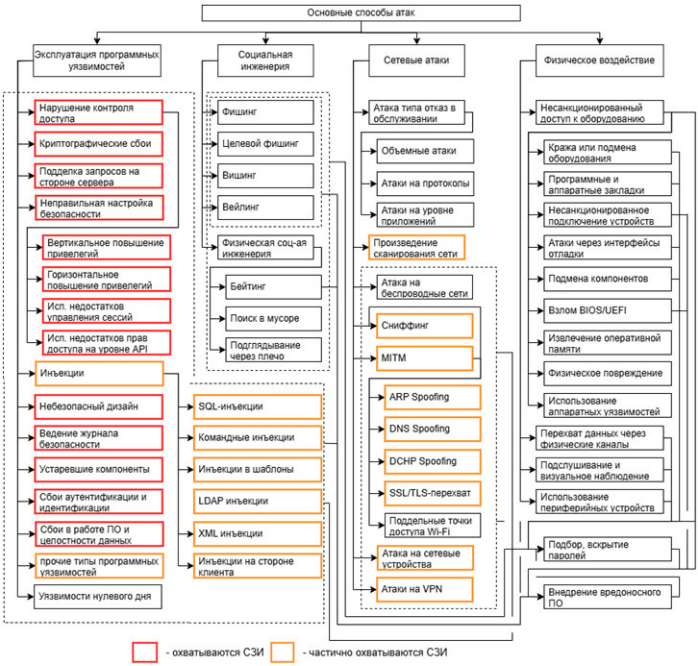


Рис. 2. Классификация возможных способов атак

3. Возможность использования современных средств защиты информации для предотвращения реализации способов атак

Как уже было показано выше современные сертифицированные средства обнаружения уязвимостей охватывают в основном программные уязвимости и уязвимости сетевых протоколов.

При поиске программных уязвимостей средства защиты информации ориентируются на сформированные базы данных угроз. Стоит учитывать, что добавление новых найденных уязвимостей в базу данных занимает некоторое время. Поэтому иногда достаточно только сообщения об обнаруженной некоторой новой уязвимости, чтобы нарушитель смог ей воспользоваться.

Также существует понятие уязвимостей нулевого дня – это уязвимости программного обеспечения, о которых разработчики еще не знают, следовательно, они не могут о них сообщить или их устранить. В

следствии этой специфики данных уязвимостей, они не могут быть включены в базы данных сканеров, следовательно, не могут быть ими обнаружены, но могут быть использованы нарушителями.

Некоторые средства защиты информации могут выявлять слабые места в сетевой структуре. Это может частично помочь в предотвращение некоторых сетевых атак, однако не гарантирует полной безопасности. Например, для противодействия угрозам в беспроводных сетях требуется их глубокий анализ, доступный лишь через специализированные инструменты вроде Aircrack-ng, которые не являются сертифицированными. Эта ситуация дает преимущество злоумышленникам, неограниченным нормативными требованиями.

Средства защиты не охватывают социальную инженерию из-за её зависимости от человеческого фактора. Для противодействия таким атакам необходимы обучение сотрудников и специализированные инструменты. Ключевой элемент безопасности – проверка способности персонала распознавать и предотвращать манипуляции злоумышленников.

Также средства сканирования не выявляют уязвимости, связанные с физическими воздействием на программно-аппаратные элементы построения информационной системы, так как они не предназначены для оценки защиты на физическом уровне.

Все указанные факторы дают преимущество нарушителю, целенаправленно использующему предметно-ориентированные средства, а также уязвимости нулевого дня, которые еще не зарегистрированы в базах сертифицированных сканеров, и/или применяющему способы социальной инженерии или физического воздействия для проведения атак.

Заключение

Результаты проведенного анализа указывают на необходимость доработки существующей методологии выявления, анализа и устранения уязвимостей путем системного описания возможных атак, и типовых правил применения средств тестирования, которые бы обеспечили наиболее полный охват обнаружения возможных уязвимостей в современных системах. В ряде случаев приемлемым выглядит подход, признающий невозможность (или отсутствие необходимых ресурсов) оперативного устранения возможных уязвимостей информационной системы и смещения парадигмы защиты на дополнительную, в т.ч. эшелонированную защиту конкретных свойств информации и/или реагирование на инциденты безопасности и устранение их последствий.

Список литературы

1. Приказ от 11.02.2013 № 17 «Об утверждении Требований о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах».
2. Приказ от 18.02.2013 № 21 «Об утверждении Состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных».
3. Приказ от 14.03.2014 № 31 «Об утверждении Требований к обеспечению защиты информации в автоматизированных системах управления производственными и технологическими процессами на критически важных объектах, потенциально опасных объектах, а также объектах, представляющих повышенную опасность для жизни и здоровья людей и для окружающей природной среды»
4. Приказ от 29.04.2021 № 77 «Об утверждении Порядка организации и проведения работ по аттестации объектов информатизации на соответствие требованиям о защите информации ограниченного доступа, не составляющей государственную тайну».
5. Методический документ. Методика оценки угроз безопасности информации, утверждена ФСТЭК России 05.02.2021.
6. Регламент включения информации об уязвимостях программного обеспечения и программно-аппаратных средств в банк данных угроз безопасности информации ФСТЭК России, утвержден ФСТЭК России 26.06.2018.
7. Методический документ. Руководство по организации процесса управления уязвимостями в органе (организации), утвержден ФСТЭК России от 17.05.2023.
8. Positive Technologies [Электронный ресурс] : Актуальные киберугрозы: итоги 2022 года. – Режим доступа: https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2022/
9. Positive Technologies [Электронный ресурс] : Актуальные киберугрозы: III квартал 2024 года. – Режим доступа: https://www.ptsecurity.com/ru-ru/research/analytics/aktualnye-kiberugrozy-iii-kvartal-2024-goda/#id1
10. Грей, Дж. Социальная инженерия и этичный хакинг на практике / Дж. Грей. – М: ДМК Пресс, 2023. – 226 с.

Извлечение ключевых слов из русскоязычных текстов с использованием Apache OpenNLP

А. Э. Галимов

Студент бакалавр

Н. К. Самойлов

Старший преподаватель

Введение

Извлечение ключевых слов является одной из фундаментальных задач обработки естественного языка (NLP), используемой в информационном поиске и аннотировании текстов.

Apache OpenNLP — это библиотека для обработки естественного языка (NLP), которая содержит модели машинного обучения для решения различных задач NLP.

Работа с русскоязычными текстами является сложной задачей в силу богатой морфологии русского языка, изменяемости окончаний и сложной структуры предложений.

Данная статья посвящена рассмотрению процесса извлечения ключевых слов с помощью Apache OpenNLP и подходам, используемым для достижения этой цели.

1. Модули Apache OpenNLP

Библиотека OpenNLP построена на модульной архитектуре, где каждая NLP-задача решается отдельным компонентом [1].

Эти модули могут работать независимо и комбинироваться под решение необходимой задачи анализа текста.

Для выполнения задачи извлечения ключевых слов из русскоязычных текстов требуется последовательное выполнение следующих этапов [2]:

- Токенизации текста
- POS-теггинга
- Нормализации
- Выбора ключевых слов

Основные модули OpenNLP приведены в таблице.

© Галимов А. Э., Самойлов Н. К., 2025

Таблица

Модули Apache OpenNLP

Название модуля	Выполняемая функция
Tokenizer	Выполняет разбиение текста на отдельные слова и знаки препинания
POS Tagger	Определяет части речи для каждого слова в тексте
Named Entity Recognizer (NER)	Идентифицирует именованные сущности
Parser	Анализирует синтаксическую структуру предложения.
Lemmatizer	приводит слова к их начальной форме

2. Токенизация

Токенизация является одним из ключевых этапов обработки естественного языка и представляет собой разбиение текста на отдельные компоненты — токены. В зависимости от задачи, токенами могут быть слова, знаки препинания или даже символы. Этот этап важен, так как POS-теггинг требует работы с уже выделенными токенами, а не с исходным необработанным текстом.

В Apache OpenNLP используется подход к токенизации на основе регулярных выражений.

Регулярные выражения представляют собой формальный язык для задания шаблонов поиска в тексте.

Их основа — конечные автоматы. Согласно теореме Клина, для любого регулярного выражения можно построить детерминированный конечный автомат (ДКА), который сможет распознавать заданные шаблоны, и наоборот.

Алгоритм работы конечного автомата в токенизаторе:

- Инициализация – создается список активных состояний, в который добавляется начальное состояние.
- Обход текста – анализируется текущий символ и сопоставляется с возможными переходами автомата.
- Проверка эпсилон-переходов.
- Фиксация токенов – когда автомат достигает состояния, означающего конец слова, выделяется токен.
- Повторение – алгоритм продолжается, пока весь текст не будет обработан.

3. Работа POS-теггера

POS-теггинг (Part-of-Speech Tagging) – это процесс автоматической разметки частей речи в тексте [3]. Формально, каждое слово из входного текста должно быть сопоставлено с тегом, который представляет его часть речи. Например: существительное (NOUN), глагол (VERB), прилагательное (ADJ).

В OpenNLP POS-теггер использует предобученные модели, которые основаны на размеченных корпусах текстов. Корпус текста — это структурированный набор текстов, подготовленный по определённым правилам для анализа языка и статистических исследований. POS-теггинг в OpenNLP реализуется с помощью статистических методов машинного обучения: метода максимальной энтропии и метода, основанного на скрытых марковских моделях.

Модель максимальной энтропии основывается на принципе, согласно которому из всех возможных распределений вероятностей предпочтение отдается тому, которое имеет максимальную энтропию при заданных ограничениях [4]. Проще говоря — это способ построения модели, которая делает минимальные предположения, основываясь на имеющихся данных, и при этом остается устойчивой для новых данных.

Далее приведена классификационная формула (1) метода максимальной энтропии.

$$P(c \mid d, \lambda) = \frac{\exp \sum_i \lambda_i f_i(c_i, d)}{\sum_{\tilde{c} \in C} \exp \sum_i \lambda_i f_i(\tilde{c}_i, d)}, \quad (1)$$

где f_i - классификационный индикатор (0 или 1), λ_i - вес классификационного индикатора f_i, c - класс-гипотеза, C - множество всех возможных классов, d - классифицируемый документ.

Данная формула отражает вероятность присвоения тегу слова в контексте документа.

Обучение и работа модели заключается в подборе таких весов, которые бы максимизировали вероятность правильного тегирования на обучающем корпусе.

Скрытая марковская модель – статистическая модель, в которой система описывается как марковский процесс с неизвестными (скрытыми) состояниями.

В контексте POS-теггинга скрытые состояния соответствуют тегам частей речи, а наблюдаемые состояния — словам в тексте. В том числе учитывается последовательность слов. Далее приведена формула (2) для вычисления вероятности наблюдаемой последовательности слов.

$$P(O \mid \lambda) = \sum_{S} P(O \mid S, \lambda) P(S \mid \lambda), \quad (2)$$

где $\lambda = (\pi, A, B)$ - параметры модели, S - возможные последовательности скрытых состояний, $P(S \mid \lambda)$ - вероятность набора конкретной последовательности S, $P(O, S \mid \lambda)$ - вероятность получения последовательности слов O.

Данное выражение определяет полную вероятность наблюдаемой последовательности, независимо от конкретного пути через скрытые состояния (алгоритм прямого – обратного хода) [5].

Далее необходимо осуществить определение наиболее вероятной последовательности частей речи для заданной последовательности слов с помощью алгоритма Витерби.

4. Нормализация

При обработке текстов на русском языке важным этапом является нормализация. Нормализация — это процесс приведения словоформ к их базовой или стандартной форме для того, чтобы унифицировать данные.

Также, на данном этапе, исключаются слова, относящиеся к стоп-списку.

В русском языке нормализация делится на два этапа: лемматизацию и стемминг.

Стемминг — это процесс отсечения окончаний, суффиксов, префиксов от слов с целью выделения их основы. Стемминг не учитывает лексическое значение слова и может приводить к искусственным или несуществующим словам, например:

– «бегать», «бегал», «бегающий» преобразуется в «бег»
– «учёный», «учёные», «учёного» преобразуется в «учён»

Для стемминга русскоязычных текстов широко используется морфологический анализатор MyStem, разработанный компанией Яндекс.

Лемматизация — это более точный метод, чем стемминг, поскольку он приводит слово к его начальной словарной форме (лемме), учитывая его часть речи и грамматические характеристики, например:

– «бегал» преобразуется в «бегать»
– «учёные» преобразуется в «учёный»

Важной особенностью является необходимость использования собственных словарей, которые содержат соответствия между словоформами и их леммами.

5. Выбор ключевых слов

На данном этапе у нас уже есть набор нормализованных слов, отфильтрованных по частям речи, с акцентом на существительные, которые наиболее информативны при анализе текстов.

К сожалению, Apache OpenNLP не предоставляет встроенных функций для определения значимости слов. Для выполнения этой задачи используется метод частотного анализа или алгоритм TF-IDF.

TF-IDF используется для оценки, какую важность имеет слово в тексте по сравнению с другими текстами в корпусе. С помощью этого подхода можно исключить часто встречающиеся, но незначимые слова.

Далее приведена формула (3) для подсчета показателя TF-IDF.

$$\omega_{ij} = tf_{i,j} \log \frac{N}{df_i}, \quad (3)$$

ω_{ij} - показатель TF-IDF, $tf_{i,j}$ - частота появления слова в документе (TF), N - общее количество документов в корпусе текста, $\log \frac{N}{df_i}$ - обратная частота документа (IDF).

Обобщая, показатель TF-IDF растет, если слово часто встречается в рамках одного документа и при этом встречается редко во всем наборе документов.

При частотном анализе частота встречаемости слова в тексте соотносится с их важностью. Для этого строится частотный словарь, где каждому уникальному слову сопоставляется число его вхождений в текст. Далее слова ранжируются по убыванию частоты, и отбираются наиболее часто встречающиеся. Метод подходит для выделения ключевых слов в небольших текстах, где значимые слова встречаются чаще, чем второстепенные.

Заключение

В данной работе рассмотрены методы извлечения ключевых слов из русскоязычных текстов с использованием библиотеки Apache OpenNLP. Основное внимание уделено предобработке данных: токенизации, POS-теггингу, нормализации и фильтрации по частям речи. Важную роль в выделении значимых терминов играет сочетание частотного анализа и TF-IDF.

Хотя библиотека Apache OpenNLP не является самой передовой технологией на данный момент (в силу развития нейросетевых методов), она предоставляет хорошие базовые методы обработки естественного языка. Поэтому и по сей день Apache OpenNLP является полезным инструментом для обучения и исследований, особенно в

контексте работы с русскоязычными текстами, где важно учитывать специфические языковые особенности.

Список литературы

1. Apache OpenNLP Documentation [Электронный ресурс] : официальная документация. – Режим доступа: https://opennlp.apache.org/docs/

2. Analyzing Text with Apache OpenNLP [Электронный ресурс] : блог. – Режим доступа: https://www.softobotics.com/blogs/analyzing-text-with-apache-opennlp/

3. Part-of-speech tagging [Электронный ресурс] : статья. – Режим доступа: https://en.wikipedia.org/wiki/Part-of-speech_tagging/

4. Maximum Entropy Classifier [Электронный ресурс] : блог. – Режим доступа: https://www.bazhenov.me/blog/2013/04/23/maximum-entropy-classifier.html

5. Применение библиотеки Apache OpenNLP для анализа текстов [Электронный ресурс] : статья. – Режим доступа: https://habr.com/ru/articles/837324/

Оценка риска информационной безопасности систем обработки информации с использованием нечетких продукционных когнитивных карт

А. А. Гончарова

Студент бакалавр

В. Ю. Храмов

Доцент

Введение

В настоящее время практически все организации сталкиваются с необходимостью оценки риска для снижения количества опасных событий и достижения поставленных целей.

Цели организации могут затрагивать различные аспекты ее деятельности: от стратегии до выпуска конкретной продукции, разработки процессов и проектов. Цели могут быть определены в социальной, экологической, технологической, коммерческой, финансовой и экономической областях, а также в области репутации организации, ее безопасности и социального, культурного, политического воздействия на население[1].

Всей деятельности организации соответствует риск. Менеджмент риска помогает в принятии решений в условиях неопределенности и возможности возникновения событий или обстоятельств (плановых и непредвиденных), воздействующих на достижение целей организации.

Оценка риска является частью процесса менеджмента риска и представляет собой структурированный процесс, в рамках которого идентифицируют способы достижения поставленных целей, проводят анализ последствий и вероятности возникновения опасных событий для принятия решения о необходимости обработки риска.

Данная работа посвящена методике оценки риска с помощью нечетких продукционных когнитивных карт.

1. Методика оценки риска информационной безопасности

Методика оценки рисков информационной безопасности строится с использованием нечетких продукционных когнитивных карт, функций принадлежности нечетких множеств и матриц риска, а также нечеткого логического вывода с использованием алгоритма, основанного на

© Гончарова А. А., Храмов В. Ю., 2025

теореме гипотез (формуле Байеса)[2-5]. Методика включает шесть шагов.

Шаг 1. Построение НПКК.

Шаг 2. Построение ФП для термов лингвистических переменных, характеризующих переменные состояния концептов НПКК.

В качестве форм задания термов ФП используются треугольные или трапециевидные ФП.

Термы могут принимать следующие значения: L – Low (Низкая (-ий)); M – Medium (Средняя (-ий)); MH – Medium High (Довольно высокая (-ий)); H – High (Высокая (-ий)); VH – Very High (Очень высокая (-ий)).

При построении ФП терм лингвистической переменной X должен удовлетворять условиям (1)-(5).

$$\mu_{C_1}(x_1) = 1, \mu_{C_m}(x_1) = 1. \tag{1}$$

$$\left(\forall T_i \in T \setminus \{T_m\}\right)\left(0 < \sup_{x \in X} \mu_{C_i \cup C_{i+1}}(x) < 1\right). \tag{2}$$

$$\left(\forall T_i \in T\right)\left(\exists x \in X\right)\left(\mu_{C_i}(x) = 1\right). \tag{3}$$

$$\left(\forall \beta\right)\left(\exists x_1 \in R_1\right)\left(\exists x_2 \in R_2\right)\left(\left(\forall x \in X\right)\left(x_1 < x < x_2\right)\right). \tag{4}$$

$$\begin{aligned}&\left(\forall i \in I, i \neq 1 \wedge i \neq m\right)\left(card\left(T_{i-1} \cap T_{i+1}\right) = 0 \vee\right.\\ &\left.\vee\left(\left(card\left(T_{i-1} \cup T_{i+1}\right) = 1\right) \wedge \mu\left(T_{i-1} \cap T_{i+1}\right) = 0\right)\right)\end{aligned}. \tag{5}$$

Шаг 3. Построение системы нечетких продукционных правил (матриц риска), описывающих состояние концептов C_i.

Правила записываются в форме продукций ЕСЛИ a, ТО b, где a – условие, b – заключение.

Вычисление коэффициента определенности заключения $ct(b)$ осуществляется по формуле:

$$ct(b) = ct(a) * ct(правила). \tag{6}$$

где $ct(a)$ – коэффициент определенности условия ($ct(a) \in [0,1]$), $ct(правила)$ – коэффициент определенности правила ($ct(правила) \in [0,1]$).

По умолчанию матрицы риска формируются и заполняются автоматически по алгоритму построения матриц риска.

Далее эксперт по своему усмотрению может внести изменения в автоматически сформированные матрицы риска, если того требует рассматриваемая задача.

Шаг 4. Задание четких значений входных переменных состояния концептов и построение схемы нечеткого логического вывода.

Шаг 5. Нахождение значений промежуточных переменных состояния концептов и переменной риска по алгоритму логического вывода.

Шаг 6. Вычисление значения выходной переменной риска по формуле метода взвешенного среднего:

$$X_R = \Pi_{риск} = \frac{\sum_{i=1}^{m} \alpha_i * X_{R_i}^0}{\sum_{i=1}^{m} \alpha_i}. \quad (7)$$

где $X_{R_i}^{\ 0}$ (i = 1, 2, …, m) – центральные значения (центр тяжести) нечетких подмножеств (термов) переменной X_R, α_i – уровень активности i-го правила – значение степени принадлежности i-го терма переменной X_R, m – число термов лингвистической переменной X_R.

2. Пример оценки риска информационной безопасности без мер защиты

В качестве примера реализации методики оценки рисков информационной безопасности, допустим, что требуется оценить риск возможного воздействия вирусной атаки на информационный ресурс, размещаемый на сервере локальной вычислительной сети АСУ, рассматривая в качестве уязвимости отсутствие обновлений антивирусного программного обеспечения.

Шаг 1. Используя трехфакторную формулу риска:

$$R = P_{угроза} * P_{уязвимость} * P_{ценность_ресурса}. \quad (8)$$

представим соответствующую ей схему расчета в виде НПКК на рисунке 1, где C_1 – угроза, C_2 – уязвимость, C_3 – информационный ресурс, C_4 – реализация угрозы, C_5 – риск (потенциальный ущерб); соответственно X_1 – вероятность возникновения угрозы, X_2 – вероятность наличия уязвимости, X_3 – ценность (стоимость) информационного ресурса, X_4 – вероятность успешной реализации угрозы, X_5 = R – уровень риска (значение ожидаемого потенциального ущерба от специальных программных воздействий).

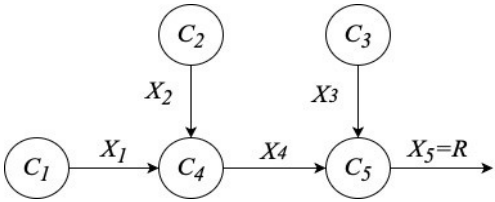


Рис. 1. Схема НПКК для оценки рисков

Шаг 2. Используя аппарат нечеткой логики, можно полагать, что каждая из указанных переменных состояния представляет собой лингвистическую переменную, принимающую одно из следующих значений: L – Low (Низкая (-ий)); M – Medium (Средняя (-ий)); MH – Medium High (Довольно высокая (-ий)); H – High (Высокая (-ий)); VH – Very High (Очень высокая (-ий)). Три входных и конечное нечеткое подмножество задаются, в свою очередь, собственной функцией принадлежности, построенной экспертным путем, вид которых показан на рисунке 2 с учетом ограничений.

Допустим, что в конкретном рассматриваемом случае входные переменные НПКК (т.е. три базовых фактора риска) принимают значения: $X_1^* = 0{,}85,\ X_2^* = 0{,}9,\ X_3^* = 0{,}75$.

Шаг 3. Для описания состояния концептов и можно записать систему нечетких продукционных правил в виде:

концепт C_4:

П1: Если X_1 есть Низкая и X_2 есть Низкая, то X_4 есть Низкая;

...

П25: Если X_1 есть Очень_высокая и X_2 есть Очень_высокая, то X_4 есть Очень_высокая;

концепт C_5:

П26: Если X_3 есть Низкая и есть X_4 Низкая, то X_5 есть Низкая;

...

П50: Если X_3 есть Очень_высокая и X_4 есть Очень_высокая, то X_5 есть Очень_высокая.

Получаем 2 * 5 * 5 = 50 правил, которые представляются в виде матриц риска или таблиц решений, $ct(правил) = 1$ (рисунок 3).

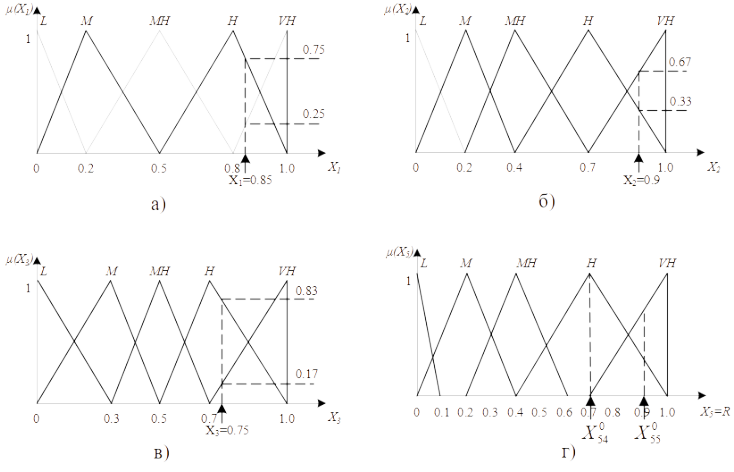


Рис. 2. Функции принадлежности нечетких множеств: а – угроза, б – уязвимость, в – ценность ресурса, г – риск

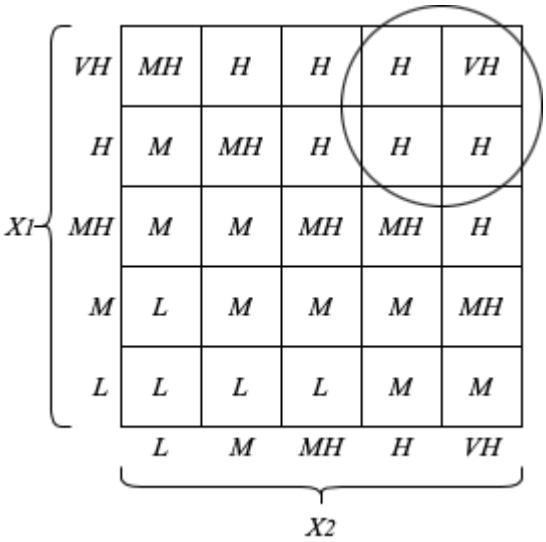


Рис. 3. Матрицы риска: а – реализация угрозы, б – риск

Обратившись к рисункам 2 и 3, видим, что переменные X_1, X_2, X_3, X_4 принимают только значения H и VH, т. е. из 50 правил

активными окажутся только 8 правил, соответствующие выделенным блокам из четырех соседних клеток в правом верхнем углу матриц риска.

Шаг 4. Для принятых значений входных переменных схема нечеткого логического вывода имеет вид, представленный на рисунке 4.

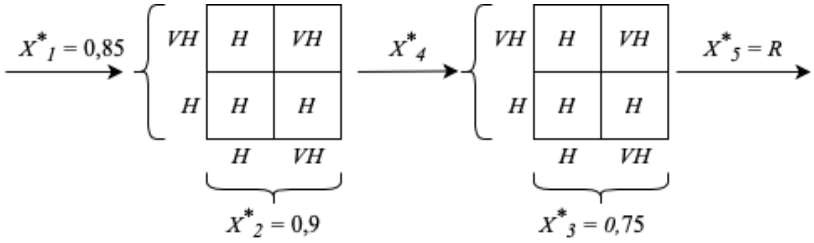


Рис. 4. Схема нечеткого логического вывода для оценки риска

Шаг 5. Вычислим значение промежуточной переменной состояния X_4 концепта C_4 по алгоритму логического вывода.

Продукционные правила для концепта C_4 имеют следующий вид:

Если X_1 = H и X_2 = H, то X_4 = H;

Если X_1 = H и X_2 = VH, то X_4 = H;

Если X_1 = VH и X_2 = H, то X_4 = H;

Если X_1 = VH и X_2 = VH, то X_4 = VH.

По функциям принадлежности (рисунок 2 а, б) определяем значения $\mu(x_1^*)$ и $\mu(x_2^*)$, которые соответственно равны <<0,75/H>, <0,25/VH>> и <<0,33/H>, <0,67/VH>>.

Далее в соответствии с алгоритмом для каждого правила применяется операция MIN:

правило 1: min (0,75; 0,33) = 0,33;

правило 2: min (0,75; 0,67) = 0,67;

правило 3: min (0,25; 0,33) = 0,25;

правило 4: min (0,25; 0,67) = 0,25.

Так как $ct(правил) = 1$, то полученные значения не изменяются.

Для одинаковых наименований термов продукционных правил, стоящих в заключении продукций (H (Высокая) и VH (Очень_высокая)) применяется операция MAX:

max (0,33; 0,67; 0,25) = 0,67;

max (0,25) = 0,25.

Таким образом, переменная состояния X_4 концепта C_4 принимает следующее нечеткое значение:
<<0,67 / H>, <0,25 / VH> / вероятность успешной реализации угрозы>.

Аналогичным образом осуществляется вычисление переменной состояния концепта (значение потенциального ущерба):
<<0,67 / H>, <0,17 / VH> / потенциальный ущерб>.

Шаг 6. Вычислим центральные значения переменных риска $X_{55}{}^0$ (Очень высокий) и $X_{54}{}^0$ (Высокий) (рисунок 2 г).

Используя алгоритм определения центральных значений термов функций принадлежности, получим, что $X_{55}{}^0 = 0{,}91$ и $X_{54}{}^0 = 0{,}7$.

Применяя формулу (7), вычисляем дефазифицированное («четкое») значение переменной уровня риска.

Следовательно, искомое значение ожидаемого потенциального ущерба составит 74%.

Заключение

Данная статья посвящена методике оценки риска с помощью нечетких продукционных когнитивных карт. Описан алгоритм оценки риска, а также пример его использования. Метод может быть особенно полезен в ситуациях дефицита точных данных, когда традиционные вероятностные подходы оказываются недостаточно гибкими. Он позволяет сочетать экспертные оценки с математическим моделированием. Это повышает точность прогнозирования и снижает субъективность в оценке рисков.

В дальнейшем предполагается использование данной методики для разработки программного обеспечения для оценки риска с помощью НПКК.

Список литературы

1. Марков, А. Управление рисками – нормативный вакуум информационной безопасности / А. Марков, В. Цирлов // Открытые системы. СУБД: Журнал для профессионалов в области информационных технологий. – 2007. – №8. – С. 63-67.

2. Конышева, Л.К. Основы теории нечетких множеств: Учебное пособие / Л.К. Конышева, Д.М. Назаров. – СПб.: Питер, 2011. – 192 с.

3. Борисов, В. В. Анализ динамики состояния сложных систем на основе обобщенных нечетких продукционных когнитивных карт / В.В. Борисов, А.С. Федулов, Е.С. Устиненков // Нейрокомпьютеры: разработка, применение. - 2007. - № 1. - С. 17-23.

4. Гузаиров, М. Б. Системный анализ информационных рисков с применением нечетких когнитивных карт / М. Б. Гузаиров, В. И.

Васильев, Р. Т. Кудрявцева // Инфокоммуникационные технологии. - 2007. - Т. 5. - № 4. - С. 42-48.

5. Васильева, В. И. Оценка рисков информационной безопасности с использованием нечетких продукционных когнитивных карт / В. И. Васильева, А. М. Вульфина, М. Б. Гузаирова // Информационные технологии. - 2018. - Т. 24. - № 4. - С. 266-272.

Разработка серверного приложения для сбора рыночных исследований

Р. Р. Джаббаров

Студент бакалавр

Е. А. Копытина

Доцент

Введение

Крупные и средние предприятия всё чаще сталкиваются с необходимостью внешнего финансового анализа или бенчмаркинга для оценки своего текущего положения на рынке по сравнению с конкурентами. Ручное проведение таких исследований связано с рядом проблем: высокой трудоёмкостью, риском ошибок и низким уровнем актуальности данных.

Целью данной работы заключается в разработке серверного приложения, автоматизирующего процесс сбора, анализа рыночных исследований и генерации финансовых отчётов в формате Excel, что позволит ускорить и упростить работу аналитика. Аналогичные подходы к решению задач с генерацией документов были подробно рассмотрены в [1-6]

1. Постановка задачи

К разрабатываемому приложению были выдвинуты следующие функциональные требования:

– Автоматизация сбора анкет от предприятий;
– Экспорт анкет в Excel-файл;
– Импорт анкет из Excel-файла в БД приложения;
– Автоматизированная проверка и валидация анкетных данных;
– Генерация персональных финансовых отчётов для каждого предприятия;
– Для аналитика реализовать возможность создания финансовых исследований.

2. Реализация логики

Для реализации серверного приложения был выбран следующий ряд технологий:

© Джаббаров Р. Р., Копытина Е. А., 2025

– Java – объектно-ориентированный язык программирования, предназначенный для разработки различных приложений, включая корпоративные, и серверные, обеспечивая быстродействие, безопасность и надежность [7];

– Spring Framework – это фреймворк с открытым исходным кодом для языка программирования Java. Он был создан для упрощения разработки и поддержки масштабируемых, слабосвязанных и повторно используемых приложений;

– PostgreSQL – реляционная база данных с открытым кодом, которая поддерживается в течение 30 лет разработки и является одной из наиболее известных среди всех существующих реляционных баз данных;

– Apache POI – Java-библиотека для работы с файлами Microsoft Office, включая Excel. Позволяет программно создавать, редактировать и читать электронные таблицы, что необходимо для генерации финансовых отчётов и обработки импортируемых анкет;

– MinIO – это высокопроизводительная система объектного хранения с открытым кодом, которая совместима по API с облачным сервисом хранения Amazon S3.

Основой разработанного приложения являются следующие доменные сущности:

– Research (Исследование) – шаблон анкеты, представленный в JSON-формате, поддерживающий версионность и возможность дополнения;

– Research Answer (Ответы на исследования) – массив ответов предприятий для конкретного исследования;

– Dataset (Бенчмаркинг) – шаблон бенчмаргинга, представленный в JSON-формате, представляет собой набор агрегирующих формул, совместим с конкретным исследованием;

– Dataset Answer (Ответы на бенчмаркинг) – массив вычисленных метрик для конкретного бенчмаркинга.

ER-диаграмма с данными доменами представлен на рис. 1.

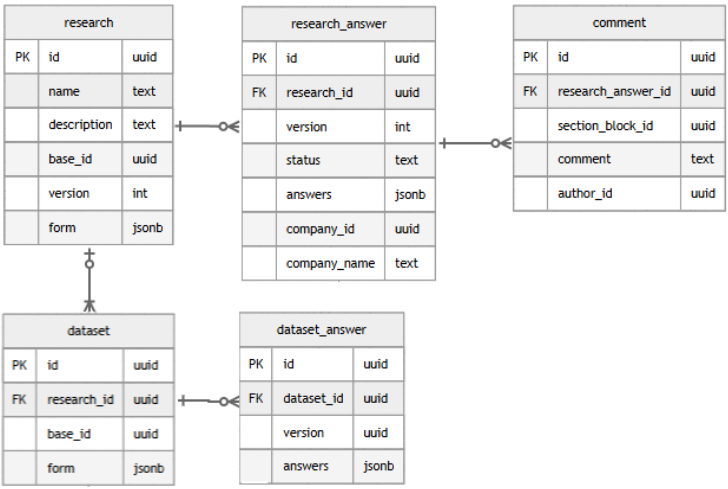


Рис. 1. ER-диаграмма

Шаблоны анкеты и бенчмаркинга представлен иерархической структурой:

– Section – страница анкеты;
– Group – группы вопросов;
– Pool – пулы метрик;
– Metric – метрики.

Такая структура позволяет эффективно организовать вопросы и вычисляемые данные. Иерархия узлов анкеты представлена на рис. 2.

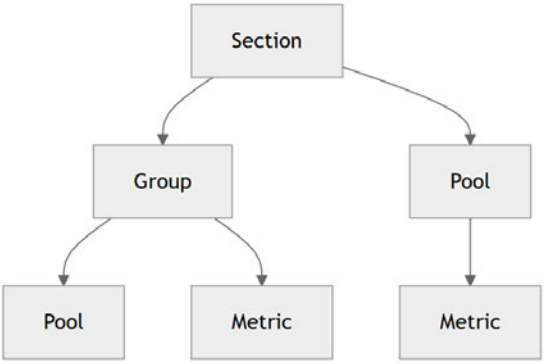


Рис. 2. Иерархия узлов анкеты

3. Реализация вычисляемых метрик

Особое внимание было уделено реализации вычисляемых метрик. В системе реализован абстрактный класс FormulaNode, от которого наследуются специфические типы формул, такие как:

– MetricFormulaNode – отражает значение метрики;

– ConstantFormulaNode – отражает фиксированное значение;

– OperationFormulaNode – реализует математические операции, такие как сложение, умножение, вычитание, деление;

– AggregateFormulaNode – реализует агрегирующие операции. Используется в шаблоне бенчмаркинга для подсчета медиан, перцентиль на основе ответов предприятий.

Формулы имеют рекурсивную структуру, обеспечивая высокую гибкость при вычислении значений. Диаграмма классов формул представлена на рис. 3.

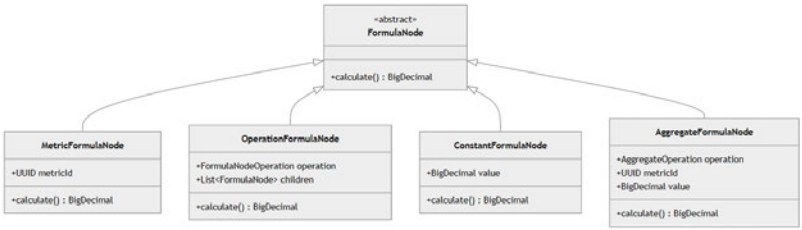


Рис. 3. Диаграмма классов для вычисляемых метрик

4. Описание взаимодействия клиентской части с сервером

Клиентская часть приложения взаимодействует с сервером посредством REST API, обеспечивая обмен данными в стандартном JSON-формате. Серверная часть, построенная на Spring Framework, отвечает за хранение анкет в реляционной базе данных, их обработку и выполнение сложных расчетов финансовых метрик с использованием библиотеки для работы с электронными таблицами.

Перед заполнением анкеты пользователь проходит процедуру аутентификации, после чего система загружает персонализированный список доступных исследований с учетом его прав доступа. Интерфейс выбора исследования представлен на рис. 4.

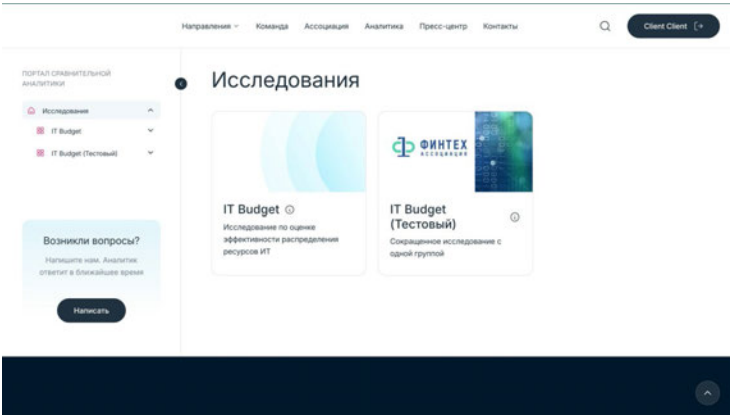


Рис. 4. Список исследований

После выбора исследования пользователю необходимо выбрать конкретную анкету из списка. Одно и то же предприятие может заполнять конкретное исследование несколько раз в квартал. Данная страница представлена на рис. 5.

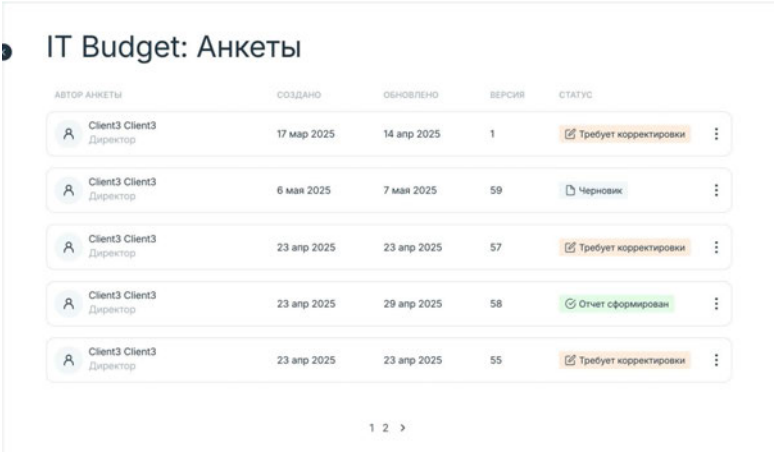


Рис. 5. Список анкет

После выбора анкеты пользователь попадает на страницу, которая представлена на рис. 6.

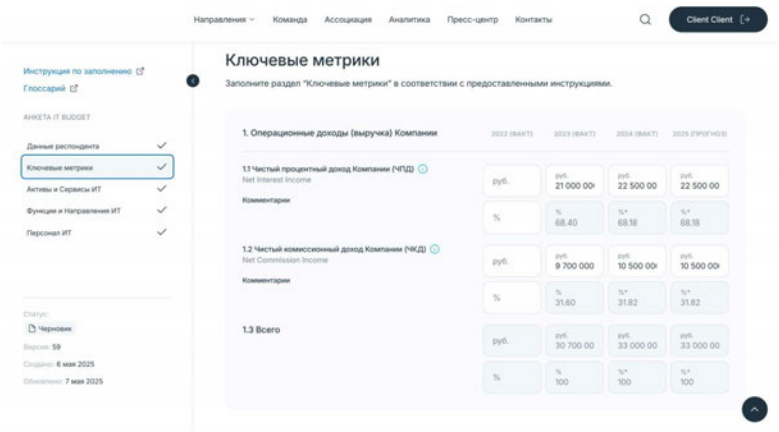


Рис. 6. Страница анкеты

Клиентская часть приложения получает из серверной части шаблон анкеты, которую отображает пользователю. Используя формулы, представленные в шаблоне, он может динамически рассчитывать метрики при вводе пользователем в реальном времени.

Заключение

Разработанное серверное приложение существенно сокращает время и трудоёмкость проведения рыночных исследований, повышает точность анализа и обеспечивает удобство взаимодействия пользователей с системой.

Автоматизация процессов сбора данных и генерации персонализированных отчётов позволяет компаниям оперативно получать актуальные аналитические данные, необходимые для принятия эффективных управленческих решений. Перспективами дальнейшего развития приложения могут стать интеграция с другими аналитическими платформами и расширение функционала для работы с большими данными.

Список литературы

1. Копытина, Е. А. Разработка веб-сервиса для формирования горячего коммерческого предложения ООО «ЧерноземАгромаш» / Е. А. Копытина, Ф. А. Десятириков, А. В. Копытин // Информатика: проблемы, методы, технологии : Материалы XXII Международной научно-практической конференции им. Э.К. Алгазинова, Воронеж, 10–12 февраля 2022 года / Под редакцией Д.Н. Борисова. – Воронеж:

Общество с ограниченной ответственностью "Вэлборн", 2022. – С. 990-996.

2. Свидетельство о государственной регистрации программы для ЭВМ № 2023661037 Российская Федерация. Приложение для автоматизации документооборота дополнительного образования : № 2023618813 : заявл. 03.05.2023 : опубл. 25.05.2023 / А. С. Голышева, Е. А. Копытина ; заявитель Федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный технический университет». – EDN NBUVMM.

3. Претков, А. И. Автоматизации формирования документов основных профессиональных образовательных программ высшего образования факультета компьютерных наук Воронежского государственного университета / А. И. Претков, Е. А. Копытина // Труды молодых учёных факультета компьютерных наук ВГУ : Сборник статей. – Воронеж : Воронежский государственный университет, 2023. – С. 94-107. – EDN HKXMIS.

4. Новохатский, Н. М. Разработка мобильного приложения для медицинских клиник "Медангел" / Н. М. Новохатский, А. В. Копытин, Е. А. Копытина // Сборник студенческих научных работ факультета компьютерных наук ВГУ : Сборник статей. В 2-х частях / Под редакцией Д.Н. Борисова. Том Выпуск 13. Часть 1. – Воронеж : Воронежский государственный университет, 2019. – С. 150-155.

5. Проектная деятельность в дополнительном образовании на примере практик российских ВУЗов / Е. А. Копытина, Л. А. Гладкова, М. Е. Семенов [и др.] // Информатика: проблемы, методы, технологии : Материалы XXII Международной научно-практической конференции им. Э.К. Алгазинова, Воронеж, 10–12 февраля 2022 года / Под редакцией Д.Н. Борисова. – Воронеж: Общество с ограниченной ответственностью "Вэлборн", 2022. – С. 1342-1355.

6. Подольский, К. Д. Формирование предварительного коммерческого предложения ООО «ЧерноземАгромаш» средствами MS Excel / К. Д. Подольский, Я. Э. Головин, Е. А. Копытина // Сборник студенческих научных работ факультета компьютерных наук ВГУ : Сборник научных работ. В 2-х частях / Под редакцией Д.Н. Борисова. Том Выпуск 14. Часть 1. – Воронеж : Воронежский государственный университет, 2020. – С. 143-147.

7. Официальная документация Java 21 SE [Электронный ресурс]. – Oracle. – Режим доступа: https://docs.oracle.com/en/java/javase/21– Заглавие с экрана. – (Дата обращения: 18.04.2025).

Разработка телеграм-бота для поиска данных о зарубежных авто

М. С. Долженко

Студент бакалавр

В. А. Ушаков

Ассистент

Введение

В современном мире, где технологии развиваются с невероятной скоростью, информационные ресурсы и автоматизированные системы становятся неотъемлемой частью повседневной жизни. Люди все чаще обращаются к цифровым решениям для упрощения рутинных задач, поиска информации и взаимодействия с различными сервисами. Одним из таких решений являются чат-боты, которые уже давно перестали быть просто инструментом для развлечения и превратились в мощный инструмент для бизнеса, образования и повседневного использования. Их популярность обусловлена простотой взаимодействия, доступностью и широкими возможностями интеграции с другими платформами. В этом контексте разработка телеграм-бота для поиска данных о зарубежных автомобилях представляется актуальной и востребованной задачей.

Автомобильная промышленность — это одна из самых динамично развивающихся отраслей, которая ежегодно предлагает потребителям новые модели, технологии и решения. Однако с увеличением количества доступных на рынке автомобилей растет и сложность выбора. Потенциальные покупатели сталкиваются с необходимостью анализа огромного объема информации: от технических характеристик и цен до отзывов владельцев и данных о надежности. Особенно это актуально для зарубежных автомобилей, которые могут отличаться специфическими особенностями, такими как адаптация к местным условиям, наличие сервисных центров и стоимость обслуживания. В таких условиях автоматизированный помощник, способный оперативно предоставлять актуальную информацию, становится незаменимым инструментом.

© Долженко М. С., Ушаков В. А., 2025

1. Интеграция со сторонним источником

Перед началом разработки стоит учесть один из основных принципов построения качественного ПО – масштабируемость. Разумеется, необходимо спроектировать такую архитектуру ИС и зависимость классов, которые позволят за малые ресурсы добавлять новые источники. С такой проблемой помогают справиться функциональные возможности языка Kotlin, а именно “интерфейсы”

Написанный интерфейс позволит во времени выполнения программы определять тип источника, а также производить поиск данных автомобиля по VIN-коду. Примечательно, что возвращаемый тип задается общий на все источники. Можно было сказать, что это ограничение контракта, но такая жёсткая типизация перевешивает все минусы и является целевым решением. Для создания новых поисковиков следует наследоваться от интерфейса Searcher.

Бесплатных и свободных источников с API крайне мало, либо они не имеют поддержки или SLA от производителя. Следовательно, стоит рассматривать интернет-портал с низким уровнем защиты, позволяющим проводить парсинг их контента [1], пример которого представлен на рис. 1.

```
<div class="maincontent" id="maincontent">



    <div class="car">
        <div class="car-schema" itemtype="http://schema.org/Car" itemscope>
            <meta itemprop="bodyType" content="4DR SPOR" />

            <meta itemprop="fuelType" content="DIESEL" />

            <meta itemprop="knownVehicleDamages" content="SIDE" />

            <meta itemprop="mileageFromOdometer" content="102368" />

            <meta itemprop="modelDate" content="2014" />

            <meta itemprop="vehicleIdentificationNumber" content="1C4RJFCM3EC427300" />

            <meta itemprop="vehicleTransmission" content="AUTOMATIC" />
        </div>
```

Рис. 1. Пример ответа источника данных Auction History

2. Механизм наполнения историчными данными БД

Основной функциональной возможностью приложения является формирование историчного отчета об автомобиле, но для его создания необходимо наполнить БД самими данными.

Наполнение БД проходит несколько этапов, финальный код которых представлен в листинге 1.

Первым этапом является наполнение очереди: после каждого вызова источника данных формируется запись в таблицу “history_queue_entity”. Все поля, которые удалось обработать, помещаются в ассоциативный массив, переводятся в JSON и сохраняются в таблице.

Листинг 1

Наполнение БД историчными данными

```
@Scheduled(cron = "\${queue-puller.cron}")
@Transactional
fun fillHistoryByNewValues() {
    val newValues = queueRepository.select1RandomEntity() ?:
return val typeRef = object : TypeReference<HashMap>() {}
    val historyValues = runCatching {
    objectMapper.readValue(newValues.data,
typeRef)}.getOrNull() ?: return
historyService.tryToFillHistoryByNewValues(
    newValues.vin,
    historyValues,
    newValues.sourceType
    )
    queueRepository.delete(newValues)
}
```

Для периодического чтения очереди необходимо включить в Spring запланированные задачи. Эта возможность станет доступной после указания аннотации над конфигурационным Bean.

Вторым этапом является чтение и обработка. Чтение записей из очереди сообщений выполняет компонент HistoryQueueEntityPuller, который запускается каждую минуту. Сохранение данных в таблицу “history_entity” происходит в несколько этапов:

- Выбирается 1 случайная запись из очереди и блокируется внутренним инструментарием СУБД (SELECT FOR UPDATE)[2];
- из выбранной записи берётся vin и ищутся в таблице “history_entity” последние данные по автомобилю;
- если данные, полученные из историчной таблицы, сохранены позже, чем месяц назад, то новые данные из источника сохраняются;
- запись из таблицы очереди удаляется.

Разумеется, стоит предусмотреть случай, если при обработке данных произойдет ошибка. В этом случае, следует воспользоваться транзакциями и их свойствами, доступными в СУБД.

3. Формирование мгновенного отчёта

Ранее было описано, что каждый из источников унаследован от интерфейса Searcher. Это позволяет работать со всеми источниками, как со списком одного типа без явного приведения, что сокращает программный код.

К счастью, Spring умеет работать с таким списком и позволяет инжектировать в этот Bean необходимые зависимости.

Так, мы можем единовременно запустить корутину[3] для каждого из элемента списка (листинг 2).

Листинг 2

Запуск корутин в сервисе формирования мгновенного отчёта

```
fun searchInfo(vin: String): Map<SourceType, CarInfoDto?> {
    return runBlocking {
        searchers.map { searcher ->
            async {
                searcher.getType() to searcher.findByVin(vin)
            }
        }.awaitAll()
    }.toMap()
}
```

При вызове механизма формирования мгновенного отчета система выводит список данных по автомобилю. Однако, этот набор не будет читабелен для пользователя, поэтому необходимо обработать список с помощью преобразователя.

Создадим новый преобразователь – CarInfoDtoPrettierService, код которого представлен в листинге 3. Данный класс преобразовывает исходную DTO[4] в читаемый набор строк для пользователя

Листинг 3

Преобразование ответа источника в сообщение Telegram

```
override fun prettyInfos(infos: Map<SourceType, CarInfoDto?>)
= infos
    .filter { it.value != null }
    .map { info ->
        "${info.key.title}:\n${prettyInfo(info.value!!)}"
    }

private fun prettyInfo(info: CarInfoDto) =
    info.data
        .filter { it.value != null }
        .toList()
        .joinToString(separator = "\n") { "${it.first} -
${it.second!!}" }
```

Обработанные сообщения отправляются с помощью Telegram API отдельными сообщениями, разбитыми по источникам. Пример такого ответа представлен на рис. 2

Рис. 2. Примера мгновенного отчёта по автомобилю

4. Формирование исторического отчёта об автомобиле

Для получения историчного отчета об автомобиле воспользуемся ранее созданным и описанным выше механизмом формирования историчного отчета и создадим новую команду согласно паттерну “Команда”.

Целевым классом является HistoryCommand, который принимает запрос пользователя и разбивает исходный текст вызова на составляющие. Важно выделить из результата VIN-код автомобиля.

Далее, вызвав механизм формирования историчного отчета, будет получен список данных по автомобилю. Однако, этот набор так же, как и в прошлом разделе, не будет читабелен для пользователя, поэтому необходимо обработать список с помощью нового преобразователя.

Создадим новый преобразователь – VehicleHistoryPrettierService. Данный класс преобразовывает исходную DTO в читаемый набор строк для пользователя.

Однако стоит учесть, что прошлый преобразователь выдавал мгновенные данные, которых как известно фиксированное количество,

но теперь к общему числу прибавляются ещё все историчные данные из БД. Таким образом, здесь раскрывается смысл разбиения отчета на несколько сообщений для пользователя, так как Telegram имеет ограничение на количество символов в одном сообщении[5]. Пример сообщения представлен на рис. 3.

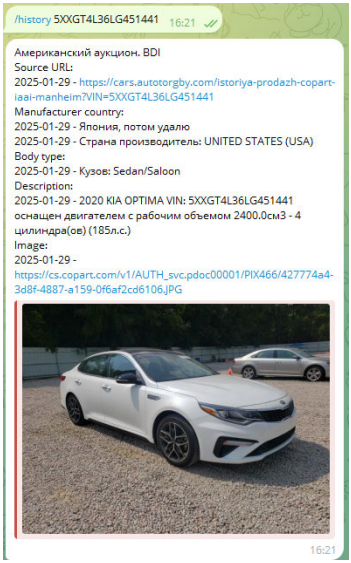


Рис. 3. Пример историчного отчёта по автомобилю

Заключение

Данная статья посвящена реализации телеграм бота для поиска данных о зарубежных авто. Описаны алгоритмы парсинга данных из открытых источников и формирование историчного отчёта. В результате данной разработки получился полностью рабочий телеграм бот, реализованный в виде приложения.

Список литературы

1. Jsoup HTML Parsing [Электронный ресурс] : Jsoup. – Режим доступа : http://www.jsoup.org

2. PostgreSQL. Explicit Locking [Электронный ресурс] : PostgreSQL Docs. – Режим доступа : https://www.postgresql.org/docs/current/explicit-locking.html

3. Жемеров, Д. Kotlin в действии / Д. Жемеров, С. Исакова. – 3-е изд. – М.: ДМК Пресс, 2018. – 402 с.

4. Ограничения в телеграм-бот (Bot API), о которых никто не расскажет [Электронный ресурс] : Habr. – Режим доступа : https://habr.com/ru/companies/tensor/articles/799565/
5. The DTO Pattern (Data Transfer Object) [Электронный ресурс] : Baeldung. – Режим доступа : https://www.baeldung.com/java-dto-pattern

Анализ поведенческих характеристик пользователя для обеспечения защиты информации в информационной системе

А. С. Дьякова

Студент бакалавр

А. С. Мальцев

Доцент

Введение

В современном цифровом мире информационная безопасность становится одной из приоритетных задач для организаций и государственных структур. Одним из ключевых факторов, влияющих на уровень защищенности информационных систем, является человеческий фактор. Анализ поведенческих характеристик пользователей позволяет выявлять аномалии, связанные с внутренними и внешними угрозами, такими как инсайдерские атаки, компрометация учетных записей и несанкционированный доступ к данным.

Целью данной работы является анализ алгоритмов машинного обучения в задаче обнаружения аномальной активности пользователей информационных систем.

На начальном этапе проводился обзор предметной области, связанной с анализом поведенческих характеристик пользователей, с акцентом на идентификацию факторов, влияющих на формирование аномального поведения в информационных системах.

Вторая задача заключалась в рассмотрении существующих методов анализа пользовательского поведения, включая как эвристические, так и формализованные подходы, применяемые в практике информационной безопасности.

Третья задача состояла в проведении анализа алгоритмов машинного обучения, направленного на оценку их применимости к задачам обнаружения аномального поведения пользователей, с учётом особенностей поведенческих данных, характеристик различных типов алгоритмов, а также факторов, влияющих на точность и устойчивость принимаемых решений в условиях реальных информационных систем.

© Дьякова А. С., Мальцев А. С., 2025

1. Поведенческий анализ и человеческий фактор как источник угроз информационной безопасности

Поведенческий анализ (Behavioral Analysis) – это метод анализа действий и поведения пользователей или объектов в компьютерной системе с целью выявления потенциальных угроз безопасности.

Одним из ключевых факторов, оказывающих влияние на устойчивость информационных систем к различным видам нарушений, является человеческий фактор. Человеческий фактор в информационной безопасности – это совокупность психологических, поведенческих и социальных характеристик людей, которые могут влиять на безопасность информационных систем и данных [1]. Человеческий фактор, в данном контексте, может выступать как непосредственным источником угроз, так и посредником, способствующим их реализации. Применение методов поведенческого анализа позволяет существенно снизить риски, связанные с деятельностью пользователей, и тем самым повысить общий уровень информационной безопасности в организации.

Можно выделить ряд ключевых угроз информационной безопасности, непосредственно связанных с человеческим фактором. Одной из них является социальная инженерия — совокупность методов психологического воздействия, применяемых злоумышленниками для получения конфиденциальной информации или побуждения пользователей к действиям, угрожающим безопасности системы.

Вторую группу составляют внутренние угрозы (инсайдерские атаки), исходящие от лиц, обладающих легитимным доступом к информационным ресурсам организации. Такие угрозы отличаются высокой скрытностью и потенциально серьёзными последствиями, включая утечку данных и финансовый ущерб.

Отдельную категорию составляют атаки на цепочки поставок, направленные на компрометацию программного обеспечения через легитимные каналы его распространения.

В данном исследовании рассматриваются информационные системы, в которых обрабатывается такая информация ограниченного доступа, как персональные данные и коммерческая тайна. Для данных информационных систем наиболее распространены следующие угрозы:

- угроза восстановления аутентификационной информации;
- угроза подбора аутентификационной информации;
- угроза несанкционированного использования привилегированных учетных записей;
- угроза несанкционированного доступа к конфиденциальной информации;

– угроза несанкционированного создания или удаления учетных записей;
– угроза доступа к защищаемым файлам с использованием обходного пути;
– угроза повышения привилегий.

2. Классификация методов и подходов к анализу поведения пользователей

Современные подходы к анализу активности пользователей основаны на различных методах, включая правила на основе политик, машинное обучение и поведенческую аналитику, а также контекстуальный анализ и корреляцию событий.

Правило-ориентированный подход (Rule-Based Approach) является одним из наиболее традиционных методов мониторинга поведения пользователей в системах информационной безопасности. Его основная концепция заключается в использовании заранее заданных правил и политик, определяющих допустимые и недопустимые действия пользователей [2].

Аналитика поведения пользователей и сущностей (User and Entity Behavior Analytics, UEBA) — современный метод кибербезопасности, основанный на анализе активности пользователей и IT-активов для выявления аномалий. В отличие от традиционных методов, основанных на статических правилах и сигнатурах, UEBA применяет поведенческую аналитику и машинное обучение, что позволяет обнаруживать внутренние и внешние угрозы, даже если их действия не соответствуют известным шаблонам атак [3].

Контекстуальный анализ и корреляция событий представляют собой важнейшие методы, применяемые в информационной безопасности для выявления сложных киберугроз, которые остаются незамеченными при использовании традиционных сигнатурных или правило-ориентированных систем. Основная идея данных методов заключается в интеграции информации из различных источников, учёте контекста происходящих событий и выявлении отклонений, которые могут свидетельствовать о кибератаках, инсайдерских угрозах или иных инцидентах безопасности.

3. Описание входных данных

Для решения поставленных задач по анализу поведенческих характеристик пользователей был выбран набор данных для обнаружения кибервторжений. Данный датасет предназначен для выявления атак на основе анализа сетевого трафика и поведения пользователей в информационной системе. Комплексность и

многоуровневая структура данных позволяют использовать его для построения моделей машинного обучения и проведения поведенческого анализа.

Набор данных включает признаки, характеризующие сетевую активность, а также особенности поведения пользователей и сущностей. В табл. 1 приведено краткое описание основных признаков.

Таблица 1

Описание признаков входного набора данных

Название признака	**Описание**
network_packet_size	Размер сетевого пакета в байтах (от 64 до 1500 байт)
protocol_type	Тип используемого сетевого протокола (TCP, UDP, ICMP)
encryption_used	Применяемый протокол шифрования (AES, DES, None)
login_attempts	Количество попыток входа в систему
session_duration	Продолжительность пользовательской сессии в секундах
failed_logins	Число неудачных попыток входа
unusual_time_access	Флаг входа в систему в необычное время
ip_reputation_score	Оценка репутации IP-адреса
browser_type	Используемый пользователем тип браузера (Chrome, Firefox, Edge, Safari)

Целевая переменная (attack_detected) представляет собой бинарную классификацию событий: значение "1" соответствует обнаруженной атаке, а значение "0" — нормальной активности. Таким образом, структура данного набора данных позволяет не только фиксировать отдельные факты подозрительного поведения, но и анализировать их в контексте для последующего построения интеллектуальных моделей обнаружения инцидентов информационной безопасности.

После рассмотрения структуры и содержания признаков, представленных во входном наборе данных, необходимо было провести предварительный анализ взаимосвязей между ними. Одним из ключевых инструментов такого анализа является построение матрицы корреляций. Каждая ячейка матрицы содержит коэффициент корреляции

Пирсона, принимающий значения от –1 до 1. Значение, близкое к 1, указывает на сильную положительную связь между признаками, значение, близкое к –1, свидетельствует о сильной отрицательной связи, а значение около 0 указывает на отсутствие линейной зависимости. Матрица корреляции признаков для исходного набора данных представлена на рис. 1.

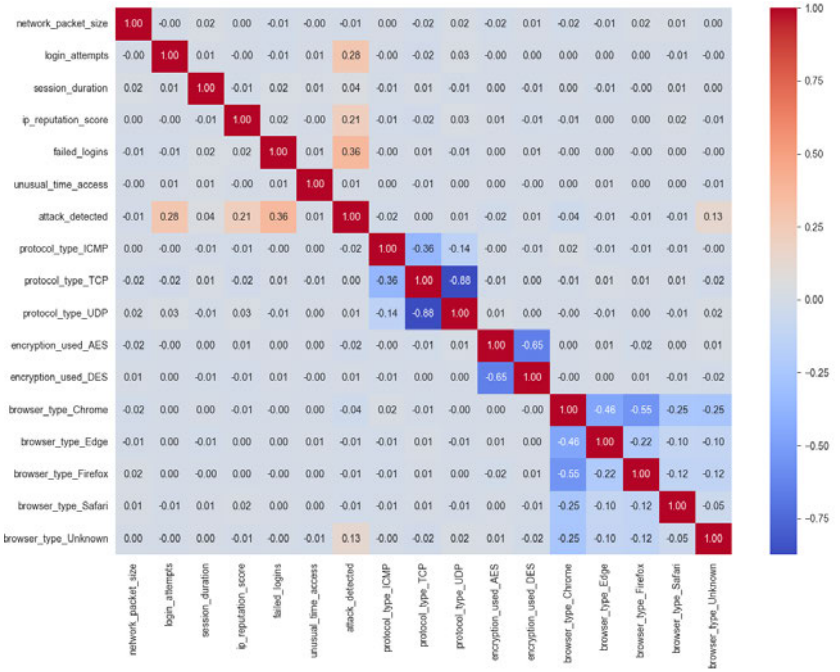


Рис. 1. Матрица корреляции признаков

Анализ матрицы корреляций показал, что количество неудачных попыток входа положительно коррелирует с целевой переменной (коэффициент 0,36), что свидетельствует о важности данного признака для выявления атак, связанных с подбором учетных данных. Также установлена умеренная положительная связь между количеством попыток входа и целевой переменной. Репутационный балл IP-адреса демонстрирует корреляцию на уровне 0,21, что позволяет рассматривать его как дополнительный индикатор при комплексной оценке аномальной активности.

4. Анализ алгоритмов машинного обучения

После анализа корреляций признаков следующим этапом стало построение моделей машинного обучения для обнаружения аномальной активности пользователей. Для этого использовались как классические алгоритмы (логистическая регрессия, SVM, KNN [4]), так и современные ансамблевые методы — Random Forest [5], XGBoost и CatBoost.

Сравнительный анализ моделей проводился с использованием следующих метрик качества: точность классификации (Accuracy), полнота (Recall), точность (Precision), F1-мера (F1-score) и площадь под ROC-кривой (ROC-AUC). Результаты сравнительного тестирования моделей приведены в табл. 2.

Таблица 2

Результаты сравнения моделей машинного обучения

Модель	**Accuracy**	**Recall**	**Precision**	**F1-score**	**ROC-AUC**
CatBoost	0.8936	0.8839	0.9135	0.8897	0.8917
Random Forest	0.8931	0.8822	0.9181	0.8887	0.8810
XGBoost	0.8926	0.8818	0.9168	0.8882	0.8885
SVM (RBF)	0.8181	0.8114	0.8222	0.8140	0.8666
KNN	0.7778	0.7685	0.7842	0.7710	0.8233
SVM (Linear)	0.7448	0.7412	0.7427	0.7418	0.8101
Logistic Regression	0.7442	0.7428	0.7421	0.7424	0.8102

Анализ полученных результатов показал, что наивысшие показатели были достигнуты ансамблевыми моделями, в частности CatBoost, Random Forest и XGBoost. Модель CatBoost продемонстрировала наилучший баланс между всеми оценочными метриками, обеспечив высокую точность классификации при уверенной способности различать классы.

Оставшиеся модели, включая SVM, KNN и логистическую регрессию, показали более низкие значения всех метрик. Несмотря на преимущества, такие как простота интерпретации и относительно низкие вычислительные затраты, они уступают ансамблевым методам по качеству предсказаний.

Модель CatBoost основана на градиентном бустинге над решающими деревьями с оптимизациями для категориальных признаков и порядковой обработки данных. Ключевым элементом метода является минимизация функции потерь следующего вида:

$$L = \sum_{i=1}^{n} l(y_i, F_{m-1}(x_i) + \gamma_m h_m(x_i)), \quad (1)$$

где L – функция потерь, l – локальная функция потерь для одного наблюдения, y_i – истинная метка, $F_{m-1}(x_i)$ – предсказание ансамбля на предыдущей итерации, $h_m(x_i)$ – новое дерево, добавляемое на m-ой итерации, γ_i – оптимальный коэффициент бустинга.

Для более глубокой оценки качества классификации модели CatBoost использовалась матрица ошибок, характеризующая типичные ошибки алгоритма. Матрица ошибок для CatBoost представлена на рис.2.

Рис. 2. Матрица ошибок для CatBoost

Анализ матрицы ошибок модели CatBoost выявил низкое число ложноположительных срабатываний, что отражает высокую точность в распознавании нормальной активности. Основная часть ошибок связана с ложноотрицательными предсказаниями, что типично для задач выявления аномалий. В целом, модель демонстрирует удовлетворительный баланс между точностью и полнотой, эффективно обнаруживая большинство аномальных сценариев.

Заключение

В данной статье был проведён комплексный анализ алгоритмов машинного обучения для задачи обнаружения аномальной активности пользователей в информационных системах. Особое внимание уделено изучению влияния человеческого фактора на информационную безопасность, а также классификации угроз, связанных с поведением пользователей. Рассмотрены современные подходы к поведенческому анализу, включая использование методов машинного обучения.

На практическом этапе исследования был проведён анализ исходных данных, выявлены наиболее значимые признаки, построены и сравнены модели машинного обучения. Модель CatBoost показала наилучший баланс между точностью и полнотой классификации, что подтвердилось результатами анализа матрицы ошибок. Полученные результаты продемонстрировали перспективность использования поведенческого анализа и методов машинного обучения для повышения эффективности систем обнаружения аномальной активности.

Список литературы

1. Лебедев, М. А. Роль человеческого фактора в информационной безопасности / М. А. Лебедев, Д. А. Демкин // Международный научный журнал «Вестник науки». – 2024. – № 8. – С. 141-144.

2. Understanding role and rule-based, mandatory and discretionary access control models [Электронный ресурс] : Avigilon. – Режим доступа : https://www.avigilon.com/blog/access-control-models

3. UEBA: как анализ поведения помогает защищать данные [Электронный ресурс] : Хабр. – Режим доступа : https://habr.com/ru/articles/878142/

4. Моисеева, Т. А. Сравнительный анализ процедур кластеризации для обнаружения аномалий показателей, характеризующих функционирование контактной сети железных дорог / Т. А. Моисеева, Т. М. Леденева // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2020. – № 2. – С. 115-126.

5. Павлычев, А. В. Использование алгоритма машинного обучения Random Forest для выявления сложных компьютерных инцидентов / А. В. Павлычев, М. И. Стародубов, А. Д. Галимов // Вопросы кибербезопасности. – 2022. – № 5(51). – С. 74-81.

Реализация Exactly-Once Semantics в Kafka

А. С. Елисеев

Студент бакалавр

Н. К. Самойлов

Старший преподаватель

Введение

Надёжная доставка сообщений остаётся одной из ключевых задач для распределённых систем — особенно там, где ошибки в обработке событий могут привести к серьёзным последствиям: в платёжных сервисах, IoT-платформах, аналитических решениях реального времени. Apache Kafka, разработанная в LinkedIn и с 2012 года входящая в экосистему Apache, получила широкое распространение благодаря масштабируемости, отказоустойчивости и высокой пропускной способности.

Тем не менее, даже столь зрелая архитектура не застрахована от проблемы дублирования сообщений. Повторная доставка или потеря событий способны исказить бизнес-данные, увеличить расходы и нарушить консистентность системы. Классические подходы — at-least-once (минимум одна доставка) и at-most-once (без повторов, но с возможными потерями) — далеко не всегда отвечают требованиям, которые предъявляют современные приложения. Там, где невозможно мириться ни с дублированием, ни с потерями, актуальной становится семантика exactly-once (EOS), позволяющая добиться строгой однократной и точной доставки каждого сообщения.

1. Основы семантики доставки сообщений

Семантика доставки сообщений в распределённых системах определяет правила, по которым сообщения передаются от производителя к потребителю через брокер, например, такой как Apache Kafka. От того, какая модель доставки выбрана, зависит баланс между скоростью, надёжностью и целостностью данных [1].

В системах, использующих семантику at most once, предпочтение отдаётся скорости и минимальным накладным расходам. Производитель отправляет сообщения без ожидания подтверждения — например, с параметром acks, установленным в 0. Такой подход позволяет

© Елисеев А. С., Самойлов Н. К., 2025

максимально быстро обрабатывать сообщения, но несёт риск их потери при сбоях на стороне производителя, брокера или потребителя, ведь дублирование полностью исключено за счёт возможных потерь.

Семантика at least once применяется там, где важно не упустить ни одного сообщения, даже если в результате возможны дубли. В этом случае производитель ожидает подтверждения при отправке — параметр acks может быть равен 1 или all. При отсутствии подтверждения сообщение отправляется повторно. На стороне потребителя автоматический коммит смещений обычно отключается, и фиксация смещения происходит вручную только после успешной обработки сообщения. Такой механизм гарантирует, что каждое сообщение будет обработано как минимум один раз, но в случае сбоя между обработкой и коммитом могут возникнуть дубликаты.

Exactly-once — самая строгая модель доставки, гарантирующая, что каждое сообщение будет доставлено и обработано ровно один раз, без потерь и дублирования. Для реализации этой семантики в Kafka используются два ключевых механизма: идемпотентный производитель и транзакции. Идемпотентность обеспечивается за счёт уникального идентификатора производителя и последовательного номера для каждого сообщения, что позволяет брокеру игнорировать повторные отправки одного и того же сообщения. Транзакции, в свою очередь, гарантируют, что все сообщения внутри одной транзакционной сессии либо будут записаны и обработаны целиком, либо не будут записаны вовсе. Такой подход позволяет строить надёжные цепочки обработки данных и исключать риск потери или дублирования информации даже в случае сбоев [2].

2. Идемпотентность в Kafka

В Apache Kafka идемпотентность означает, что повторная отправка одного и того же сообщения производителем приводит к одному и тому же результату — сообщение будет записано в топик ровно один раз. Это свойство важно для реализации семантики exactly-once, при которой каждое сообщение доставляется без дублирования.

Для включения идемпотентности достаточно установить параметр enable.idempotence в значение true. После этого рекомендуется скорректировать настройки max.in.flight.requests.per.connection, retries и acks. Количество одновременных запросов на одно соединение (max.in.flight.requests.per.connection) не должно превышать 5, чтобы сохранить порядок доставки сообщений и избежать риска дублирования. Параметр retries должен быть больше нуля, чтобы гарантировать успешную доставку сообщений даже при временных ошибках. Параметр acks рекомендуется выставлять в значение all, что

обеспечивает подтверждение доставки от всех реплик брокера и повышает надёжность работы [3].

Эти настройки вместе позволяют добиться гарантии, что каждое сообщение будет записано ровно один раз, независимо от количества попыток отправки. На рис. 1 показан типичный сценарий работы идемпотентного продюсера в Kafka: если по каким-либо причинам подтверждение о приёме сообщения (ack) не доходит до продюсера, и он повторно отправляет то же сообщение, Kafka определяет дубликат и не фиксирует его во второй раз.

На рис. 1 представлена схема, демонстрирующая, как Kafka предотвращает дублирование сообщений при повторных отправках.

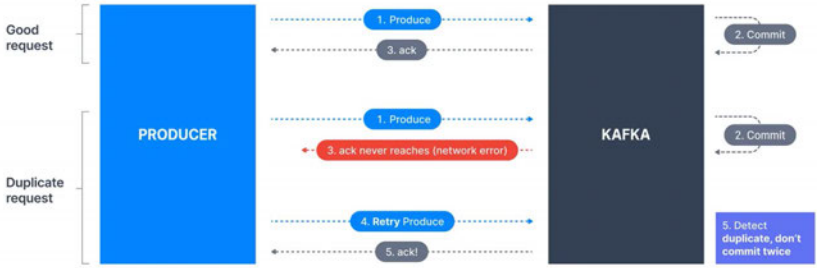


Рис. 1. Идемпотентный продюсер

Применение идемпотентного производителя особенно важно в системах, где дублирование сообщений недопустимо, например, в финансовых сервисах. Дополнительные накладные расходы минимальны: требуется хранить несколько числовых полей для отслеживания идентификаторов и последовательных номеров, что практически не сказывается на производительности по сравнению с обычным производителем.

Идемпотентный режим также может быть полезен в гибридных сценариях с семантикой at least once, если повторные записи происходят за счёт внутренних механизмов Kafka, а не внешних приложений.

3. Транзакции в Kafka

Транзакции в Apache Kafka предоставляют возможность атомарной записи сообщений сразу в несколько топиков и партиций, следуя принципу «всё или ничего». Это означает, что все сообщения внутри одной транзакции будут либо полностью записаны и станут доступны для потребителей, либо, в случае сбоя, не попадут в хранилище вовсе. Такой подход позволяет избежать частичной обработки данных и

гарантирует целостность системы при обработке сложных бизнес-операций [4].

Основу транзакционного механизма в Kafka составляют несколько ключевых компонентов. Каждому производителю, который работает с транзакциями, присваивается уникальный идентификатор (transactional.id), необходимый для сохранения транзакционного состояния даже при перезапусках приложения. Управление транзакциями осуществляет специальный компонент брокера — координатор транзакций. Он отвечает за обработку всех действий, связанных с данным transactional.id, отслеживает состояние текущей транзакции и обеспечивает её целостность. Информация о каждой транзакции хранится во внутреннем топике Kafka под названием __transaction_state, где фиксируются такие состояния, как «транзакция в процессе», «подготовка к коммиту» и «транзакция завершена». Эпоха (epoch) — это дополнительная метадата, которая указывает на количество подключений, осуществлённых с данным идентификатором транзакции. В системе также предусмотрено понятие зомби-экземпляра — это ситуация, когда предыдущий процесс производителя начал запись сообщений, но не смог корректно завершить транзакцию из-за сбоя.

Работа с транзакциями в Kafka начинается с того, что производитель отправляет запрос брокеру для определения координатора своей транзакции. После этого он регистрируется у координатора, указывая свой transactional.id, и получает уникальный идентификатор производителя вместе с эпохой. При этом удаляются все незавершённые транзакции, которые могли остаться от предыдущих экземпляров с тем же идентификатором. Запуск транзакции осуществляется с помощью метода beginTransaction, после чего все сообщения, отправленные производителем, автоматически становятся частью транзакции. В журнале транзакций при этом фиксируется, что операция находится в процессе выполнения. Производитель может добавлять новые партиции для записи, отправлять сообщения в пользовательские топики, а также фиксировать смещения прочитанных данных, чтобы обеспечить согласованность обработки. Для фиксации смещений используется внутренний топик __consumer_offsets, в который производитель записывает необходимые данные в рамках той же транзакции.

Завершение транзакции осуществляется вызовом метода commitTransaction. В этот момент все изменения фиксируются как в журнале транзакций, так и в пользовательских топиках. Сообщения и смещения, относящиеся к успешной транзакции, становятся видимыми для потребителей. Если транзакция прерывается по каким-либо

причинам, все связанные с ней сообщения и смещения остаются недоступными для чтения [5].

Доступность сообщений для потребителей зависит от значения параметра isolation.level. Если выбран режим read_committed, потребитель получает только подтверждённые сообщения, то есть данные, по которым транзакция завершилась успешно. В режиме read_uncommitted становятся видимыми все сообщения, даже если транзакция была прервана или не закончена. Благодаря такому механизму семантика exactly-once достигается просто: потребитель может читать сообщения из одного топика, записывать их в другой в рамках той же транзакции и одновременно фиксировать смещения. В случае прерывания транзакции все промежуточные результаты автоматически становятся невидимыми, а система возвращается к последнему стабильному состоянию, что исключает появление дубликатов или потерю данных.

На рис. 2 представлена схема работы транзакций в Kafka. На этом рисунке отражён весь жизненный цикл транзакции: от запроса координатора, инициализации транзакционного идентификатора и запуска транзакции до записи сообщений, фиксации смещений и завершения транзакции с помощью commit. Также показано, как изменения в статусе транзакции отражаются во внутреннем топике __transaction_state, а смещения потребителей записываются в __consumer_offsets и становятся доступны для чтения только после коммита.

4. Практическое применение EOS

Семантика Exactly-Once (EOS) в Apache Kafka гарантирует, что каждое сообщение будет доставлено и обработано ровно один раз, даже при возникновении сбоев. Эта возможность становится особенно значимой в тех областях, где потеря или дублирование событий способно привести к серьёзным последствиям — от искажения результатов анализа до прямых финансовых потерь.

В финансовых системах, например, при обработке банковских переводов или операций на торговых платформах, даже одна ошибочная запись может стать причиной двойного списания средств, неверных отчётов или нарушений требований законодательства. EOS в Kafka позволяет полностью исключить такие ошибки, обеспечивая однократную фиксацию каждой транзакции и тем самым защищая бизнес и клиентов от последствий некорректной обработки данных.

В реальном времени критически важна точность анализа данных — будь то мониторинг производственных процессов, обработка телеметрии или системы управления безопасностью. Дублирование

сообщений способно существенно исказить выводы, а потеря событий — сделать картину неполной. Благодаря EOS, каждая единица информации оказывается учтённой строго один раз, что позволяет системам строить корректные и надёжные аналитические модели.

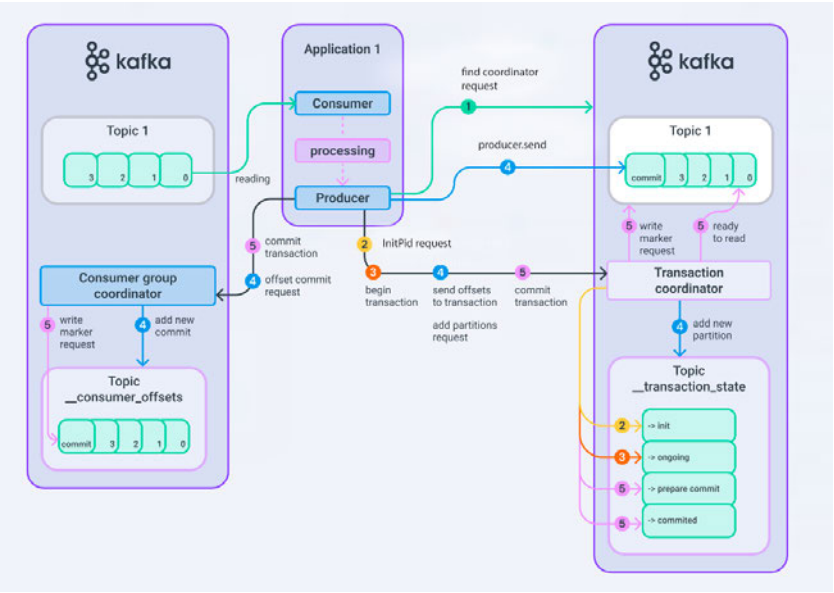


Рис. 2. Транзакции в Kafka

IoT-платформы обрабатывают огромные объёмы событий от датчиков и устройств, отслеживая параметры среды, состояние оборудования или безопасность объектов. В таких сценариях потеря сообщения может привести к пропуску критически важного сигнала, а дублирование — к ложной тревоге или неправильному решению. Поддержка EOS обеспечивает, что данные от каждого устройства надёжно зафиксированы и не повторяются, сохраняя точность и достоверность всех последующих действий системы.

В логистике и управлении цепочками поставок ошибки учёта, возникшие из-за дублирования или потерь сообщений, приводят к задержкам, неточным запасам и финансовым издержкам. Благодаря реализации EOS в Kafka становится возможным фиксировать каждое событие движения товара или изменение статуса заказа единожды, что способствует прозрачности процессов и минимизации рисков.

Особое значение точность доставки сообщений приобретает в медицинских информационных системах. Здесь некорректная обработка данных может повлиять на постановку диагноза, ведение электронной истории болезни или даже на безопасность пациента. EOS гарантирует однократную и точную доставку медицинских данных, исключая риски, связанные с дублированием или потерей информации.

Заключение

В этой статье были подробно рассмотрены механизмы реализации семантики Exactly-Once (EOS) в Apache Kafka, а также их практическая значимость для различных сфер применения. Архитектура Kafka сочетает масштабируемость, отказоустойчивость и высокую производительность, а поддержка EOS добавляет принципиально новый уровень надёжности, позволяя гарантировать однократную доставку каждого сообщения даже в условиях сбоев и повторных попыток отправки.

Ключевыми инструментами для достижения EOS выступают идемпотентность и транзакции. Идемпотентный продюсер обеспечивает, что сообщение попадёт в топик только один раз, независимо от количества попыток отправки, а механизм транзакций позволяет атомарно обрабатывать группы сообщений в нескольких партициях. Такой подход делает Kafka эффективным решением для задач, где даже минимальное искажение данных недопустимо.

Практика показывает, что EOS востребована в финансовых сервисах, реальном времени, IoT, логистике и медицинских системах — везде, где точность и целостность данных критичны для безопасности, правильности решений и доверия пользователей. Использование EOS в Apache Kafka позволяет строить надёжные и масштабируемые платформы, отвечающие самым высоким требованиям к обработке информации.

Список литературы

1. Exactly-once в Kafka: как это работает [Электронный ресурс] : статья о реализации семантики "ровно один раз" в Kafka. – Режим доступа: https://bigdataschool.ru/blog/kafka-exactly-once.html

2. Kafka. Гарантированная доставка сообщений ровно один раз [Электронный ресурс] : статья о реализации exactly-once в Kafka от компании Badoo. – Режим доступа: https://habr.com/ru/companies/badoo/articles/333046/

3. Idempotent Kafka Producer [Электронный ресурс] : статья об идемпотентности продюсера в Kafka. – Режим доступа: https://learn.conduktor.io/kafka/idempotent-kafka-producer/

4. Гарантии доставки сообщений в Kafka [Электронный ресурс] : документация по архитектуре и гарантиям доставки от Arenadata. – Режим доступа: https://docs.arenadata.io/ru/ADStreaming/current/concept/architecture/kafka/delivery_guarantees.html?ysclid=m7xre9jj6p193172473

5. Transactional publication to Kafka [Электронный ресурс] : статья о транзакциях и публикации сообщений в Kafka. – Режим доступа: https://bigdataschool.ru/blog/news/kafka/transactional-publication-to-kafka.html

Оценка целесообразности автоматизации функционального тестирования программного обеспечения на примере веб-приложения

И. С. Ермаков

Студент бакалавр

И. В. Илларионов

Доцент

Введение

В условиях стремительного развития информационных технологий и усложнения программных продуктов обеспечение их высокого качества становится одной из ключевых задач для современных IT-компаний. Тестирование программного обеспечения является неотъемлемой частью процесса разработки, направленной на выявление дефектов и повышение надежности конечного продукта.

Однако, с ростом масштабов проектов и увеличением скорости выпуска обновлений традиционные подходы к ручному тестированию сталкиваются с серьезными ограничениями в виде значительных временных и трудовых затрат, человеческого фактора и невозможности обеспечить полноценное регрессионное тестирование в сжатые сроки.

Автоматизация тестирования представляется перспективным решением данных проблем, позволяющим повысить эффективность процесса обеспечения качества, ускорить выявление дефектов и сократить затраты.

Особенно актуальным это становится для веб-приложений, характеризующихся высокой динамичностью, постоянным обновлением функционала и необходимостью поддержки различных браузеров и устройств.

1. Постановка задачи

В рамках настоящего исследования поставлена цель – оценить целесообразность внедрения автоматизированного функционального тестирования веб-приложений.

Для её достижения необходимо:

– Исследовать теоретические аспекты ручного и автоматизированного тестирования;

© Ермаков И. С., Илларионов И. В., 2025

– Разработать автоматизированные функциональные тесты для выбранного приложения;
– Провести сравнительный анализ экономических затрат на ручное и автоматизированное тестирование.

2. Проектирование тестовых сценариев

Были спроектированы два тестовых сценария для веб-приложения SauceDemo: TC-001 ("Полный цикл покупки") и TC-002 ("Проверка сортировки товаров"). Проектирование включает описание шагов, входных данных и ожидаемых результатов для каждого сценария.

TC-001: Полный цикл покупки

Цель: проверить корректность полного цикла покупки на сайте https://www.saucedemo.com/ – от авторизации до оформления заказа.

Таблица 1

Тестовый сценарий TC-001 «Полный цикл покупки»

Предусловие: Открыта страница https://www.saucedemo.com/. Пользователь зарегистрирован с данными: Логин: standard_user, Пароль: secret_sauce	
Шаги:	Ожидаемый результат:
1. На странице авторизации ввести логин: «standard_user».	Поле активно, в поле текст «standart_user»
2. Ввести пароль: «secret_sauce».	Поле активно, в поле пароль скрыт точками
3. Нажать кнопку «Login».	Открывается страница каталога
4. В открывшемся каталоге выбрать любой товар, например «Sauce Labs Backpack»	Открывается страница товара
5. Нажать кнопку «Add to cart»	Счетчик товаров в корзине становится «1»
6. В главном меню нажать на иконку «Корзина»	Открывается страница корзины с добавленным товаром, количество, цена корректны
7. На странице корзины нажать кнопку «Checkout»	Открывается страница для заполнения информации о получателе
8. Ввести имя «Илья»	Поле активно, в поле текст «Илья»
9. Ввести фамилию «Ермаков»	Поле активно, в поле текст «Ермаков»

10. Ввести почтовый индекс «394003»	Поле активно, в поле текст «394003»
11. Нажать кнопку «Continue»	Открывается страница с информацией о заказе, все данные корректны, сумма заказа корректна
12. Нажать кнопку «Finish»	Появляется сообщение "Thank you for your order!" на странице завершения заказа.

ТС-002: Проверка сортировки товаров

Цель: проверить корректность сортировки товаров на странице каталога по цене и названию.

Таблица 2

Тестовый сценарий ТС-002 «Проверка сортировки товаров»

Предусловие	Пользователь авторизован на сайте https://www.saucedemo.com/ (логин: "standard_user", пароль: "secret_sauce"). Открыта главная страница с каталогом товаров.
Шаги:	Ожидаемый результат:
1. На главной странице найти выпадающий список фильтров.	Открывается выпадающий список фильтров
2. Выбрать в списке фильтров опцию «Name (Z to A)».	Список товаров обновляется в соответствии с фильтром
3. Проверить порядок товаров в каталоге.	Товары расположены корректно
4. Снова открыть выпадающий список фильтров.	Открывается выпадающий список фильтров
5. Выбрать в списке фильтров опцию «Price (low to high)».	Список товаров обновляется в соответствии с фильтром
6. Проверить порядок товаров в каталоге.	Товары расположены корректно

3. Реализация автоматизированных тестов

Для реализации автоматизации были выбраны два ключевых сценария:

– TC-001 – Полный цикл покупки: от авторизации до оформления заказа;
– TC-002 – Проверка сортировки товаров.

Тесты написаны с использованием Selenium WebDriver на Python с применением паттерна PageObject, который обеспечивает модульность и повторное использование кода.

Для реализации тестов был применён паттерн Page Object Model (POM), который позволяет структурировать код, разделяя логику страниц и тестовые сценарии. Для веб-приложения были созданы следующие классы страниц:

– LoginPage – отвечает за страницу авторизации.
– ProductsPage – отвечает за страницу каталога товаров.
– CartPage – отвечает за страницу корзины.
– CheckoutPage – отвечает за страницу оформления заказа.

Каждый класс содержит локаторы элементов страницы и методы для взаимодействия с ними (например, ввод данных, нажатие кнопок, получение текста). Тесты используют эти классы, что делает код более читаемым и упрощает его поддержку.

Реализация класса страницы авторизации представлена на рис. 1.

```python
from selenium.webdriver.common.by import By

class LoginPage:  2 usages
    def __init__(self, driver):
        self.driver = driver
        self.username_input = (By.ID, "user-name")
        self.password_input = (By.ID, "password")
        self.login_button = (By.ID, "login-button")

    def open(self):  2 usages
        self.driver.get("https://www.saucedemo.com/")

    def login(self, username, password):  2 usages
        self.driver.find_element(*self.username_input).send_keys(username)
        self.driver.find_element(*self.password_input).send_keys(password)
        self.driver.find_element(*self.login_button).click()
```

Рис. 1. Реализация класса страницы авторизации

Реализация класса страницы каталога товаров представлена на рис. 2.

```python
from selenium.webdriver.common.by import By

class ProductsPage:  2 usages
    def __init__(self, driver):
        self.driver = driver
        self.add_to_cart_backpack = (By.ID, "add-to-cart-sauce-labs-backpack")
        self.cart_badge = (By.CLASS_NAME, "shopping_cart_badge")
        self.cart_link = (By.CLASS_NAME, "shopping_cart_link")
        self.sort_dropdown = (By.CLASS_NAME, "product_sort_container")
        self.item_prices = (By.CLASS_NAME, "inventory_item_price")
        self.item_names = (By.CLASS_NAME, "inventory_item_name")

    def add_backpack_to_cart(self):  1 usage
        self.driver.find_element(*self.add_to_cart_backpack).click()

    def get_cart_badge(self):  1 usage
        return self.driver.find_element(*self.cart_badge).text

    def go_to_cart(self):  1 usage
        self.driver.find_element(*self.cart_link).click()

    def sort_by(self, option):  2 usages
        from selenium.webdriver.support.ui import Select
        dropdown = self.driver.find_element(*self.sort_dropdown)
        Select(dropdown).select_by_visible_text(option)

    def get_prices(self):  1 usage
        return self.driver.find_elements(*self.item_prices)

    def get_item_names(self):  1 usage
        return self.driver.find_elements(*self.item_names)
```

Рис. 2. Реализация класса страниц каталога товаров

Реализация класса страницы корзины представлена на рис. 3.

```python
from selenium.webdriver.common.by import By

class CartPage:  2 usages
    def __init__(self, driver):
        self.driver = driver
        self.checkout_button = (By.ID, "checkout")

    def go_to_checkout(self):  1 usage
        self.driver.find_element(*self.checkout_button).click()
```

Рис. 3. Реализация класса страницы корзины

Реализация класса страницы оформления заказа представлена на рис. 4.

```python
from selenium.webdriver.common.by import By

class CheckoutPage:  2 usages
    def __init__(self, driver):
        self.driver = driver
        self.first_name_input = (By.ID, "first-name")
        self.last_name_input = (By.ID, "last-name")
        self.postal_code_input = (By.ID, "postal-code")
        self.continue_button = (By.ID, "continue")
        self.finish_button = (By.ID, "finish")
        self.confirmation_message = (By.CLASS_NAME, "complete-header")

    def fill_checkout_form(self, first_name, last_name, postal_code):  1 usage
        self.driver.find_element(*self.first_name_input).send_keys(first_name)
        self.driver.find_element(*self.last_name_input).send_keys(last_name)
        self.driver.find_element(*self.postal_code_input).send_keys(postal_code)

    def continue_checkout(self):  1 usage
        self.driver.find_element(*self.continue_button).click()

    def finish_checkout(self):  1 usage
        self.driver.find_element(*self.finish_button).click()

    def get_confirmation_message(self):  1 usage
        return self.driver.find_element(*self.confirmation_message).text
```

Рис. 4. Реализация класса страницы оформления заказа

4. Экономическая модель оценки целесообразности автоматизированного тестирования

Для оценки целесообразности внедрения автоматизации в процесс тестирования требуется определить затраты на ручное и автоматизированное выполнение тестов. Расчеты будут проводиться исходя из данных, полученных в ходе опроса работников отдела тестирования.

Для расчета инвестиций, необходимых для внедрения и поддержки автоматизированных тестов за определенный период, используется следующая формула:

$$I_p = I_0 + C_0 + \sum_{n=1}^{k} \left(C_e + C_a + C_m \right) \quad (1)$$

I_0 – первоначальные затраты (время настройки среды);

C_0 – разработка автотестов (время × ставка × количество тестов);

k – общее количество тестовых циклов;

C_e – выполнение тестов (в нашем случае ≈ 0, т.к. они запускаются автоматически);

C_a – анализ результатов (время анализа × доля неуспешных тестов × ставка);

C_m – поддержка (вероятность изменений × количество тестов × время × ставка).

Формула оценки расходов на ручное тестирование имеет следующий вид:

$$G_p = G_0 + \sum_{n=1}^{k} \left(G_e + G_a + G_m \right) \quad (2)$$

После того как были автоматизированы тесты, описанные выше, делается вывод о целесообразности автоматизации.

Используя (1), (2), рассчитана величина вложений для реализации автоматизированного и ручного тестирований.

Для автоматизированного тестирования необходимы затраты в 57 585 рублей. Для ручного тестирования – 61 036 рублей.

Таким образом, автоматизировав всего 2 теста, была получена выгода в 10%, при этом, можно ожидать, что последующая автоматизация тестирования приведет к еще большей разнице в эффективности.

Заключение

В ходе статьи было проведено исследование, помогающее оценить целесообразность автоматизации тестирования на примере веб-приложения. Несмотря на затраты на начальной стадии, автоматизация позволяет снизить суммарные издержки, ускорить процесс тестирования и повысить стабильность.

Все поставленные в начале задачи были выполнены.

Список литературы

1. Котляров, В. П. Основы тестирования программного обеспечения : учебное пособие / В. П. Котляров. – 2-е изд. – М. : ИНТУИТ, 2016. – 248 с.

2. Подходы к автоматизации тестирования веб-приложений [Электронный ресурс]. – Режим доступа : https://otus.ru/nest/post/1083

3. Гребенюк, В. М. Оценка целесообразности внедрения автоматизированного тестирования [Электронный ресурс] / В. М. Гребенюк // Науковедение. – 2013. – № 1. – Режим доступа : https://naukovedenie.ru/PDF/13tvn113.pdf

Использование графов знаний для анализа и визуализации зависимостей между веб-сайтами

К. М. Зинченко

Студент бакалавр

В. В. Гаршина

Доцент

Введение

В настоящее время активно идет работа над созданием новых систем, расширением функционала существующих веб-сайтов. В связи с этим появляются проблемы в работе традиционных алгоритмов поиска. Ранее сайты представляли собой набор статических страниц с четкой структурой, где информация была легко индексируема стандартными поисковыми алгоритмами. На данный момент разрабатываются веб-приложения с динамически генерируемым контентом, сложными взаимосвязями между данными и многослойной архитектурой, где информация распределена между различными системами и сервисами.

Одной из основных проблем при создании поисковых систем является то, что современный контент обладает сложной семантикой, где одни и те же термины могут использоваться в разном контексте, а важная информация может быть скрыта в глубине взаимосвязанных данных. Для решения данной проблемы может использоваться семантический поиск, работа которого основана на онтологиях и графах знаний. Они являются мощным инструментом структурирования знаний и организации информации в различных предметных областях. Их основное предназначение заключается в создании формализованной модели определенной сферы знаний, которая позволяет четко определять сущности, их свойства и взаимосвязи. При использовании онтологий в информационном поиске повышается релевантность результатов. Данная работа посвящена разработке и программной реализации семантического извлечения и индексирования информации с использованием LLM и онтологий.

1. Формирование онтологии предметной области

Онтология предметной области факультета строится путем структурированного выделения ключевых сущностей и связей между

© Зинченко К. М., Гаршина В. В., 2025

ними, что позволяет формализовать знания о его организации, учебных процессах и научной деятельности. Такой подход обеспечивает единое семантическое представление данных, пригодное для автоматизированной обработки и анализа [1]. Онтологическое моделирование факультета позволяет устранить семантическую неоднозначность данных и обеспечивает эффективный поиск, классификацию и интеграцию информации [2].

Созданная онтология факультета представлена на рис. 1.

Рис. 1. Онтология факультета

2. Программный обход сайта и извлечение текстовых данных

Обход веб-сайтов с целью извлечения текстовой информации осуществляется с помощью специализированных программных средств, среди которых фреймворк Scrapy является одним из наиболее эффективных решений для крупномасштабного сбора данных [3]. Данный процесс включает несколько ключевых этапов:

– определение структуры обхода;
– загрузка и обработка веб-страниц;
– очистка и нормализация текста;
– сохранение результатов.

В качестве стартового URL выбран домен cs.vsu.ru, исключены переходы по ссылкам, не имеющих данного домена, а также установлена глубина обхода 5. Для выделения текста из HTML-кода, полученного в ходе обхода, сначала исключаются повторяющиеся элементы, навигационные панели, а затем, с помощью встроенных в фреймворк методов, извлекается полезная информация.

Результаты рекурсивного обхода веб-сайта сохраняются в формате JSON в виде объектов со следующими полями:

– url;
– title;
– text.

Пример информации, полученной с веб-страницы, представлен на рис. 2.

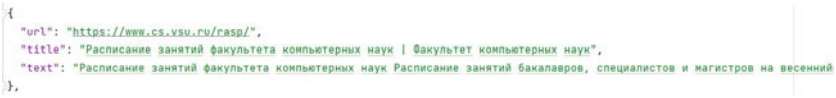


Рис. 2. Пример извлеченной информации

3. Извлечение именованных сущностей из текстов и индексирование текстов

Часто встречающейся задачей при обработке веб-контента является автоматическое выделение именованных сущностей (Named Entity Recognition, NER). Появление ошибок в их определении обычно связано с неоднозначностью естественного языка, недостаточным качеством разметки или ограничениями используемых алгоритмов [4].

Для решения этой задачи применяются методы машинного обучения, основанные на больших языковых моделях (LLM). Выбор именно LLM обусловлен их способностью учитывать контекстные зависимости и высокую вариативность текстовых данных.

В качестве LLM для выделения сущностей была выбрана модель Mistral Large 2, взаимодействие с которой построено на использовании API модели. Помимо выделения сущностей, модель нормализует вид упоминания имен собственных, например, приводя фамилии с инициалами к единому виду: и.о. фамилия. В дальнейшем это будет использоваться при сопоставлении найденных в текстах сущностей и слов, характеризующих определенный класс в онтологии.

Использование заранее определенных синонимов решает проблему контекста предметной области, а именно обнаружение схожих по смыслу, но различных по структуре выражений. К ним можно отнести специфические сокращения, использование которых характерно только

для данной предметной области и применение к ним методов получения векторных представлений не является точным.

В результате обработки текстов, полученных с веб-страниц, индексируются сущности, выделенные из текстов, со словами и синонимичными выражениями, полученными из онтологии предметной области. В случае нахождения соответствия между найденными сущностями и классами онтологии, URL веб-сайта, по которому находится текст, используется при создании экземпляра класса онтологии.

4. Использование заполненной онтологии для семантического поиска

Для решения проблемы семантического поиска, используются онтологии — формализованные модели знаний, описывающие предметную область через классы, экземпляры, атрибуты и отношения. Онтология позволяет перейти от поиска по ключевым словам к поиску по смыслу, что значительно повышает точность и полноту выдачи.

При рассмотрении онтологии предметной области выделяются абстрактные категории, описывающие сущности предметной области, классы организуются в иерархии, что позволяет системе учитывать отношения обобщения-специализации. У каждого класса есть набор атрибутов, которые уточняют его свойства для семантического поиска Помимо этого, классы связаны между собой отношениями, которые определяются экспертом при создании онтологии. Семантический поиск использует эти связи для выявления скрытых зависимостей и контекстуальных связей, которые неочевидны при текстовом сопоставлении. Благодаря этому система может находить не только документы, содержащие ключевые слова, но и связанные с ними сущности.

Поисковый запрос в семантическом поиске интерпретируется не как строка символов, а как структурированный смысловой паттерн. Процесс обработки включает несколько этапов:

– анализ и нормализация запроса — выделение ключевых терминов, устранение неоднозначностей;

– сопоставление с онтологией — идентификация классов и отношений, соответствующих запросу;

– расширение запроса — использование отношений онтологии для включения связанных сущностей.

Заключение

В данной работе был разработан программный модуль, предназначенный для автоматического наполнения онтологии на основе

анализа веб-сайтов с использованием методов обхода страниц и выделения сущностей при помощи языковых моделей (LLM). Реализованное решение позволяет эффективно извлекать структурированные данные из неформатированного текстового контента, идентифицировать ключевые сущности, их атрибуты и взаимосвязи, а затем сохранять их в онтологической модели.

Полученная онтология служит основой для семантического поиска, обеспечивая более точное и осмысленное извлечение информации по сравнению с традиционными методами, основанными на ключевых словах. Благодаря использованию LLM система способна распознавать контекст, разрешать неоднозначности и выявлять скрытые связи между объектами, что значительно повышает релевантность результатов поиска.

Разработанный модуль включает в себя:

– механизм обхода веб-страниц для сбора текстовых данных;

– компонент выделения сущностей с применением LLM, классифицирующий объекты и их отношения;

– онтологическое хранилище, обеспечивающее структурированное представление знаний;

– семантический поисковый механизм, использующий онтологию для интерпретации запросов.

В перспективе планируется интеграция данного модуля в систему обработки тематических данных, что позволит масштабировать решение для работы с различными доменными областями.

Список литературы

1. Грубер, Т. Р. Подход к переводу спецификаций переносимых онтологий / Т. Р. Грубер – М. : KNOWLEDGE SYSTEMS LABORATORY, 1993. – 27 с.

2. Стааб, С. Справочник по онтологиям / С. Стааб, Р. Студер. – М. : Springer, 2009. – 832 с.

3. Документация Scrapy [Электронный ресурс] – Режим доступа : https://docs.scrapy.org/en/latest/index.html

4. Лампл, Г. Нейронные архитектуры для распознавания именованных объектов. / Г. Лампл, М. Баллестерос, С. Субраманиан, К. Каваками, К. Даер. // Carnegie Mellon University : NAACL-HLT (Сан Диего, 2-4 июня 2016 г.). – Калифорния, 2016. – С. 260-270.

5. Анализ текстовых данных с помощью NLTK и Python [Электронный ресурс] – Режим доступа: https://habr.com/ru/companies/otus/articles/774498/

Исследование явной схемы численного интегрирования бигармонического уравнения

Д. С. Кораблин

Студент бакалавр

М. И. Попов

Доцент

Введение

Одним из самых распространенных методов численного интегрирования краевых задач для уравнений в частных производных в настоящее время остается метод конечных разностей. Конечно-разностные схемы разделяются на явные и неявные. Преимущество явных схем состоит в том, что они являются абсолютно устойчивыми. Данный факт исключает исследование схем на устойчивость, что является отдельной и порой сложной задачей. Существенный недостаток неявных схем – необходимость решать системы алгебраических уравнений. Порядок таких систем может доходить до сотен тысяч. Для их решения требуются огромные вычислительные мощности. Явные схемы лишены данного недостатка. Расчет по явной схеме не требует больших затрат памяти ЭВМ, однако он может занимает довольно много времени. Еще одно преимущество явных схем – это возможность распараллеливания вычислений. В данной работе построена и исследована явная конечно-разностная схема для решения задачи о свободной конвекции в квадратной каверне.

1. Постановка задачи

Конвективное движение вязкой несжимаемой жидкости в квадратной области при однородном подогреве сбоку в квадратной каверне моделируется с помощью неоднородного бигармонического уравнения:

$$\frac{\partial^4 \Phi(X,Y)}{\partial X^4} + 2\frac{\partial^4 \Phi(X,Y)}{\partial X^2 \partial Y^2} + \frac{\partial^4 \Phi(X,Y)}{\partial Y^4} = -1 \qquad (1)$$

с граничными условиями

$$\Phi(0,Y) = \Phi(1,Y) = \Phi(X,0) = \Phi(X,1) = 0, \qquad (2)$$

© Кораблин Д. С., Попов М. И., 2025

$$\frac{\partial\Phi(0,Y)}{\partial X}=\frac{\partial\Phi(1,Y)}{\partial X}=\frac{\partial\Phi(X,0)}{\partial Y}=\frac{\partial\Phi(X,1)}{\partial Y}=0, \qquad (3)$$

где $\Phi(X,Y)$ – функция тока, X,Y – декартовы координаты.

Для того чтобы использовать явную схему в стационарной задаче предположим зависимость функции тока от фиктивного времени θ. Прибавим к правой части уравнения (1) производную по фиктивному времени $\frac{\partial\Phi(X,Y,\theta)}{\partial\theta}$, и добавим начальное условие равенства нулю функции тока в начальный момент времени. Получим следующую начально-краевую задачу

$$\frac{\partial\Phi(X,Y,\theta)}{\partial\theta}+\frac{\partial^4\Phi(X,Y,\theta)}{\partial X^4}+2\frac{\partial^4\Phi(X,Y,\theta)}{\partial X^2\partial Y^2}+\frac{\partial^4\Phi(X,Y,\theta)}{\partial Y^4}=-1 \qquad (4)$$

$$\Phi(X,Y,0)=0, \qquad (5)$$

$$\Phi(0,Y,\theta)=\Phi(1,Y,\theta)=\Phi(X,0,\theta)=\Phi(X,1,\theta)=0, \qquad (6)$$

$$\frac{\partial\Phi(0,Y,\theta)}{\partial X}=\frac{\partial\Phi(1,Y,\theta)}{\partial X}=\frac{\partial\Phi(X,0,\theta)}{\partial Y}=\frac{\partial\Phi(X,1,\theta)}{\partial Y}=0, \qquad (7)$$

Решение задачи (4)-(7) сходится к решению задачи (1)-(3) при $\theta\to\infty$.

2. Построение конечно-разностной схемы

Для построения конечно-разностной схемы временной отрезок разделен на равные промежутки $\Delta\theta$. В каждый момент времени $k\Delta\theta,\ \ k=1,...$ область решения $\overline{D}$ системы заменена сеткой с шагами ΔX, ΔY, $\overline{w}_{\Delta X,\Delta Y}=\{(X_i,Y_j)=(i\Delta X,j\Delta Y)\}\quad i=0,..,n,\ j=0,..,m$. Вместо функции непрерывного аргумента на $\overline{D}$ рассматривается функция дискретного аргумента $\Phi(X_i,Y_j,\theta_k)_{\Delta X,\Delta Y,\Delta\theta}$, которая обозначается $\Phi_{i,j}^k$. Выбор разбиения осуществлен таким образом, чтобы в него попадали точки границы области и центр области.

Конечно-разностную аппроксимацию производной $\frac{\partial\Phi(X,Y,\theta)}{\partial\theta}$ в произвольном узле (i,j) получим применением конечно-разностного оператора первого порядка

$$T\Phi=\left.\frac{\partial\Phi(X,Y,\theta)}{\partial\theta}\right|_{i,j,k}=\frac{\Phi_{i,j}^{k+1}-\Phi_{i,j}^k}{\Delta\theta}+O(\Delta\theta). \qquad (8)$$

Для построения конечно-разностного аналога частных производных $\partial^4\Phi(X,Y,\theta)/\partial X^4$ и $\partial^4\Phi(X,Y,\theta)/\partial Y^4$ используем центрально-разностный оператор 2-го порядка примененный дважды по соответствующей переменной [1]:

$$B_{\Delta 1}\Phi = \left.\frac{\partial^4\Phi}{\partial X^4}\right|_{i,j,k} = \frac{\Phi^k_{i+2,j} - 4\Phi^k_{i+1,j} + 6\Phi^k_{i,j} - 4\Phi_{i-1,j} + \Phi^k_{i-2,j}}{(\Delta X)^4} + O\left[(\Delta X)^2\right], \quad (9)$$

$$B_{\Delta 3}\Phi = \left.\frac{\partial^4\Phi}{\partial Y^4}\right|_{i,j,k} = \frac{\Phi^k_{i,j+2} - 4\Phi^k_{i,j+1} + 6\Phi^k_{i,j} - 4\Phi^k_{i,j-1} + \Phi^k_{i,j-2}}{(\Delta Y)^4} + O\left[(\Delta Y)^2\right]. \quad (10)$$

Аппроксимацию смешанной производной получим аналогично

$$B_{\Delta 2}\Phi = \left.\frac{\partial^4\Phi}{\partial X^2\partial Y^2}\right|_{i,j,k} = \frac{1}{\Delta X^2\Delta Y^2}\left(\Phi^k_{i+1,j+1} - 2\Phi^k_{i+1,j} + \Phi^k_{i+1,j-1} - 2\Phi^k_{i,j+1} + \right.$$
$$\left. +4\Phi^k_{i,j} - 2\Phi^k_{i,j-1} + \Phi^k_{i-1,j+1} - 2\Phi^k_{i-1,j} + \Phi^k_{i-1,j-1}\right) + O\left[(\Delta X)^2, (\Delta Y)^2\right] \quad (11)$$

Начальное условие (5) на сетке приобретет вид

$$\Phi^0_{i,j} = 0, \quad i,j = \overline{0,n}. \quad (12)$$

Дискретный аналог граничных условий, накладываемых на саму функцию, получим сужением условий (6) на граничные узлы сетки

$$\Phi^k_{0,j} = \Phi^k_{n,j} = \Phi^k_{i,0} = \Phi^k_{i,n} = 0, \quad k = 1,2,..., i = \overline{0,n},\ j = \overline{0,n}. \quad (13)$$

Аналог граничных условий, накладываемых на частные производные первого порядка (7), находим из соотношения для конечно-разностного аналога первой производной.

$$\frac{-3\Phi^k_{0,j} + 4\Phi^k_{1,j} - \Phi^k_{2,j}}{2\Delta X} = \frac{3\Phi^k_{n,j} - 4\Phi^k_{n-1,j} + \Phi^k_{n-2,j}}{2\Delta X} =$$
$$= \frac{-3\Phi^k_{i,0} + 4\Phi^k_{i,1} - \Phi^k_{i,2}}{2\Delta Y} = \frac{3\Phi^k_{i,n} - 4\Phi^k_{i,n-1} + \Phi^k_{i,n-2}}{2\Delta Y} = 0. \quad (14)$$

Обозначим $B_\Delta\Phi = B_{\Delta 1}\Phi + 2B_{\Delta 2}\Phi + B_{\Delta 3}\Phi$. Непрерывная задача (4)-(7) аппроксимируется маршевой итерационной схемой, каноническая форма которой [2]

$$\frac{\Phi_{k+1} - \Phi_k}{\Delta\theta} + B_\Delta\Phi_k = -1, \quad k = 0,1,..., \quad \Phi_0 = 0. \quad (15)$$

3. Вычисление оптимального итерационного шага

На равномерных сетках $\Delta X = \Delta Y = h$ и $\Delta\theta = \tau$ введем конечномерное гильбертово пространство H_h и рассмотрим в нем оператор $B = h^4 B_h$ не зависящий явно от h.

Докажем положительную определенность оператора B. Поскольку $B = B_1 + 2B_2 + B_3$ и линейная комбинация положительно-определенных операторов с положительными коэффициентами положительно определена [3], достаточно показать лишь положительную определенность операторов B_1, B_2, B_3.

Матрица B_1 представляет собой блочную квазидиагональную матрицу, каждый блок которой симметричная пятидиагональная матрица

$$B_1(n) = \begin{pmatrix} 6 & -4 & 1 & 0 & \cdots & 0 \\ -4 & 6 & -4 & 1 & \cdots & 0 \\ 1 & -4 & 6 & -4 & \cdots & 0 \\ 0 & 1 & -4 & 6 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 6 \end{pmatrix}. \qquad (16)$$

Вычисление определителя пятидиагональной матрицы

$$\Delta(n) = \begin{vmatrix} 1 & a_1 & a_2 & 0 & \cdots & 0 \\ a_1 & 1 & a_1 & a_2 & \cdots & 0 \\ a_2 & a_1 & 1 & a_1 & \cdots & 0 \\ 0 & a_2 & a_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{vmatrix} \qquad (17)$$

сводится к решению системы разностных уравнений [92]:

$$\begin{gathered} A_n - A_{n-1} + a_2^2 A_{n-3} - a_2^4 A_{n-4} + a_1 B_{n-1} - a_1 a_2 B_{n-2} = 0, \\ B_n + a_2 B_{n-1} - a_1 A_{n-1} = 0. \end{gathered} \qquad (18)$$

Здесь $A_k = \Delta(k)$, а $B_{k-1} = \Delta(1, 2/k)$ – минор, получаемый из определителя $\Delta(k)$ вычеркиванием первой строки и второго столбца. Исключая B_{n-1} и B_{n-2} из первого уравнения получим характеристическое уравнение

$$z^5 + (a_2 - 1)z^4 + (a_1^2 - a_2)z^3 + a_2(a_1^2 - a_2)z^2 + a_2^3(1 - a_2)z - a_2^5 = 0. \quad (19)$$

При $a_1 = -4/6$, $a_2 = 1/6$ все корни равны $\lambda_k = 1/6, k = 1..6$, поэтому

$$\det[B_1(n)] = 6^n A_n = 6^n (C_1 + C_2 n + C_3 n^2 + C_4 n^3 + C_5 n^4)\left(\frac{1}{6}\right)^n = \\ = (C_1 + C_2 n + C_3 n^2 + C_4 n^3 + C_5 n^4), \quad (20)$$

причем постоянные C_1, C_2, C_3, C_4, C_5 определяются из начальных условий $B_1(0) = 1, B_1(1) = 6, B_1(2) = 20, B_1(3) = 50, B_1(4) = 105$. Вычислив постоянные, получим $\det[B_1(n)] = 1 + \frac{7}{3}n + \frac{23}{12}n^2 + \frac{2}{3}n^3 + \frac{1}{12}n^4$. Поскольку $\det[B_1(n)] > 0 \quad \forall n \in \mathbb{N}$, $B_1(n)$ удовлетворяет критерию Сильвестра. Матрица B_3 симметрична и в другом ортогональном базисе при замене i на j совпадает с матрицей B_1, то есть B_3 и B_1 подобны, поэтому B_3 положительно определен. Матрица B_2 представляет собой произведение положительно определенных квазидиагональной и квазитрехдиагональной матриц. В силу перестановочности этого произведения оно положительно определено. Таким образом, оператор B положительно определенный.

Перепишем схему (15) в виде

$$\Phi_{k+1} = S\Phi_k - \tau, \quad n = 0,1,..., \quad \Phi_0 = 0, \quad S = E - \tau B_h. \quad (21)$$

S – в оператор перехода. Норма оператора перехода вычисляется по формуле

$$\| S \| = \max_k |\mu_k|, \quad (22)$$

где μ_k – собственные значения оператора S, тогда из (21) следует

$$\mu_k = 1 - \frac{\tau}{h^4}\lambda_k, \quad (23)$$

где λ_k – собственные значения оператора B, что дает возможность сформулировать задачу минимакса

$$\| S \| = \min_{\tau > 0} \max_k | 1 - \frac{\tau}{h^4}\lambda_k |. \quad (24)$$

Поскольку оператор B положительно определен, все его собственные значения положительны, и задача минимакса имеет следующее решение [4]:

$$\| S \| = \frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}} = \rho_0, \quad \tau = \tau_0 = \frac{2h^4}{\lambda_{max} + \lambda_{min}}. \qquad (25)$$

Таким образом, мы получили не только выражение для нормы оператора перехода, но и значение оптимального итерационного шага.

Заключение

С помощью метода конечных разностей построена явная конечно-разностная схема для численного решения задачи о свободной конвекции в квадратной каверне. Проведено исследование полученной схемы. Доказана положительная определенность оператора разностной схемы. Получено выражение для вычисления нормы оператора перехода и оптимального итерационного шага.

Список литературы

1. Попов, М. И. Интегрирование бигармонического уравнения по неявной схеме / М. И. Попов // Вестник Воронежского государственного университета инженерных технологий. – 2018. – Т. 80, № 2(76). – С. 114–118.

2. Ряжских, В. И. Численное интегрирование бигармонического уравнения в квадратной области / В. И. Ряжских, М. И. Слюсарев, М. И. Попов // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. – 2013. – № 1. – С. 52-62. – EDN OIDGRK.

3. Самарский, А. А. Введение в теорию разностных схем / А. А. Самарский. – М. : Наука, 1971. – 552 с.

4. Фаддеев, Д. К. Вычислительные методы линейной алгебры / Д. К. Фаддеев, В. Н. Фаддеева. – М. : ФИЗМАТЛИТ, 1963. – 735 с.

Применение нейронных сетей для ранжирования товаров

А. В. Котов

Студент бакалавр

А. В. Акимов

Старший преподаватель

Введение

В условиях стремительного роста объемов онлайн-торговли и количества представленных товаров на платформах электронной коммерции особую актуальность приобретает задача организации эффективного и интуитивного поиска. Пользователь, сталкиваясь с огромным ассортиментом продукции, ожидает получить максимально релевантные результаты сразу после ввода запроса. Именно поэтому качество поисковой выдачи напрямую влияет на общее впечатление от взаимодействия с сайтом — так называемый User Experience.

Практика показывает, что большинство пользователей начинают взаимодействие с маркетплейсами и интернет-магазинами именно с поисковой строки, а не с навигации по категориям или главной странице. Поэтому точность, скорость и уместность поисковых результатов становятся критически важными факторами, определяющими уровень удовлетворенности пользователя и его дальнейшие действия. Чем быстрее пользователь находит нужный товар, тем выше вероятность того, что он совершит покупку, что, в свою очередь, положительно сказывается на конверсии и общих бизнес-метриках платформы [1].

Конверсия — это один из ключевых показателей эффективности онлайн-платформы. В контексте электронной коммерции конверсией называют соотношение числа пользователей, совершивших целевое действие (например, покупку), к общему числу посетителей сайта, выраженное в процентах.

Современные нейросетевые подходы позволяют значительно улучшить качество ранжирования товаров в поисковой выдаче за счёт более глубокого понимания смысла пользовательских запросов и семантической близости к описаниям товаров. Такие модели превосходят традиционные алгоритмы сопоставления ключевых слов и

© Котов А. В., Акимов А. В., 2025

обеспечивают более точную интерпретацию намерений пользователя, что делает поиск не только эффективным, но и персонализированным.

1. Постановка задачи

Традиционные методы, основанные на ручных правилах, эвристиках или классических алгоритмах машинного обучения (например, логистическая регрессия, градиентный бустинг), часто оказываются недостаточно эффективными в современных условиях.

Основной причиной этого является неспособность традиционных подходов полноценно учитывать смысловую составляющую естественного языка, в котором формулируются запросы пользователей. Такие методы, основаны не полнотекстовом поиске, то есть ищет совпадения ключевых слов, частотами терминов и другими статистическими характеристиками, не учитывая контекст, синонимы и семантическую близость между словами.

В отличие от них, нейронные сети, особенно современные архитектуры на основе трансформеров, способны эффективно обрабатывать текстовую информацию, выявлять скрытые взаимосвязи между запросом и описанием товара, а также обучаться на больших объёмах данных. Достигается это за счет работы не с текстом атрибутов товаров (названия, описания и т.д.), а с векторами, в которых закодирована семантическая информация о товаре. Такие вектора получаются на выходе из нейронной сети, в которую будут передаваться все товары. Это делает их перспективным инструментом для построения систем интеллектуального ранжирования [2].

2. Создание алгоритма нейросетевого ранжирования

Разработанная система ранжирования товаров представляет собой гибридную архитектуру, совмещающую возможности полнотекстового и векторного поиска на базе поискового движка Elasticsearch. Такой подход позволяет одновременно учитывать как синтаксическое сходство между запросом и описанием товара, так и их семантическую близость. Основой системы является взвешенная формула, комбинирующая векторный и полнотекстовый поиск.

Используется стандартный механизм полнотекстового поиска Elasticsearch. В качестве метрики сходства между запросом и текстовыми полями товара (например, названием или описанием) применяется межстрочное расстояние Левенштейна (Levenshtein distance) [3], позволяющее учитывать опечатки, перестановки символов и прочие неточности ввода. Расстояние Левенштейна между двумя строками s и t рассчитывается следующим образом:

$$D(i,j)=\begin{cases} 0, если\ i=0\ и\ j=0, \\ i, если\ j=0, \\ j, если\ i=0, \\ min\begin{pmatrix} D(i-1,j)+1, \\ D(i,j-1)+1, \\ D(i-1,j-1)+m \end{pmatrix}, иначе, \end{cases} \quad (1)$$

где i, j – длинны строк s и t соответственно, $m=0$, если $s[i]=t[j]$, иначе $m=1$.

Каждому товару на этапе парсинга одновременно с сохранением основных полей таких, как название, описание и т.д. сопоставляется векторное представление (эмбеддинг), сгенерированное при помощи нейросетевой модели обработки естественного языка. Эти эмбеддинги сохраняются в Elasticsearch наряду с остальными полями товара.

При выполнении запроса пользовательский текст также преобразуется в вектор, и поиск осуществляется по косинусному расстоянию между вектором запроса и векторами товаров, что позволяет учитывать семантическую близость, даже при отсутствии точных совпадений слов. Косинусное расстояние между двумя векторами a и b определяется как:

$$\text{cosine distance} = 1 - \frac{a*b}{\|a\|*\|b\|}. \quad (2)$$

Гибридный алгоритм ранжирования объединяет два независимых источника информации о схожести между пользовательским запросом и товаром. Итоговая оценка в гибридном подходе вычисляется с помощью формулы взвешенной суммы:

$$Score = A*Score_{text} + B*Score_{vector}, \quad (3)$$

где A, B – коэффициенты (веса) для полнотекстового и векторного поиска соответственно. Веса определяются вручную на этапе проектирования системы. $Score_{text}, Score_{vector}$ – оценки релевантности полнотекстового и векторного поиска соответственно.

3. Реализация поисковой системы

В рамках научной работы была реализована поисковая система, ориентированная на поиск и ранжирование товаров с использованием нейросетевых моделей. Для реализации backend-части проекта был использован язык программирования Python и фреймворк FastAPI,

обеспечивающий высокую производительность, асинхронную обработку запросов и удобную структуру для построения REST API.

Для семантического поиска была использована библиотека SentenceTransformers, предоставляющая инструменты для преобразования текстовых запросов и описаний товаров в векторное представление. Это позволило производить сравнение по смысловому содержанию, а не только по ключевым словам.

Для векторизации использовались следующие модели.

- all-mpnet-base-v2 — одна из самых мощных моделей из семейства Sentence-BERT, основанная на архитектуре MPNet. Она обеспечивает высокую точность в задачах поиска и семантического сопоставления текстов. Подходит для английского языка и демонстрирует отличные результаты в бенчмарках по семантическому поиску.

- multi-qa-mpnet-base-dot-v1 — модель, обученная на многозадачном корпусе вопросов и ответов, оптимизированная для задач поиска и извлечения релевантной информации. Использует dot-product для быстрого сравнения эмбеддингов. Поддерживает английский язык.

- distiluse-base-multilingual-cased-v2 (Multilingual MiniLM) — компактная и быстрая мультиязычная модель, поддерживающая 15+ языков, включая русский. Подходит для мультиязычных приложений, где важно сохранять баланс между точностью и производительностью.

Система позволяет преобразовывать как пользовательские поисковые запросы, так и тексты товаров в векторы, после чего производилось сравнение с помощью косинусного расстояния или dot-продукта, в зависимости от используемой модели. На выходе пользователь получал отсортированный по релевантности список товаров.

Система разработана с учетом реальных потребностей пользователей и особенностей электронной коммерции, обеспечивая удобный интерфейс взаимодействия, высокую скорость обработки запросов и высокую релевантность поисковой выдачи.

Был создан веб-интерфейс, позволяющий пользователю ввести поисковый запрос и получить релевантную выдачу товаров. Интерфейс обеспечивает мгновенный отклик (ElasticSearch в связке в библиотекой SentenceTransformers позволяет получать поисковую выдачу за 0.2-0.5 секунд в зависимости от размера модели) и интуитивную визуализацию результатов, что значительно улучшает пользовательский опыт. Реализация интерфейса позволила протестировать работу всей системы в условиях, приближенных к реальному пользовательскому сценарию, и оценить качество поиска на практике.

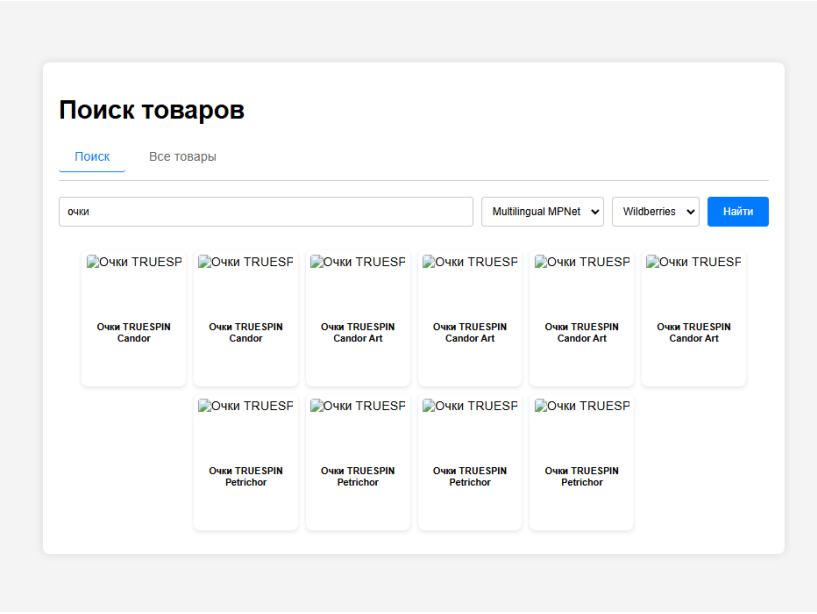


Рис. 1. Интерфейс поисковой системы

Также был разработан специализированный алгоритм парсинга товаров с различных интернет-площадок, который позволяет автоматически собирать и обновлять базу данных товаров. Парсер извлекает ключевую информацию, такую как название, описание, изображения и ссылки на товары, что обеспечивает актуальность и полноту представленной информации. Эта функциональность позволила построить релевантный поисковый индекс и обеспечить масштабируемость системы.

Поисковая система использует описанный выше гибридный подход оценки релевантности товаров и сочетает в себе классические методы извлечения информации и современные нейросетевые алгоритмы сопоставления смыслов, что делает ее эффективным инструментом для решения задачи точного и логически обоснованного ранжирования товаров в ответ на пользовательские запросы.

Для оценки качества работы моделей использовалась адаптированная версия метрики METEOR [4], специально разработанная под задачу ранжирования товаров в поисковой выдаче. Эта метрика, названная RankMETEOR, учитывает как наличие релевантных документов в выдаче, так и их порядок, что позволяет

более точно измерять соответствие между гипотезой модели и эталонным списком результатов.

4. Метрика RankMETEOR

Метрика METEOR (Metric for Evaluation of Translation with Explicit ORdering) была предложена в качестве более чувствительного средства оценки качества машинного перевода по сравнению с более ранними метриками, такими как BLEU. В отличие от BLEU, фокусирующегося на точности n-грамм [4], METEOR учитывает точные совпадения, синонимы, морфологические формы, а также порядок слов, что делает её более соответствующей человеческому восприятию качества перевода.

Формально, метрика METEOR основывается на сопоставлении гипотезы (перевода, полученного моделью) с эталонным переводом на уровне слов. При этом вычисляются две ключевые величины — точность (precision) и полнота (recall). Пусть m — количество совпавших слов между гипотезой и эталоном, t — общее число слов в гипотезе, а r — количество слов в эталоне. Тогда:

$$P = \frac{m}{t},\ R = \frac{m}{r}. \tag{4}$$

Далее вычисляется гармоническое среднее точности и полноты с заданным весом α, обычно в пользу полноты:

$$F_{mean} = \frac{P * R}{\alpha * P + (1-\alpha) * R}. \tag{5}$$

Следующим важным компонентом METEOR является штраф за фрагментированность совпадений, который зависит от количества непрерывных блоков совпадений (chunks). Блоком считается последовательность совпадающих слов, расположенных в одинаковом порядке в гипотезе и в эталоне. Пусть c — количество таких блоков, тогда штраф (penalty) рассчитывается следующим образом:

$$\text{Pen} = \gamma\left(\frac{c}{m}\right)^{\beta}, \tag{6}$$

где γ и β — эмпирически подобранные параметры. Итоговый балл METEOR вычисляется как:

$$\text{METEOR} = F_{\text{mean}} * (1 - \text{Pen}). \quad (7)$$

Таким образом, даже при высоком уровне совпадения, значительная дезорганизация порядка слов приводит к снижению оценки за счёт штрафа.

В научной работе была предложена адаптация метрики METEOR для задачи оценки качества поисковой выдачи. Ключевая идея заключается в том, что вместо сравнения отдельных слов, как это делается в оригинальной METEOR, сопоставляются документы, входящие в результат поисковой системы. Таким образом, каждый документ рассматривается как атомарная единица сравнения, аналогичная слову в контексте машинного перевода.

В дальнейшем данная адаптированная метрика будет обозначаться как RankMETEOR.

5. Тестирование разработанного программного модуля

Анализ результатов показал, что все протестированные модели продемонстрировали хорошие результаты на англоязычном наборе данных, как в случае с простыми, так и с более сложными поисковыми запросами. Независимо от длины запроса или его лексической сложности, модели уверенно справлялись с задачей ранжирования документов по релевантности.

Средние значения метрики RankMETEOR по всем поисковым запросам для каждой модели распределились следующим образом:

При работе с англоязычным датасетом, модель mpnet и модель qa-mpnet показали наивысшую производительность, обе получив среднюю оценку 0.8230. Модель multilingual показала немного более низкое, но всё ещё высокое значение — 0.7334, тогда как miniLM продемонстрировала самую скромную эффективность со средней оценкой 0.5337. По поисковым выдачам видно, что сам смысл слов алгоритм понимает. К примеру, алгоритм понимает, что скрипка, гитара, барабаны – это музыкальные инструменты. Изначально веса были распределены поровну (0.5 и 0.5). При изменении весов, средние оценки остаются в примерно в этом же диапазоне. В основном изменяется только порядок товаров.

При переходе от англоязычного датасета к русскоязычному наблюдается резкое снижение качества работы алгоритма ранжирования для запросов, требующих понимания контекста для получения релевантной выдачи. При этом полнотекстовый поиск работает исправно ввиду того, что ElasticSearch поддерживает работу на любых

языках. Оценки метрики RankMETEOR падают по всем запросам, которые требуют работы с векторами что свидетельствует о неспособности моделей точно определять релевантность результатов при работе с русским текстом. Поисковые выдачи на русском языке не оцениваются метрикой RankMETEOR т.к. реализованный алгоритм, очевидно, не способен уловить смысл слов и контекст. Такое поведение закономерно и связано прежде всего с тем, что ни одна из используемых моделей — включая mpnet, qa-mpnet, miniLM и multilingual [5] — не обучалась напрямую на русскоязычных данных [6], либо объем таких данных в их обучающих корпусах был недостаточным.

Даже модель multilingual, формально поддерживающая 50 языков, включая русский, не демонстрирует приемлемого качества в задачах смыслового сопоставления и ранжирования русскоязычных документов. Это объясняется тем, что мультиязычные модели обучаются сразу на множестве языков, что неизбежно снижает их специализацию на конкретном языке, особенно если этот язык представлен в обучающем датасете фрагментарно.

На сегодняшний день в русскоязычном сегменте отсутствуют по-настоящему крупные и качественные предобученные языковые модели, сопоставимые с англоязычными аналогами вроде GPT-4, BERT или LLaMA. Большинство доступных решений либо мультиязычные (например, mBERT, XLM-R), либо маломасштабные и устаревшие (например, RuBERT, russianGPT), что ограничивает их эффективность в сложных задачах, таких как семантический поиск или генерация ответов. Для достижения сравнимого качества необходима разработка специализированных моделей, обученных на больших русскоязычных корпусах с богатыми семантическими разметками.

Заключение

В ходе выполнения научной работы была разработана и протестирована система семантического поиска и ранжирования товаров с использованием нейросетевых моделей. Были реализованы ключевые компоненты: парсер для автоматического сбора информации о товарах, интерфейс для ввода поисковых запросов и отображения результатов, а также механизм оценки качества поисковой выдачи. Основной акцент был сделан на применение современных моделей из библиотеки sentence-transformers, что позволило значительно улучшить понимание смыслов пользовательских запросов и товарных описаний.

Для объективной оценки результатов была предложена и использована адаптированная метрика RankMETEOR, учитывающая не только полноту совпадений, но и порядок релевантных документов. Это позволило детально сравнить работу различных моделей и выявить

наиболее эффективные из них на англоязычном наборе данных. Результаты показали, что модели mpnet и qa-mpnet демонстрируют наивысшее качество ранжирования по сравнению с другими архитектурами.

Однако при переходе к русскоязычным запросам и данным качество резко ухудшилось. Это связано с тем, что все использованные модели изначально обучались на англоязычных корпусах и не адаптированы к русскому языку. Кроме того, в настоящий момент на рынке практически отсутствуют большие предобученные языковые модели, специализированные для задач смыслового сопоставления именно на русском языке. Этот факт ограничивает возможности построения качественных систем семантического поиска для русскоязычных пользователей и подчеркивает необходимость дальнейших исследований в этом направлении.

Полученные результаты подтвердили потенциал нейросетевых моделей в улучшении поиска и ранжирования товаров в англоязычной среде, а также выявили существенные проблемы при работе с русскоязычными запросами, что определяет направление для будущих улучшений и исследований.

Список литературы

1. 1PS.RU. Маркетплейсы: как заработать свой первый миллион [Электронный ресурс]. – Режим доступа: https://ridero.ru/books/marketpleisy_kak_zarabotat_svoi_pervyi_million/freeText/?srsltid=AfmBOor2TBGuou3Rj8TSTMNRbYdSS5dt3ekQSkrjSG6b_P6KlKcuqTLc

2. Vector Search: The Future of E-commerce Search [Электронный ресурс]: статья. – Режим доступа: https://www.prefixbox.com/blog/vector-search-the-future-of-e-commerce/

3. Расстояние Левенштейна Wikipedia [Электронный ресурс]: статья. – Режим доступа: https://ru.wikipedia.org/wiki_Расстояние_Левенштейна

4. METEOR [Электронный ресурс]: статья. – Режим доступа: https://ru.wikipedia.org/wiki/METEOR

5. sentence-transformers Organization Card [Электронный ресурс]: статья. – Режим доступа: https://huggingface.co/sentence-transformers

6. SentenceTransformers: Pretrained Models [Электронный ресурс]: статья. – Режим доступа: https://sbert.net/docs/sentence_transformer/pretrained_models.html

Система управления финансовым портфелем на базе глубокого обучения с подкреплением

Д. С. Котов

Студент бакалавр

Д. И. Соломатин

Старший преподаватель

Введение

Как известно, поведение финансовых рынков сильно зависит от стихийных бедствий, международных конфликтов, информационных поводов и других событий, которые напрямую с ними не связаны, но создают неопределённость или изменяют настроения инвесторов. В условиях этой нестабильности управление финансовым портфелем становится сложной задачей, для решения которой недостаточно использовать традиционные методы, основанные на статистическом анализе и оптимизации [1].

С развитием области искусственного интеллекта и, следовательно, методов машинного обучения открываются новые возможности для разработки интеллектуальных систем, способных адаптироваться к рыночным условиям в реальном времени. В последнее время большой популярностью пользуется метод обучения с подкреплением — подход, основанный на имитации процесса обучения методом проб и ошибок, который человек использует для достижения своих целей. Он отлично подходит для задач оптимизации, лежащей в основе управления портфелем, позволяя не только повышать эффективность инвестиций, но и уменьшать риски за счёт более гибкой и самообучающейся модели поведения.

Целью данной работы является исследование и реализация подхода к управлению финансовым портфелем на базе метода обучения с подкреплением.

1. Постановка задачи

Обучение с подкреплением – это подход к обучению машины находить оптимальную политику, которая максимизирует вознаграждение и минимизирует наказания. Основная идея обучения с подкреплением, заключающаяся в максимизации вознаграждения,

© Котов Д. С., Соломатин Д. И., 2025

прекрасно согласуется с несколькими областями финансов, включая алгоритмическую торговлю и управление портфелем, поскольку концепция агента, максимизирующего прибыль в неопределенной динамической среде, имеет много общего с инвестором или торговой стратегией, взаимодействующей с финансовыми рынками. [2]

Основными компонентами системы RL являются агент, действия, окружение, состояние и вознаграждение, которые представлены на рис. 1.

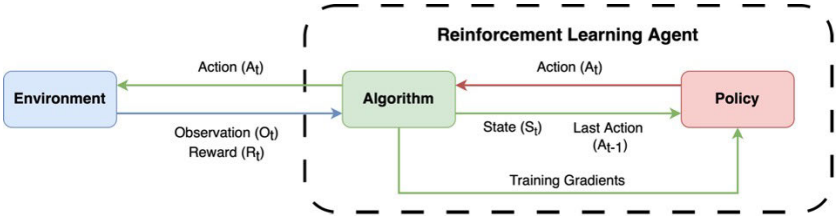


Рис. 1. Компоненты системы обучения с подкреплением

Агент — это обучаемая система, которая принимает решения о перераспределении капитала между активами. Агент реализует детерминированную политику, отображающую состояния среды в действия:

$$\pi\theta : S \to A, \tag{1}$$

где θ — параметры модели, S — множество состояний, A — множество допустимых действий.

На каждом временном шаге t агент получает состояние среды, применяет политику, получает вознаграждение и наблюдает следующее состояние. В процессе обучения параметры обновляются в направлении градиента ожидаемого вознаграждения:

$$\theta \leftarrow \theta + \lambda \bullet \nabla_{\theta} E[R], \tag{2}$$

где λ — скорость обучения, $E[R]$ — ожидаемое суммарное вознаграждение.

Состояние — это информация, доступная агенту в момент времени t. Оно состоит из:

$$s_t = (X_t, w_{t-1}), \tag{3}$$

где X_t — ценовой тензор за последние k периодов по каждому активу, а w_{t-1} — распределение капитала на предыдущем шаге.

Действие — это выбор нового распределения капитала между активами, представленного вектором:

$$w_t = a_t = \pi\theta(s_t), \tag{4}$$

где w_t — доля капитала, инвестированная в активы.

Среда представляет собой симуляцию финансового рынка на основе исторических данных. После применения действия агентом w_t, происходят рыночные изменения, и рассчитывается новое состояние и вознаграждение. Предполагается, что:

– агент не оказывает влияние на рынок (zero market impact),
– проскальзывание отсутствует (zero slippage),
– все данные ограничены историческими ценами.

Вознаграждение — это логарифмическая доходность портфеля с учетом транзакционных издержек, рассчитываемая по формуле:

$$r_t = \ln(\mu_t \bullet (y_t \bullet w_{t-1})) \tag{5}$$

где y_t — вектор относительных изменений цен, w_{t-1} — портфель на предыдущем шаге, μ_t — коэффициент, учитывающий транзакционные издержки.

Накопленное вознаграждение за эпизод длиной T рассчитывается как:

$$R = (1/t)\sum_{t=1}^{T} r_t \tag{6}$$

и является целевой функцией, которую агент стремится максимизировать.

Таким образом, задача управления портфелем формализована как задача обучения агента в частично наблюдаемой стохастической среде. Основной целью агента является максимизация логарифмической доходности с учётом рыночной динамики и транзакционных издержек, а каждое решение — это перераспределение капитала, основанное на текущем состоянии и стратегии.

2. Архитектура интеллектуальной системы

Центральное звено системы — агент, реализующий политику обучаемую с использованием алгоритма Policy Gradient.

В качестве модели, реализующей политику, используется специализированная сверточная нейронная сеть EIIE (Ensemble of Identical Independent Evaluators), предназначенная для обработки тензорных представлений временных рядов финансовых данных. Архитектура является модификацией оригинального подхода, предложенного в работе [3] и адаптирована под задачу многоканального анализа активов в условиях наличия транзакционных издержек.

Входной тензор X_t имеет размерность:

$$X \in \mathbb{R}^{B \times C \times M \times T}, \tag{7}$$

где B — размер батча (batch size), $C = 3$ — количество каналов: цена закрытия, максимум, минимум, M — количество активов (без учёта кэша), T — глубина временного окна (history length).

Архитектура модели построена на последовательном применении двух сверточных блоков с активацией ReLU и завершается выходной сверточной операцией с нормализацией softmax. Структура включает следующие этапы:

Первый сверточный слой:

$$Conv2d(C,\ F1,\ (1,\ k)) \to \mathrm{Re}\,LU, \tag{8}$$

где $C = 3$ — число входных каналов, $F1 = 2$ — количество выходных признаков, $k = 3$ — ширина временного окна.

Второй сверточный слой:

$$Conv2d(F1,\ F2,\ (1,\ T-k+1)) \to \mathrm{Re}\,LU, \tag{9}$$

где $F2 = 20$ — количество выходных признаков, $T-k+1$ — полное свёртывание по временной оси.

Интеграция вектора предыдущего портфеля:

$$last\ stocks \in \mathbb{R}^{B \times 1 \times M \times 1}, \tag{10}$$

данный тензор конкатенируется с выходом сверточной сети по размерности каналов.

Финальный сверточный слой:

$$Conv2d(F2+1,\ 1,\ (1,\ 1)), \tag{11}$$

данный слой приводит данные к размерности, пригодной для оценки весов портфеля.

Добавление компонента кэша:

$$cashbias \in \mathbb{R}^{B \times 1 \times 1 \times 1}, \tag{12}$$

значение добавляется к выходному тензору перед применением softmax, расширяя размерность до $M+1$.

Нормализация выходного вектора:

$$w = Soft\max(output). \tag{13}$$

Обеспечивает то, что результирующий портфель соответствует вероятностному распределению: сумма весов равна единице, и все значения неотрицательны [3].

Визуальное представление процесса обработки данных моделью EIIE представлено на рис. 2.

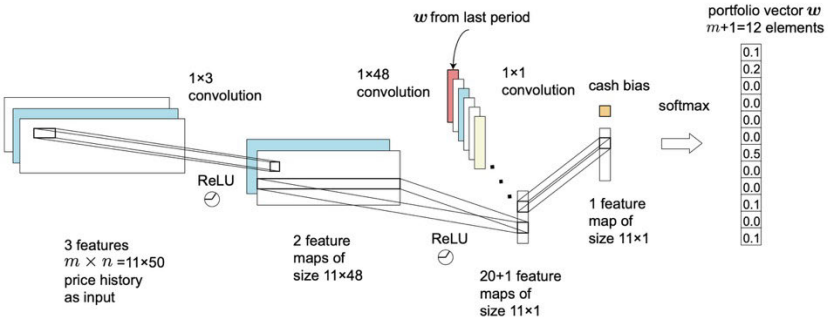


Рис. 2. Архитектура нейросетевой модели EIIE

3. Реализация

Реализация интеллектуальной системы управления портфелем выполнена на языке Python с использованием PyTorch для построения и обучения нейросетевой модели, Pandas и NumPy для обработки временных данных, CCXT для получения рыночных котировок, и Matplotlib для визуализации результатов. Все компоненты реализованы модульно, с возможностью гибкой настройки и расширения. [4]

Торговая среда реализована в модуле portfolio_optimization_env.py , где определён класс PortfolioOptimizationEnv. Этот класс моделирует рыночное окружение, с которым взаимодействует агент. Среда предоставляет стандартный интерфейс — методы `reset()` и `step()`. Метод `reset()` инициализирует новое эпизодическое взаимодействие, а метод `step()` реализует основную динамику: при получении действия от агента (вектора распределения капитала) рассчитывается новое состояние, доходность портфеля и, соответственно, награда, которая затем используется для обновления модели. Внутри среды также реализуется расчёт логарифмической доходности с учётом транзакционных издержек, где вектор предыдущего распределения учитывается через итеративное приближение коэффициента остаточной стоимости после совершения сделок.

Политика, реализующая стратегию агента, была определена в модуле eiie.py в виде класса EIIE, соответствующего архитектуре Ensemble of Identical Independent Evaluators. Эта модель представляет собой ансамбль сверточных сетей, каждая из которых обрабатывает данные по одному активу. В методе forward() нейросети описывается процесс, при котором входной тензор временных рядов проходит через сверточные слои, результатом которого является новый вектор распределения капитала.

Процесс обучения реализован в модуле `policy_gradient.py`, где описана логика оптимизации параметров нейросетевой политики с использованием градиентного подхода. Алгоритм обучения основан на методе Policy Gradient и реализован в виде цикла, в котором агент последовательно взаимодействует со средой и использует полученные награды для корректировки параметров модели.

4. Проведение экспериментов

Экспериментальная часть исследования включает построение обучающей и тестовой выборок, масштабирование данных, обучение агента на исторических данных и последующую проверку его стратегии на отложенной выборке. Все эксперименты выполнялись с использованием исторических котировок криптовалют в формате OHLCV с периодом в 30 минут.

Были созданы обучающая и тестовые среды, для обучения и тестирования агента, которые имели следующие параметры:

- начальное значение капитала 100 000 USD;
- комиссия за сделку 0.25%;
- ценовые данные по активам за последние 50 периодов;
- перераспределение каждые 30 минут;

В обучении и тестировании использовался алгоритм Policy Gradient со следующими параметрами:

- количество шагов 10 000;
- оптимизатор Adam;
- размер батча 100;
- скорость обучения 0.001;
- смещение выборки 0.001;

В эксперименте использовались данные по 11 наиболее популярным криптовалютным активам в 2021 году: BTC, ETH, XRP, LINK, LTC, BCH, DOT, BNB, ADA, XLM и TRX, торгующимся в паре с USDT. [5]

Будем называть средней доходностью портфеля за период такое значение, которое получается при условии, если весь изначальный капитал равномерно распределен между всеми активами на протяжении всего периода. Так как агент имеет возможность не только продавать/покупать, но и удерживать, то валюта, в которой представлен изначальный капитал, также является активом. Следовательно, при начальном капитале 100 000 USD и 12 активах (11 криптовалют + валюта капитала) получаем, что в каждый актив будет инвестировано по 8 333,(3) USD.

Обучающая выборка содержала исторические цены активов с 1 июня по 31 октября 2021 года (рынок восстанавливается после падения, но весьма нестабилен). Изменение стоимости выбранных активов в процентном соотношении за обучающий период представлено на рис. 3.

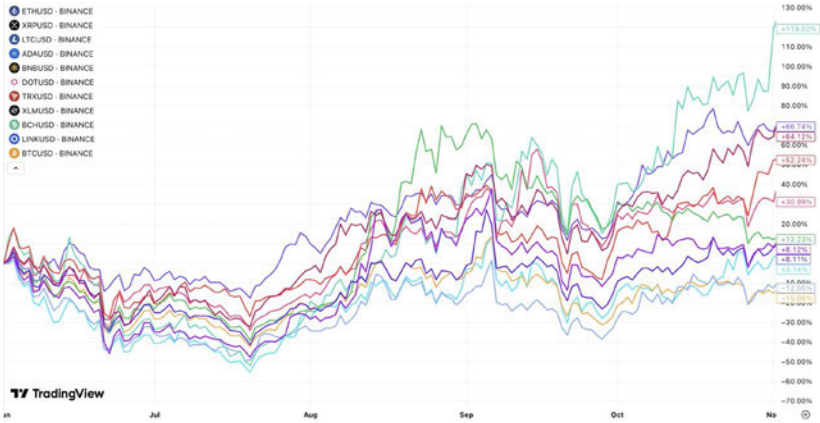


Рис. 3. Изменение стоимости активов во время обучающего периода

По данным из графика можно посчитать среднюю доходность портфеля за обучаемый период. Необходимо сложить все проценты, на которые изменилась стоимость активов, и умножить эту сумму на 8 333,(3) USD, тогда получим значение, равное 28 089,1(6) USD.

Результаты обучения, которые содержат историю изменения портфеля и перераспределение активов, представлены на рис. 4 и рис. 5.

Из графиков видно, что наблюдается высокая волатильность портфеля. Начальное значение около 100 000 постепенно снижается до минимума в районе 60 000, после чего начинается восстановление. Если сравнить итоговую доходность 58 356 USD со средней доходностью 28 089,1(6) USD, то станет ясно, что уже на этапе обучения агент активно адаптирует стратегию под текущие рыночные условия. Он учится использовать краткосрочные сигналы, перестраивая портфель почти на каждом шаге. Такая стратегия требует регулярных транзакций, что при высокой комиссии может быть убыточной без дополнительной фильтрации.

Рис. 4. Изменение стоимости портфеля во время обучения

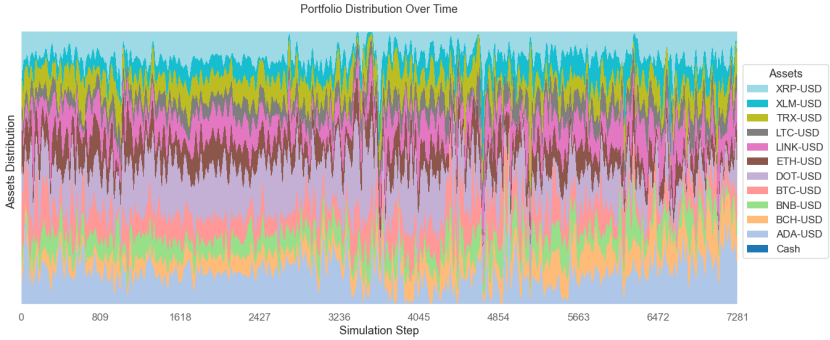


Рис. 5. Распределение активов во время обучения

Тестирование агента проводилось в среде с аналогичными параметрами, но на других данных, чтобы оценить обобщающую способность модели и ее эффективность в условиях, отличных от обучающих. Тестовая выборка содержала исторические цены активов с 1 ноября по 31 декабря 2021 года (рынок падает). Изменение стоимости выбранных активов в процентном соотношении представлено на рис. 6.

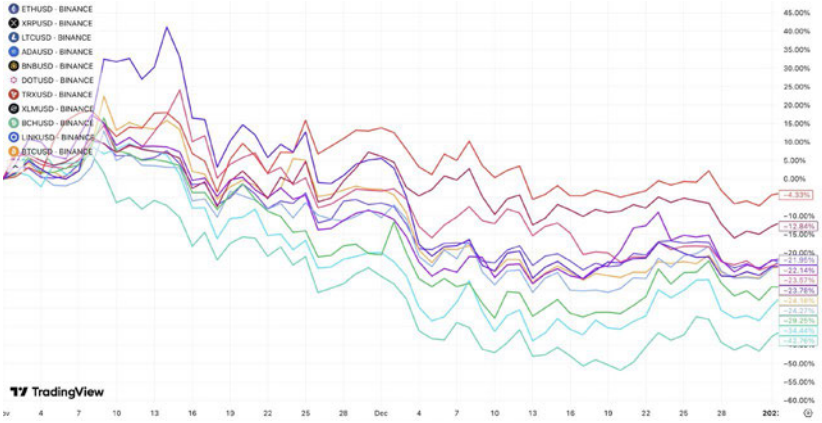


Рис. 6. Изменение стоимости активов во время тестового периода

По данным из графика можно посчитать среднюю доходность портфеля за тестовый период. Необходимо сложить все проценты, на которые изменилась стоимость активов, и умножить эту сумму на 8 333,(3) USD, тогда получим значение, равное -22 417,5 USD.

Результаты тестирования, которые содержат историю изменения портфеля и перераспределение активов, представлены на рис. 7 и рис. 8.

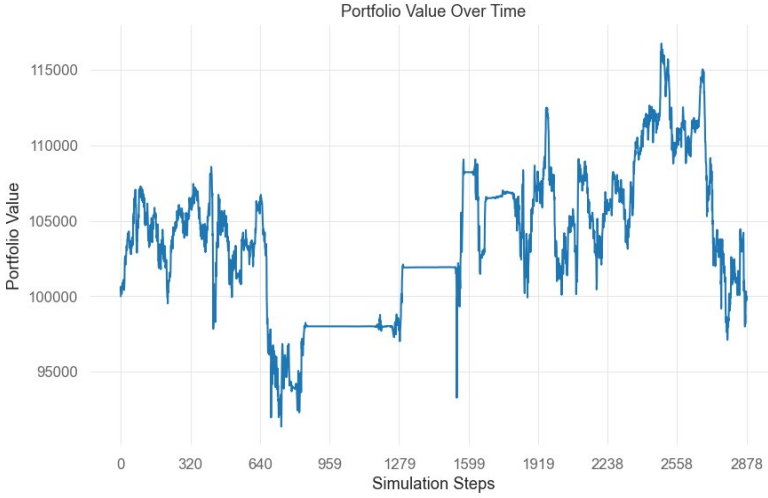


Рис. 7. Изменение стоимости портфеля во время тестирования

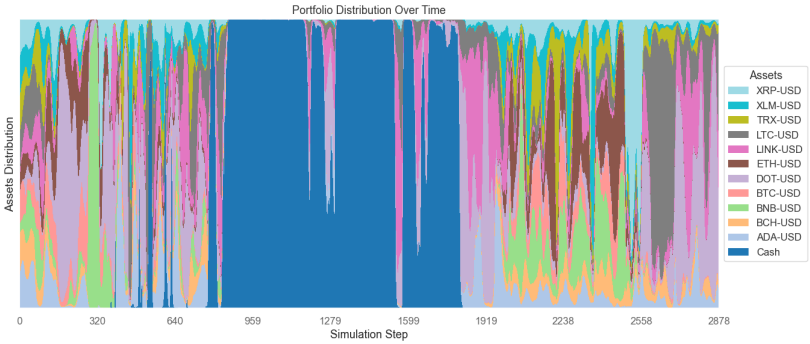


Рис. 8. Распределение активов во время тестирования

Из графиков видно, что наблюдается высокая волатильность портфеля. Начальное значение около 100 000 USD почти на протяжении всего периода находится в 10-процентном ценовом коридоре, а под конец периода наблюдается резкий рост, превышающий отметку в 115 000 USD, и падение к изначальной стоимости. Если сравнить итоговую доходность -2 384,68, USD со средней доходностью -22 417,5 USD, то, не смотря на отрицательное значение, станет ясно, что агент в условиях долгосрочного падения смог адаптироваться: в момент сильных просадок сохранял средства в кеше и находил удачные моменты для входа в позицию. Не стоит также забывать, что агент обучался на нестабильном, но компенсируемом рынке, и ожидал от новых данных подобных условий: локальных отскоков, возможности отыгрыша просадок. Поэтому резкий рост и падение в конце тестового периода могут быть обусловлены излишней доверчивостью агента к активам.

Заключение

В работе был реализован и исследован подход к управлению финансовым портфелем на основе обучения с подкреплением. Модель, основанная на алгоритме Policy Gradient и сверточной нейросети, показала способность адаптироваться к нестабильной рыночной ситуации в обучающей выборке, активно перераспределяя активы и реагируя на изменения цен.

На тестовой выборке, характеризующейся устойчивым снижением рынка, агент продемонстрировал осторожную стратегию — уход в кэш. Такое поведение позволило сохранить значительную часть капитала и избежать дополнительных убытков в фазе падения. Это может интерпретироваться как проявление адаптивности модели и её

способности учитывать риск при отсутствии выгодных торговых возможностей.

Полученные результаты подтверждают перспективность применения обучения с подкреплением для задач управления портфелем. Дальнейшее повышение эффективности возможно за счёт подбора параметров модели и среды, расширения обучающей выборки и внедрения дополнительных рыночных признаков. Это открывает путь к построению более устойчивых и надёжных инвестиционных стратегий по управлению финансовым портфелем.

Список литературы

1. Факторы, влияющие на стоимость ценных бумаг [Электронный ресурс]. – Режим доступа: https://www.indeks.ru/otvety-na-voprosy/faktory-vliyayushchie-na-stoimost-tsennykh-bumag.php

2. Tatsat, H. Machine Learning and Data Science Blueprints for Finance : книга / H. Tatsat, S. Puri, B. Lookabaugh. – 1st ed. – Sebastopol : O'Reilly Media, 2020. – 432 p.

3. A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem [Электронный ресурс]. – Режим доступа: https://arxiv.org/abs/1706.10059

4. Инструменты и библиотеки для машинного обучения [Электронный ресурс]. – Режим доступа: https://sky.pro/wiki/python/instrumenty-i-biblioteki-dlya-mashinnogo-obucheniya/

5. Десять самых значимых криптовалют 2021 года [Электронный ресурс]. – Режим доступа: https://dev.ua/ru/news/cryptoyear-2021

Сравнение характеристик моделей YOLO на примере распознавания объектов по фотографии

Н. Е. Кравцова

Студент бакалавр

Д. И. Соломатин

Старший преподаватель

Введение

YOLO (You Only Look Once) – это современный алгоритм обнаружения объектов в реальном времени, представленный в 2015 году Джозефом Редмоном, Сантошем Диввалой, Россом Гиршиком и Али Фархади в научной статье «You Only Look Once: унифицированное обнаружение объектов в режиме реального времени».

Авторы рассматривают проблему обнаружения объектов как задачу регрессии, а не классификации, пространственно разделяя ограничительные рамки и связывая вероятности с каждым обнаруженным изображением с помощью одной свёрточной нейронной сети (CNN).

Благодаря этому революционному методу модели YOLO работают значительно быстрее, чем предыдущие двухэтапные детекторы, сохраняя при этом высокую точность [1].

YOLO может выполнять такие задачи, как классификация, обнаружение и сегментация изображений. Для каждого класса задач предназначены отдельные модели [2].

В данном исследовании YOLO будет рассматриваться только для задачи обнаружения объектов на изображении.

1. Метрики для сравнения моделей YOLO

Для эффективного сравнения моделей YOLO следует учитывать несколько ключевых аспектов, включая показатели производительности, вычислительную сложность и конкретные требования приложения. Рассмотрим подробнее основные метрики для сравнения:

1. Размер изображения (Size). Размер изображения в пикселях, на котором модель обучалась или тестировалась. Этот параметр влияет на детализацию и точность обнаружения объектов. Большие размеры

© Кравцова Н. Е., Соломатин Д. И., 2025

(например, 640x640) могут улучшить mAP, но также увеличивают вычислительные затраты.

2. mAP50-95. Mean Average Precision (mAP) вычисляется как среднее значение точности (AP) для различных порогов IoU (от 0.50 до 0.95 с шагом 0.05). Это ключевой показатель для оценки точности модели в обнаружении объектов, учитывающий как точность, так и полноту. Высокий mAP50-95 указывает на хорошую производительность модели в различных условиях.

3. Скорость процессора и графического процессора (CPU FPS и GPU FPS). Эти метрики измеряют количество изображений, которые модель обрабатывает за одну секунду на CPU и GPU соответственно. Они критичны для реального времени, например, в приложениях видеонаблюдения или автономных системах. Чем выше значения, тем быстрее обработка.

4. Параметры (Params). Количество параметров модели. Эта метрика отражает сложность модели: большее число параметров может повысить точность, но также увеличивает требования к памяти и вычислительным ресурсам. Модели с меньшим числом параметров (например, YOLO Nano) оптимизированы для устройств с ограниченными ресурсами, таких, как смартфоны, видеокамеры с системой распознавания и т. п. (умные устройства).

5. FLOPs. Количество операций с плавающей запятой (в миллиардах операций), необходимых для обработки одного изображения. FLOPs помогают оценить вычислительную сложность модели. Модели с низкими FLOPs (например, 2.7 млрд) наиболее эффективны для встраиваемых систем [3].

Каждый из перечисленных выше показателей помогает исследователям и разработчикам понять, насколько хорошо модель справляется с задачей обнаружения объектов, и выбрать наиболее подходящую модель для конкретного применения.

Наиболее ценным показателем качества является mAP50-95, который представляет собой метрику Mean Average Precision (mAP). Данная метрика показывает, насколько безошибочно нейронная сеть смогла обнаружить объекты на изображениях из валидационной выборки. Если модель обучается и совершенствуется, точность должна расти от эпохи к эпохе.

2. Модели YOLO

За период с 2016 года по 2024 год выпущено 11 версий YOLO (с v1 по v11), имеющих отличия в архитектуре [4]. Для исследования были выбраны наиболее популярные модели: YOLOv5, YOLOv8 и YOLO11.

Наименования всех моделей YOLO включает в себя номер модели и букву, которой обозначается размер и вычислительная сложность модели. Как правило, от «n» до «x» возрастают размер и точность модели. Рассмотрим отличия моделей подробнее:

1. n (nano) – самая маленькая и быстрая, но с низкой точностью детекции. Лучше всего подходит для мобильных устройств.
2. s (small) – компактная, баланс скорости и точности. Подходит для слабых видеокарт и процессоров.
3. m (medium) – модель среднего размера. Точность выше, чем у «s», но модель медленнее.
4. l (large) – крупная модель. Обладает высокой точностью, но требует производительного железа.
5. x (extra large) – максимальная точность, но модель очень медленная. Подходит для серверов и очень высокопроизводительных графических процессоров [5].

В исследовании участвовали модели nano, small и large, что позволило провести сравнительный анализ производительности YOLO.

3. Экспериментальное сравнение производительности моделей

В ходе данного исследования был создан свой датасет, состоящий из 8291 изображения кошек и собак, на основе нескольких готовых датасетов с последующей аугментацией. Это позволило увеличить разнообразие данных.

Для обучения были использованы параметры модели по умолчанию. Модели YOLOv5, YOLOv8, YOLO11 были дообучены на полученном датасете с нуля до получения приемлемых значений mAP, а их предобученные веса уничтожены.

Для обучения использовалась видеокарта GeForce RTX 2060 (6 ГБ). Обучение проходило в течение 10 эпох, размер батча – 16 изображений. Размер входного изображения – 640x640 пикселей.

Время обучения варьировалось в зависимости от архитектуры и сложности модели. Наиболее ресурсоемкими оказались large-версии: 8l потребовала 598 минут (почти 10 часов) для завершения обучения, 5lu — 524 минуты, а 11l — 442 минуты. В то же время компактные модели (nano и small) обучались значительно быстрее: версии 5nu, 8n и 11n завершили обучение за 13 минут, а small-модели (5su, 8s, 11s) — за 21 минуту. Такая существенная разница во времени демонстрирует высокую вычислительную эффективность компактных архитектур при работе с датасетом среднего размера. При этом важно отметить, что увеличение времени обучения large-моделей не привело к пропорциональному росту точности.

Далее обученные модели были протестированы на отдельном тестовом датасете, состоящем из 40 изображений (20 фотографий кошек и 20 фотографий собак), которые не участвовали в обучении. Тестирование проводилось на двух конфигурациях: на процессоре Ryzen 7 8700F и видеокарте GeForce GTX 960 (2 ГБ).

На рис. 1, 2 можно увидеть пример обработанных картинок, который даёт возможность сравнить результаты работы модели по обнаружению объектов. Каждая строка соответствует одному исходному изображению, в пределах строки (по горизонтали) отображается одно и то же фото, обработанное разными small-моделями YOLO.

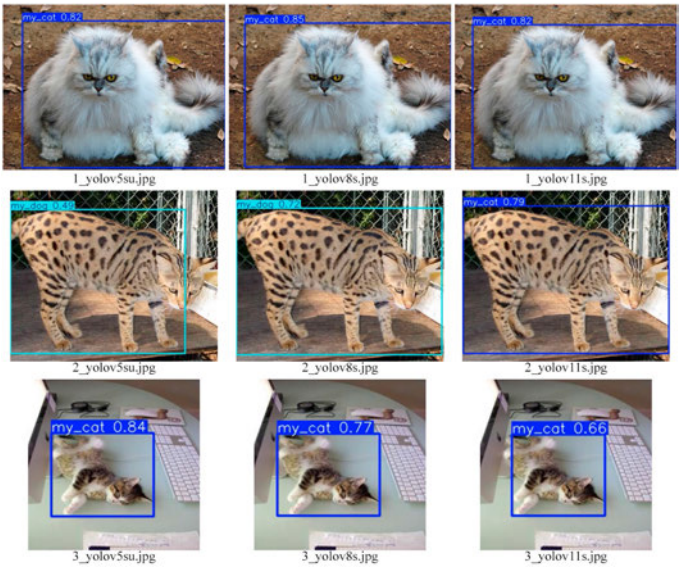


Рис. 1. Результаты детекции кошек small-моделями

У данного типа моделей есть возможность предоставлять информацию в виде выделенной части изображения с указанием класса и вероятности детектирования класса. Часть изображения, ограниченная прямоугольной областью, представляет собой множество точек, относящихся к определённому моделью классу.

Можно заметить, что на рис. 2 на фото «2_yolo11s» после детекции присутствуют две рамки. Это связано с тем, что YOLO работает по принципу скользящего окна: анализирует изображение на разных масштабах и в разных областях, генерируя множество потенциальных рамок для каждого объекта. Если несколько предсказанных рамок

пересекаются (например, модель определила, что объект находится в двух соседних областях), могут появиться дубликаты. Сократить дубликаты можно, например, повысив порог уверенности.

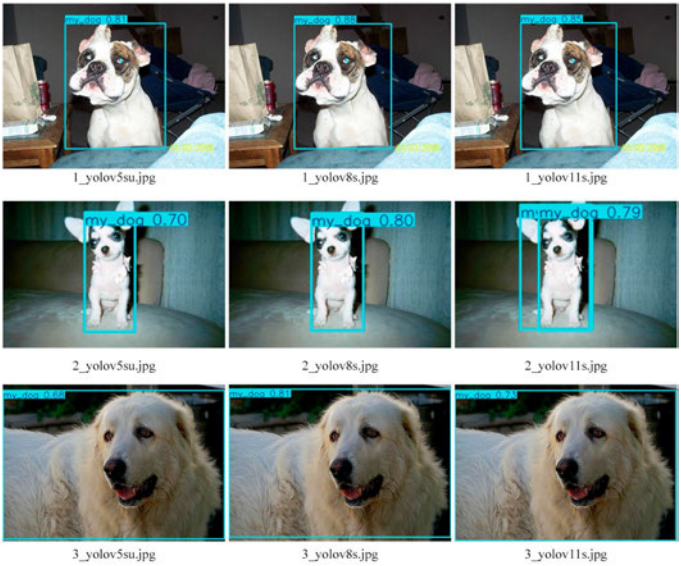


Рис. 2. Результаты детекции собак small-моделями

Результаты тестирования демонстрируют существенные различия в качестве распознавания между моделями семейства YOLO при использовании одинаковых параметров обучения. Наибольшую надежность показали компактные модели YOLOv11: модель 11n верно распознала 85% кошек и 72% собак, а 11s – 83% и 88% соответственно. Худшие результаты показали large-модели (5lu, 8l, 11l) в детекции собак: лишь 60%, 55% и 45% собак соответственно были правильно распознаны, вероятность определения кошек для данных моделей составила 65%, 70% и 80% соответственно.

Стоит отметить, что ошибки классификации преимущественно касались определения собак как кошек, причем вероятность таких ошибочных предсказаний варьировалась от 20% до 55% в зависимости от модели.

Модель 11s не только показала максимальные значения средней вероятности правильного определения (83% для кошек и 88% для собак), но и совершила минимальное количество ошибок распознавания.

По результатам тестирования были получены данные, приведённые в таблице. На рис. 3 представлена гистограмма, позволяющая наглядно сравнить скорость процессора, скорость видеокарты и точность (mAP50-95) для разных моделей.

Таблица

Сравнение производительности моделей YOLO по метрикам

Модель	Размер (пикс.)	mAP 50-95	Скорость процессора (кадры/с)	Скорость видеокарты (кадры/с)	Параметры (млн)	FLOPs (млрд)
5nu	640	59,8	33,13	64,97	2,50	7,07
5su	640	60,1	15,77	40,02	9,11	23,83
5lu	640	55,3	3,92	12,01	53,13	134,67
8n	640	62,3	31,06	66,31	3,01	8,08
8s	640	61,9	13,79	37,98	11,13	28,44
8l	640	55,4	3,28	11,45	43,61	164,83
11n	640	61,0	28,46	59,34	2,58	6,37
11s	640	64,7	13,9	37,66	9,41	21,43
11l	640	57,2	4,71	14,99	25,28	86,83

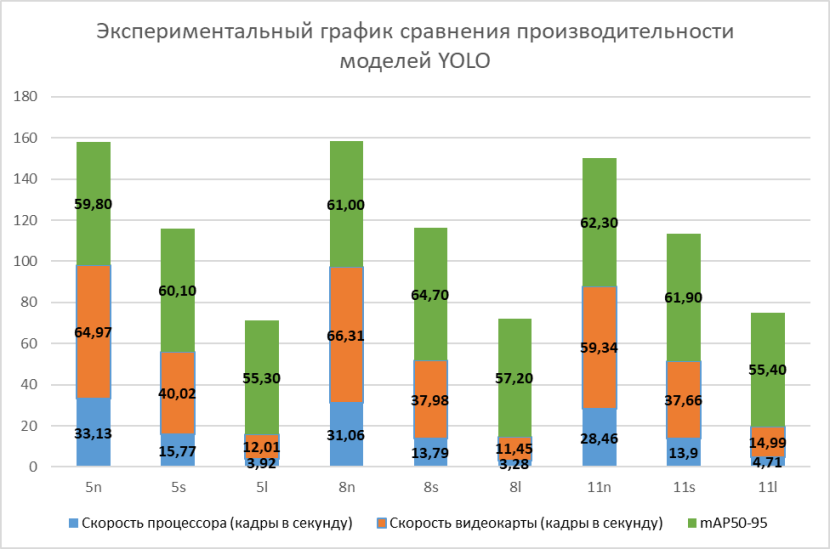


Рис. 3. Экспериментальный график сравнения производительности моделей YOLO

Сравнение моделей YOLO различных версий и размеров показало, что компактные nano-версии демонстрируют наилучший баланс между скоростью обработки и точностью обнаружения объектов. В частности, модели YOLOv5nu и YOLOv8n показали высокую скорость обработки как на CPU (33,13 и 31,06 кадров/с соответственно), так и на GPU (64,97 и 66,31 кадров/с), сохраняя при этом достойные показатели точности (mAP50-95 около 60). Благодаря этому они являются оптимальным выбором для систем, работающих в реальном времени на оборудовании с ограниченными вычислительными ресурсами.

При этом large-модели всех версий работают значительно медленнее (3–5 кадров/с на CPU и 11–15 кадров/с на GPU) и показывают неожиданно низкую точность по сравнению с компактными версиями, что может быть связано с их избыточной сложностью для относительно простого датасета с кошками и собаками.

Особого внимания заслуживает модель YOLO11s, которая при сравнимой с другими small-версиями скорости обработки (13,9 кадров/с на CPU и 37,66 кадров/с на GPU) показала наивысшую точность среди всех тестируемых моделей (mAP50-95 64,7). Этот результат свидетельствует о прогрессивных изменениях в архитектуре YOLO11, позволяющих достигать более высокой точности без существенного увеличения вычислительной нагрузки. Примечательно, что YOLO11l при меньшем количестве параметров (25,28 млн против 43,61–53,13 млн у других large-моделей) показала сопоставимую точность и даже немного более высокую скорость обработки, что подтверждает эффективность оптимизаций в новой версии алгоритма.

Заключение

Для большинства практических задач, особенно требующих работы в реальном времени, оптимальным выбором становятся компактные версии моделей (nano или small), тогда как large-модели стоит рассматривать только для специализированных применений, где их потенциал может быть раскрыт полностью.

Исследование также подтвердило важность выбора модели в соответствии с конкретными требованиями к производительности и точности, поскольку даже в рамках одного семейства разные размеры моделей демонстрируют принципиально разное соотношение этих характеристик.

Список литературы

1. YOLO Object Detection Explained [Электронный ресурс]. – Режим доступа: https://www.datacamp.com/blog/yolo-object-detection-explained

2. How to detect objects on images using the YOLOv8 neural network [Электронный ресурс]. – Режим доступа: https://dev.to/andreygermanov/a-practical-introduction-to-object-detection-with-yolov8-neural-network-3n8c

3. Ultralytics YOLO Docs. Глубокое погружение в метрики производительности [Электронный ресурс]. – Режим доступа: https://docs.ultralytics.com/ru/guides/yolo-performance-metrics/

4. YOLO Explained: From v1 to v11 [Электронный ресурс]. – Режим доступа: https://viso.ai/computer-vision/yolo-explained/

5. Ultralytics YOLO Docs. Обнаружение объектов [Электронный ресурс]. – Режим доступа: https://docs.ultralytics.com/ru/tasks/detect/#models

Разработка системы автоматизированного обнаружения дефектов и аномалий на изображениях

М. С. Ларина

Студент бакалавр

Р. Р. Отырба

Ассистент

Введение

Обнаружение аномалий – это процесс автоматического выявления таких элементов на изображении (пикселей, регионов или объектов), которые существенно отклоняются от нормальных образцов или ожидаемых паттернов, и рассматриваются как дефекты, повреждения или иные нарушения ожидаемого внешнего вида объекта определённой категории. Выявление дефектов и аномалий всегда было важной задачей во многих сферах: промышленность, медицина, сельское хозяйство, видеонаблюдение и т.д.

Актуальность обусловлена необходимостью автоматизации процесса выявления дефектов и аномалий, поскольку традиционные методы ручного обнаружения теряют эффективность при обработке больших объемов данных.

Данная задача является нетривиальной по ряду причин.

- Несбалансированность данных: аномалии обычно представляют малую часть от общего набора событий;
- Неопределённость аномалий: аномалии представляют собой непредсказуемые события, их нельзя точно классифицировать;
- Различная степень выбросов: в разных предметных областях аномалии могут иметь различные формы. Даже в рамках одной области аномалии могут существенно отличаться, что делает сложным определение универсальных критериев их выявления.

Современные подходы к решению этой задачи преимущественно основаны на методах глубокого обучения. Наибольшее распространение получили следующие направления:

- методы на основе реконструкции (reconstruction-based) [1];
- методы на основе синтеза аномалий (synthesizing-based) [2];

© Ларина М. С., Отырба Р. Р., 2025

– методы на основе векторных представлений (embedding-based) [3].

Большинство существующих решений работают по принципу «один класс – одна модель». Это накладывает существенные ограничения на использование таких систем в реальных условиях. Многие исследователи предлагают многоклассовые подходы, которые также имеют некоторое ограничение – модель работает только на тех классах, которые использовались при ее обучении.

Целью данной работы является разработка универсальной системы автоматизированного обнаружения дефектов и аномалий на изображениях с использованием методов глубокого обучения.

1. Предлагаемый подход

В работе выдвигается гипотеза, согласно которой задачу обнаружения аномалий можно трактовать как задачу обнаружения изменений на изображении. Аномалии такие же непредсказуемые события, как и изменения. Если для обнаружения изменений требуется пара разновременных снимков, то в случае с аномалиями аналогичной парой могут выступать нормальный образец и образец с аномалиями.

Такой взгляд позволяет переосмыслить подход к обнаружению аномалий. В рамках предложенного подхода разработана модель, принимающая на вход два изображения – нормальное и аномальное. Оба изображения пропускаются через общий кодер для извлечения признаков, затем признаки поступают в модуль согласования, после чего при помощи декодера формируется карта аномалий (рис. 1). Цель обучения предлагаемого подхода заключается в минимизации расхождения между предсказанной и истинной картой аномалий.

На текущий момент существует лишь одна работа, в которой также используется подход обнаружения аномалий через сравнение изображений [4].

В качестве кодера используется легковесная сеть EfficientNet с замороженными весами. Такой выбор обусловлен стремлением сделать модель универсальной – она не должна адаптироваться под конкретный класс объектов, чтобы сохранить нейтральность и непредвзятость при обработке различных данных.

Целевые объекты на входных изображениях не всегда располагаются одинаково: один и тот же объект может быть немного сдвинут, повернут или масштабирован. Поэтому признаки, извлечённые с помощью кодирующей сети, также будут отличаться по расположению, даже если само содержание схожее. В таких условиях сравнение признаков без предварительного выравнивания может привести к неверным выводам. Для решения этой проблемы

предлагается использовать модуль согласования признаков, который позволяет модели сосредоточиться именно на существенных различиях, связанных с аномалиями, а не на незначительных визуальных отклонениях, вызванных положением объектов или углом обзора.

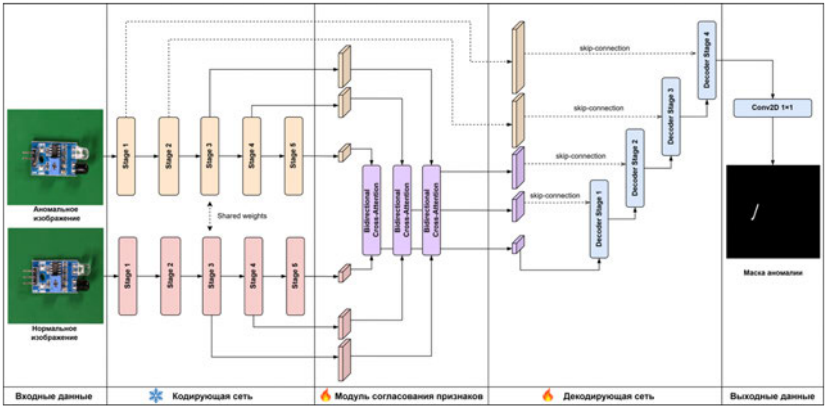


Рис. 1. Архитектура предлагаемой системы

В модуль согласования поступают только признаки среднего и высокого уровней, поскольку низкоуровневые карты содержат слишком локальные и нестабильные детали, которые лишь усложняют обработку.

В качестве модуля согласования предлагается двунаправленный механизм кросс-внимания (1), выполняющий согласование признаков в общем признаковом пространстве в два этапа (рис. 2).

$$CrossAttention(Q,K,V) = \text{Softmax}(\frac{Q \times K^T}{\sqrt{d_k}}) \times V \qquad (1)$$

1. Сначала мы согласуем нормальные признаки относительно аномальных, подавая в механизм кросс-внимания аномальные изображения как запросы, а нормальные как значения и ключи.

2. Затем уточняются аномальные признаки относительно согласованных нормальных, подавая в механизм кросс-внимания согласованные нормальные признаки как запросы, а аномальные как значения и ключи.

Для снижения вычислительной сложности предлагаемого механизма в ходе обработки запросов и ключей, размерность их каналов уменьшается в два раза через линейные слои.

В результате формируются карты, согласованные в признаковом пространстве, приближенном к пространству аномального изображения.

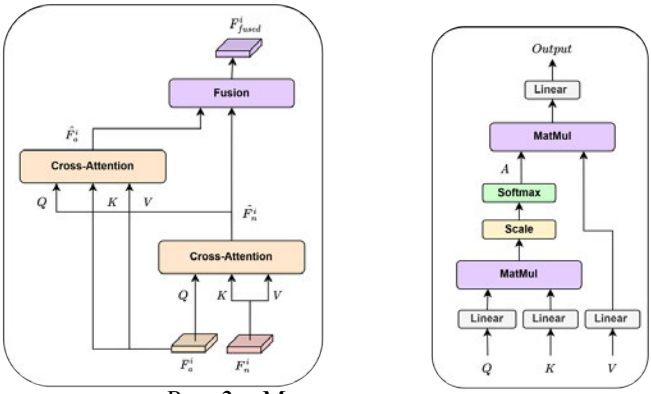


Рис. 2. Модуль согласования

На заключительном этапе признаки поступают в U-Net-подобный декодер, задача которого – восстановление пространственного расположения признаков и формирование карты аномалий. На рис. 3 представлена структура блока декодера, включающего пропускные соединения (skip-connections), которые помогают восстановить пространственные детали, потерянные в процессе кодирования, путём слияния высокоуровневых признаков с признаками, извлечёнными на более ранних этапах кодера.

2. Сравнительные эксперименты

Предлагаемая архитектура была протестирована на наборе данных VisA [5], содержащем более 10 000 изображений высокого разрешения, разделённых на 12 категорий объектов. Для каждой категории доступны изображения с аномалиями и без, а также соответствующие маски сегментации. Набор был разделен на обучающую, тестовую и валидационную выборки в соотношении 8:1:1.

Модель обучалась 40 эпох с начальной скоростью обучения 0.0001. Использовался оптимизатор AdamW и адаптивное снижение скорости обучения (ReduceLROnPlateau, коэффициент 0.5, patience 7 эпох). Аугментация включала горизонтальные и вертикальные повороты.

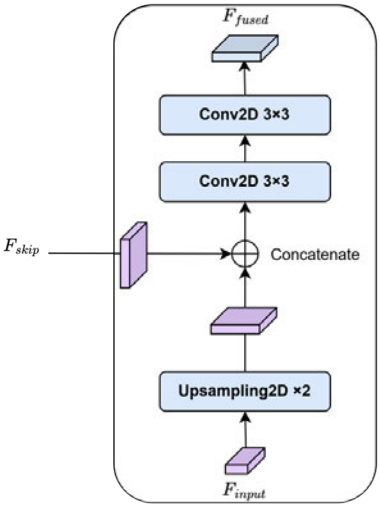


Рис. 3. Блок декодера

В качестве функции ошибки (4) использовалась комбинация функции Дайса и Фокальной ошибки:

$$L_{focal} = -\alpha_t (1 - p_t)^{\gamma} \log p_t \tag{2}$$

$$L_{dice} = 1 - \frac{2 \times \sum (y_{pred} + y_{true}) + \xi}{\sum (y_{pred} + y_{true}) + \xi} \tag{3}$$

$$L_{combo} = \beta \times L_{focal} + L_{dice} \tag{4}$$

Для сравнительного эксперимента были реализованы из упомянутой ранее работы [4] два метода согласования признаков: мягкое и жесткое.

$$F_p^{'} = \text{Softmax}(F_q, F_p^T) \times F_p \tag{5}$$

$$F_p^{'} = F_p(\text{argmax} < F_p, F_q >) \tag{6}$$

В таблице 1 приведены результаты экспериментов на наборе данных VisA по метрике F1-score и AUROC. Для удобства были введены

следующие индексы для классов: 1 – candle, 2 – capsule, 3 – cashew, 4 – chewing gum, 5 – fryum, 6 – macaroni_1, 7 – macaroni_2, 8 – pcb1, 9 – pcb2, 10 – pcb3, 11 – pcb4, 12 – pipe fryum.

Проведенные эксперименты показали, что предлагаемый метод согласования демонстрирует лучшие результаты, как по F1-score, так и по AUROC. Среднее значение F1 для метода двунаправленного кросс-внимания составило 0,7733, в то время как для мягкого и жесткого согласования – 0,7433 и 0,7304 соответственно, что свидетельствует о более высокой точности обнаружения аномалий предлагаемого метода.

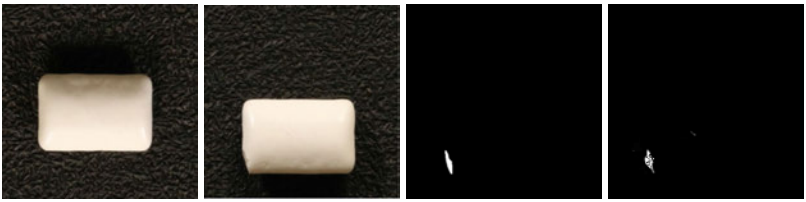
Таблица 1

Результаты сравнительного эксперимента

Классы	**Способ согласования**					
	Кросс-внимание		**Мягкое**		**Жесткое**	
	F1	**AUROC**	**F1**	**AUROC**	**F1**	**AUROC**
1	0,4595	0,6899	0,4054	0,6712	0,3639	0,6507
2	0,9425	0,9789	0,9277	0,9743	0,9072	0,9614
3	0,9662	0,9835	0,9673	0,9834	0,9603	0,9745
4	0,8994	0,9565	0,8838	0,9356	0,8810	0,9292
5	0,9594	0,9790	0,7105	0,8265	0,6807	0,7977
6	0,5186	0,7276	0,5323	0,7379	0,5243	0,7393
7	0,3915	0,7150	0,3506	0,6928	0,3245	0,6424
8	0,8942	0,9471	0,9071	0,9467	0,9118	0,9573
9	0,6951	0,8364	0,7193	0,8332	0,7085	0,8574
10	0,8129	0,8986	0,7758	0,8767	0,7859	0,8818
11	0,7795	0,9359	0,7799	0,9098	0,7667	0,9260
12	0,9608	0.9815	0,9598	0,9789	0,9501	0,9775
Average	0,7733	0,8858	0,7433	0,8639	0,7304	0,8579

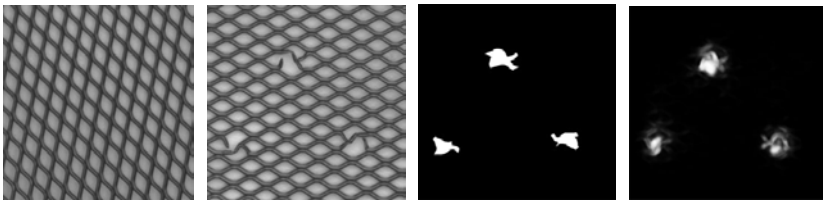
На рис. 4 приведен результат работы предлагаемой модели.

Предложенный подход позволяет обучить модель выявлять аномалии на основе изменений, используя всего одно нормальное изображение. Таким образом, данное решение является универсальным и не требует переобучения модели для работы с новым классом. На рис. 5 приведен результат работы модели на паре изображений из набора данных MVTec-AD. Изображения данного класса не использовались при обучении, но модель всё равно смогла успешно определить аномальные области.



а б в г

Рис. 4. Пример работы модели: а – нормальное изображение, б – аномальное изображение, в – правильная маска, г – предсказанная маска



а б в г

Рис. 5. Пример работы модели на новом классе: а – нормальное изображение, б – аномальное изображение, в – правильная маска, г – предсказанная маска

Заключение

В данной статье была изучена проблема обнаружения дефектов и аномалий на изображениях и предложен новый подход на основе обнаружения изменений. Разработанная модель достигает среднего значения F1-score 0,7733 по всем классам набора данных VisA. Она преодолевает ограничение «один класс – одна модель», характерное для многих существующих решений, и демонстрирует способность обобщать знания и выявлять аномалии на невиданных ранее классах. Таким образом, подход обладает признаками универсальности, хотя для достижения по-настоящему универсального алгоритма потребуется обучение на более широком и разнообразном наборе данных.

Список литературы

1. ReContrast: Domain-Specific Anomaly Detection via Contrastive Reconstruction / J. Guo [et al] // arXiv [Электронный ресурс]. – 2023. – Режим доступа: https://arxiv.org/pdf/2306.02602

2. Zhao, Y. OmniAL: A Unified CNN Framework for Unsupervised Anomaly Localization / Y. Zhao // IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – 2023. – P. 3924–3933.

3. Towards Total Recall in Industrial Anomaly Detection / K. Roth [et al] // arXiv [Электронный ресурс]. – 2022. – Режим доступа: https://arxiv.org/pdf/2106.08265

4. Gao, B. MetaUAS: Universal Anomaly Segmentation with One-Prompt Meta-Learning / B. Gao // OpenReview [Электронный ресурс]. – 2024. – Режим доступа: https://openreview.net/pdf?id=4jegYnUMHb

5. VisA (Visual Anomaly Dataset // Papers with Code [Электронный ресурс]. – Режим доступа: https://paperswithcode.com/dataset/visa

Недостаток публикаций об уязвимостях в программных продуктах: сравнение российских и мировых практик

М. С. Леонов

Студент бакалавр

Д. И. Соломатин

Старший преподаватель

Введение

Уязвимости программного обеспечения остаются ключевым фактором, определяющим уровень информационной безопасности как отдельного предприятия, так и государства в целом. Международная практика раскрытия сведений об уязвимостях сформировалась вокруг стандарта CVE (Common Vulnerabilities and Exposures) и поддерживаемой Национальным институтом стандартов и технологий США базы NVD. Только в 2024 г. в NVD было опубликовано 40 009 записей, что на 38 % превышает показатель 2022 г. и демонстрирует устойчивый рост выявляемых дефектов [1]. Публикация таких данных обеспечивает синхронность между исследователями и вендорами, ускоряет выпуск исправлений и повышает прозрачность процессов разработки. В Российской Федерации централизованным реестром является Банк данных угроз безопасности информации (БДУ ФСТЭК). Однако за тот же период 2024 г. в БДУ было внесено лишь 11 128 записей, то есть около четверти мирового объёма, причём медианная задержка между регистрацией уязвимости в CVE и появлением соответствующей записи в БДУ составила восемь суток [2]. Отставание усугубляется тем, что многие отечественные производители ограничиваются «тихими» патчами без публичных бюллетеней, опасаясь репутационных потерь или ссылаясь на требования регулирования [3]. Недостаточная открытость препятствует своевременному внедрению защитных мер, снижает доверие к отечественным продуктам на внешних рынках и ограничивает возможности академического сообщества проводить репрезентативные прикладные исследования.

Настоящая работа ставит целью комплексное сравнение российских и мировых практик раскрытия информации об уязвимостях,

© Леонов М. С., Соломатин Д. И., 2025

выявление количественных и качественных разрывов, а также формулирование рекомендаций по их сокращению. Научная новизна исследования заключается в систематизации разнородных статистических данных 2020–2024 гг., анализе нормативно-правовых барьеров (включая законопроект № 509708-8 о легализации исследовательской деятельности) и оценке влияния программ Bug Bounty на открытость экосистемы [4]. Практическая значимость проявляется в возможности использования полученных выводов регуляторами для совершенствования БДУ ФСТЭК, корпоративными центрами реагирования на инциденты (PSIRT) — для разработки внутренних политик disclosure, а также вузами — при формировании образовательных траекторий в области кибербезопасности. Статья структурирована следующим образом: после обзора международных и российских нормативов приводится методология сравнительного анализа, далее излагаются полученные результаты и их обсуждение, завершают работу рекомендации и выводы, отражающие перспективы повышения прозрачности национальной системы раскрытия уязвимостей.

1. Теоретические основы и обзор литературы

Международные практики раскрытия уязвимостей опираются на несколько взаимодополняющих нормативных и институциональных оснований. Базовым идентификатором является стандарт CVE, обслуживаемый MITRE и интегрируемый в Национальную базу данных уязвимостей NVD, где к 2025 г. накоплено свыше 290 тыс. записей; в 2023 г. было зарегистрировано 28 902 новых уязвимости, что подтверждает экспоненциальный рост их выявления. На уровне международных стандартов требования к процессу координированного раскрытия сформулированы в ISO/IEC 29147:2018, регламентирующем обязанность поставщика публиковать достоверные сведения о выявленных дефектах, и в сопряжённом ISO/IEC 30111:2019, описывающем жизненный цикл обработки уязвимостей от приёма сообщения до выпуска коррективных патчей. Для многосторонних сценариев взаимодействия эти положения детализированы в техническом отчёте ISO/IEC TR 5895:2022, а операционализацию процессов обеспечивает рамка FIRST PSIRT Services Framework (ред. 2023), задающая типовые наборы сервисов и метрик результативности продуктовых команд реагирования.

Академическая литература акцентирует влияние таких нормативов на зрелость процессов безопасности: эмпирические исследования показывают, что применение ISO-подхода сокращает медианное время исправления критических дефектов на 22–30 % и повышает долю

публичных бюллетеней. Параллельно развивается экономический аспект disclosure через программы Bug Bounty: мировые вендоры ежегодно выплачивают исследователям свыше 100 млн долл., причём один только Google перечислил 12 млн долл. в 2024 г.. В русскоязычном сегменте вопрос disclosure изучается ограниченно; однако отчёты Positive Technologies и аналитика БДУ ФСТЭК фиксируют, что российские публикации составляют около четверти мирового объёма, а средняя задержка появления записи в национальном реестре достигает нескольких недель. При этом локальные программы вознаграждений (Яндекс, Mail.ru, «Лаборатория Касперского») демонстрируют рост: с 2014 по 2023 г. совокупные выплаты превысили 120 млн руб., что способствует постепенному формированию культуры ответственного раскрытия.

Российский правовой контекст задаётся одновременно статьями 272–273 УК РФ и законопроектом № 509708-8, который призван легализовать деятельность «белых хакеров», обязывая их уведомлять вендора в течение пяти суток и воздерживаться от публикации до выхода патча. Литература подчёркивает, что неустойчивость регуляторной базы и репутационные риски остаются ключевыми факторами, сдерживающими открытость. Совокупность приведённых работ формирует теоретико-методологическую основу настоящего исследования, позволяя соотнести международные модели disclosure с российской практикой и определить направления сокращения выявленных диспропорций.

2. Методология исследования

Методологической основой работы является сопоставительный количественно-качественный анализ двух массивов открытых данных об уязвимостях за период 2020–2024 гг.: международного репозитория CVE/NVD, предоставляемого Национальным институтом стандартов и технологий США, и российского Банка данных угроз безопасности информации (БДУ ФСТЭК). Для выгрузки информации из NVD использовался официальный набор JSON-фидов, ежедневно обновляемый NIST, что обеспечивает полноту выборки; данные БДУ были получены посредством автоматизированного парсинга веб-интерфейса с учётом фильтра «тип записи – уязвимость». Каждая запись нормализовалась по трём критериям: идентификатор (CVE-ID), дата первичной публикации и оценка критичности (CVSS v3.1). После удаления дубликатов выборка составила 137 482 CVE-записей и 32 115 записей БДУ за исследуемый интервал. Для оценки задержки disclosure рассчитывалась разность дат между публикацией в CVE и появлением аналогичной записи в БДУ; медиана и межквартильный размах

определялись через непараметрические статистики. Качественный компонент исследования включал контент-анализ нормативно-правовых актов (ст. 272–273 УК РФ, законопроект № 509708-8) и корпоративных политик ответственного раскрытия пяти крупнейших отечественных вендоров, отобранных по объёму выручки в сегменте ПО. Дополнительно изучены отчёты ведущих платформ Bug Bounty (HackerOne, YesWeHack) для верификации статистики вознаграждений. Обработка данных выполнялась в Python 3.12 (библиотеки pandas, scipy, matplotlib). Ограничения исследования связаны, во-первых, с закрытой частью БДУ, не публикуемой ФСТЭК, что потенциально занижает объём российских данных, и, во-вторых, с тем, что корпоративные бюллетени нескольких отечественных производителей (в частности, в секторе САПР) недоступны без договора NDA и потому не включены в выборку. Несмотря на указанные ограничения, сопоставление двух репозиториев и нормативных практик позволяет достоверно оценить масштаб разрыва в прозрачности disclosure между Россией и мировым сообществом и выдвинуть рекомендации по его сокращению.

3. Результаты сравнительного анализа

За пятилетний интервал 2020–2024 гг. международный реестр CVE/NVD продемонстрировал устойчивый экспоненциальный рост: только в 2024 г. опубликовано 40 009 новых записей, тогда как национальный БДУ ФСТЭК за тот же период зафиксировал лишь 11 128 уязвимостей, что эквивалентно отставанию примерно на 73 % по объёму публикаций. Совокупная выборка, очищенная от дубликатов, составила 137 482 CVE-записи против 32 115 записей БДУ; таким образом, на каждую российскую публикацию приходится порядка 4,3 международных. Анализ динамики показал, что разрыв не сокращается со временем: в 2021 г. доля БДУ составляла 26 % от мирового объёма, в 2023 г. — 24 %, а в 2024 г. — 27 %, то есть колеблется вокруг четверти, не демонстрируя тренда к сближению.

Помимо количественного разрыва выявлено значимое запаздывание disclosure. Например, для критических уязвимостей CVE-2024-3094 и CVE-2024-47176 задержка между присвоением CVE и появлением записи в БДУ составила от трёх до пяти дней; агрегированная медиана по всей выборке за 2020–2024 гг. равна восьми дням, а межквартильный размах — 4–18 дней, что ставит под угрозу своевременную защиту отечественных пользователей. Дополнительно 14 % уязвимостей, признанных критическими в CVE (CVSS ≥ 9,0), вовсе отсутствуют в БДУ по состоянию на март 2025 г., что указывает на селективность национального реестра.

Исследование программ поощрения исследователей выявило ещё одну диспропорцию. Согласно отчёту Google, за 2024 г. корпорация перечислила этичным хакерам 11,8 млн USD (≈1,1 млрд руб.) через свои Vulnerability Reward Programs [5]. Для сравнения, крупнейшая российская инициатива «Охота за ошибками» компании «Яндекс» в 2023 г. выплатила 70 млн руб. [6], а VK отчиталась о 13 млн руб. вознаграждений по итогам 2022 г. [7]. Платформа Standoff Bug Bounty, запущенная Positive Technologies в 2022 г., суммарно довела выплаты до 158 млн руб. к концу 2024 г. [8]. Специализированная программа «Лаборатории Касперского» остаётся нишевой: с 2018 г. через неё принято 59 отчётов на общую сумму 81 750 долл. [9]. Даже в агрегированном виде российские выплаты на порядок уступают объёму единственной глобальной программы Google, что иллюстрирует ограниченную вовлечённость внешних исследователей в национальную экосистему disclosure.

4. Обсуждение

Отставание российской системы раскрытия уязвимостей от мировой проявляется не только в статистических показателях, но и в характере управленческих и нормативных процессов, лежащих в её основе. Количественный разрыв — 40 009 публикаций в CVE/NVD против 11 128 записей в БДУ ФСТЭК за 2024 г. — закрепился на уровне порядка 70 % и сопровождается медианной задержкой в восемь суток для сопоставимых инцидентов; показательным примером стали кейсы XZ Utils и CUPS, где информация дошла до БДУ на три и пять дней позже соответственно. Такая инерция означает, что значимая часть отечественных пользователей остаётся без своевременных индикаторов компрометации, а регулятивная статистика не отражает реальной картины угроз.

Ключевое объяснение диспропорции заключается в совокупности правовых, организационных и социокультурных факторов. До недавнего времени поиск уязвимостей без согласия правообладателя формально находился в серой зоне между ст. 272–273 УК РФ, что порождало у исследователей опасения уголовного преследования; лишь в 2024 г. внесённый законопроект № 509708-8 предложил легализовать «ответственный ресёрч», но всё ещё накладывает пятидневный мораторий на публичность. Дополнительную закрытость формируют ведомственные требования: критически важные для государства системы подчиняются принципу «need-to-know», вследствие чего ФСТЭК публикует лишь часть получаемой информации. На корпоративном уровне преобладает репутационный консерватизм: многие отечественные вендоры предпочитают «тихие» патчи, опасаясь

ущерба бренду; при этом небольшие масштабы bugbounty-программ (120 млн руб. совокупно за десятилетие против 12 млн долл. за один год у Google) свидетельствуют о ограниченной вовлечённости внешних исследователей.

Ситуация усугубляется международной изоляцией: санкционные ограничения препятствуют участию российских специалистов в глобальных платформах, тогда как внешние регуляторы требуют повышенной прозрачности при поставке ПО. Опыт Китая, где об обязательном раскрытии сообщается прежде всего государству, демонстрирует альтернативную траекторию централизации информации, но одновременно подчёркивает риски дальнейшей закрытости и возможного использования задержек в наступательных целях. Таким образом, устойчивость российской цифровой инфраструктуры зависит не столько от численного расширения БДУ, сколько от институционального перехода к культуре координированного публичного disclosure, поддерживаемого ясными правовыми гарантиями и экономическими стимулами для всех участников экосистемы.

5. Рекомендации

Повышение прозрачности национальной системы раскрытия уязвимостей требует согласованных действий регуляторов, индустрии и академического сообщества. На государственном уровне целесообразно завершить принятие законопроекта № 509708-8, расширив его норму об освобождении исследователя от ответственности при добросовестном соблюдении процедуры уведомления, а также сократив предусмотренный пятидневный мораторий до срока, синхронизированного с международной практикой responsible disclosure [4]. Одновременно БДУ ФСТЭК следует технологически интегрировать с глобальным реестром CVE путём автоматизированного импорта метаданных и публикации открытого API, что позволит устранить медианную задержку в восемь суток, выявленную в ходе исследования. Регулятор может предусмотреть для отечественных разработчиков поощрительные коэффициенты при сертификации средств защиты, если те публично публикуют бюллетени безопасности и поддерживают bugbounty-программы.

Российским вендорам рекомендуется формализовать процессы реагирования на уязвимости в соответствии с требованиями ISO/IEC 29147:2018 и ISO/IEC 30111:2019, внедрив внутренние Product Security Incident Response Teams (PSIRT) и публично размещённые политики координированного раскрытия. Эффективность PSIRT-модели может оцениваться по контрольным метрикам FIRST PSIRT Services Framework (ред. 2023), включая время до подтверждения отчёта и долю

уязвимостей, устранённых до публичного объявления. Для стимулирования внешних исследований компаниям следует расширять фонды Bug Bounty, публиковать годовые отчёты о количестве принятых заявок и средних выплатах, а также обеспечивать отказ от судебных претензий к исследователям, действовавшим в рамках оговорённого окна конфиденциальности.

Академическим учреждениям предлагается создавать открытые датасеты на базе агрегированных данных CVE и БДУ, что позволит студентам и аспирантам проводить репрезентативные исследования, а также организовывать внутривузовские соревновательные Bug Bounty с переходом лучших решений к индустриальным партнёрам. Совместные рабочие группы вузов, регулятора и бизнеса способны ускорить разработку национальных методических рекомендаций по координированному раскрытию уязвимостей. Реализация указанных мер позволит не только сократить статистический разрыв между российской и мировой практикой, но и повысить доверие к отечественным программным продуктам на международных рынках, обеспечив тем самым дополнительную киберустойчивость цифровой экономики страны.

Заключение

Проведённое сопоставление массивов CVE/NVD и БДУ ФСТЭК за 2020–2024 гг. подтвердило устойчивый количественный и временной разрыв между российской и мировой системами раскрытия уязвимостей. При общем объёме 40 009 публикаций в CVE за 2024 г. национальный реестр за тот же период зафиксировал лишь 11 128 записей, что эквивалентно отставанию примерно на три четверти [1][2]. Медианная задержка появления записи в БДУ относительно CVE составила восемь суток, а 14 % критических уязвимостей (CVSS $\geq$ 9,0) вовсе не были отражены в российской базе к марту 2025 г. Анализ нормативных и организационных факторов выявил, что ключевыми барьерами служат правовые риски, селективная публикация регулятором и репутационный консерватизм вендоров. Предложенные рекомендации включают законодательное закрепление иммунитета для добросовестных исследователей, автоматизированную синхронизацию БДУ с CVE, расширение bugbounty-программ и внедрение процессов PSIRT по стандартам ISO/IEC 29147 и 30111. Реализация этих мер способна устранить задержку disclosure, увеличить полноту национального реестра и тем самым повысить киберустойчивость цифровой экономики. Ограниченность доступных данных по закрытому сегменту БДУ и корпоративным бюллетеням определяет направление дальнейших исследований, которые следует расширить за счёт

сотрудничества регуляторов, академии и отрасли, а также анализа перспективных механизмов обмена, таких как SBOM и VEX. В целом результаты работы подтверждают необходимость институционального перехода России к модели координированного публичного раскрытия, уже зарекомендовавшей себя на международной арене как эффективный способ снижения уязвимости программных экосистем.

Список литературы

1. National Vulnerability Database. NVD Vulnerability Statistics, 2024 [Электронный ресурс]. – Режим доступа : https://nvd.nist.gov/vuln
2. Федеральная служба по техническому и экспортному контролю. Отчёт о функционировании Банка данных угроз безопасности информации за 2024 год [Электронный ресурс]. – Режим доступа : https://bdu.fstec.ru/files/statistics_2024.pdf
3. Российские IT-компании предпочитают не афишировать брешь в безопасности [Электронный ресурс]. – Режим доступа: https://www.rbc.ru/technology/12/03/2025/vuln_disclosure
4. О внесении изменений в Уголовный кодекс Российской Федерации… : проект федерального закона № 509708-8, внесён 14.09.2024 [Электронный ресурс]. – Режим доступа : https://sozd.duma.gov.ru/bill/509708-8
5. Google paid out $12 million via bug bounty programs in 2024 [Электронный ресурс]. – Режим доступа : https://www.securityweek.com/google-paid-out-12-million-via-bug-bounty-programs-in-2024/
6. В 2023 году Яндекс выплатил этичным хакерам 70 млн рублей [Электронный ресурс]. – Режим доступа : https://yandex.ru/company/news/01-12-03-2024
7. Антипов А. VK заплатила багхантерам 37 миллионов за год [Электронный ресурс]. – Режим доступа : https://www.securitylab.ru/news/544064.php
8. Positive Technologies. В 2024 году на платформе Standoff Bug Bounty доля отчётов о наиболее опасных уязвимостях в два раза превысила показатель мировых платформ [Электронный ресурс]. – Режим доступа : https://www.ptsecurity.com/ru-ru/about/news/positive-technologies-v-2024-godu-na-platforme-standoff-bug-bounty-dolya-otchetov-o-naibolee-opasnyh-uyazvimostyah-v-dva-raza-prevysila-pokazatel-mirovyh-platform/
9. Kaspersky. Sustainability Report 2022–2023: Global Transparency Initiative. Раздел «Bug Bounty Program» [Электронный ресурс]. – Режим доступа : https://esg.kaspersky.com/en/ethics-transparency/global-transparency-initiative

Управление доступом к информационным ресурсам с использованием биометрической аутентификации

Д. А. Литвин

Студент бакалавр

А. С. Мальцев

Доцент

Введение

В эпоху цифровых технологий информационная безопасность приобретает критически важное значение как для организаций, так и для частных пользователей. С ростом объёмов передаваемых данных, увеличением числа кибератак и усложнением вредоносных программ традиционные методы аутентификации всё чаще оказываются недостаточно надёжными. В связи с этим возникает необходимость в более надёжных методах защиты.

На сегодняшний день наблюдается устойчивая тенденция к отказу от однофакторной аутентификации ввиду её уязвимости. Согласно исследованию, проведённому «Лабораторией Касперского», из 193 миллионов проанализированных паролей почти половину (около 45% или 87 миллионов) можно взломать менее чем за минуту при помощи современных алгоритмов подбора [1]. Основной причиной столь высокой уязвимости становится использование предсказуемых комбинаций — словарных слов, имён, повторяющихся символов.

Одним из решений, предложенных «Лабораторией Касперского» для повышения уровня безопасности, является внедрение многофакторной аутентификации, которая сочетает два и более независимых метода.

Одним из перспективных решений в этой области является использование биометрической аутентификации. Биометрические системы, базирующиеся на уникальных физиологических и поведенческих характеристиках человека, обеспечивают надёжный контроль доступа к конфиденциальной информации. Благодаря современным алгоритмам машинного обучения они адаптируются к естественным изменениям пользователя, сохраняя высокую точность распознавания даже в нестандартных условиях [2].

© Литвин Д. А., Мальцев А. С., 2025

В данной работе предлагается дополнение на основе биометрической аутентификации, которое может быть использовано в составе многофакторной системы защиты. Целью является не замена существующих решений, а их дополнение — за счёт внедрения нового подхода, способного повысить общую надёжность и устойчивость системы. Предлагаемый метод ориентирован на достижение баланса между высоким уровнем безопасности и удобством для пользователя, а также на возможность адаптации к нестандартным условиям эксплуатации. Таким образом, разработка направлена на расширение арсенала средств многофакторной аутентификации с учётом актуальных вызовов в области информационной безопасности.

1. Выбор метода биометрической аутентификации

В условиях многофакторной аутентификации особое значение приобретает метод биометрической идентификации, который сочетает высокий уровень защиты и удобство для пользователя. Среди различных методов биометрической аутентификации геометрия лица является одним из наиболее предпочтительных вариантов. Этот подход основан на анализе уникальных параметров лица: расстояний между глазами, форм носа, контура челюсти и других ключевых характеристик [3]. Современные алгоритмы машинного обучения обеспечивают высокую точность распознавания, а наличие фронтальной камеры на большинстве устройств делает этот метод особенно удобным и доступным. В итоге геометрия лица обеспечивает баланс между высоким уровнем безопасности и комфортом при проверке подлинности личности.

При сравнении с альтернативными методами биометрической идентификации можно отметить их сильные и слабые стороны. Например, отпечатки пальцев долгие годы успешно применяются для контроля доступа благодаря своей надёжности. Однако этот метод требует соприкосновения с сенсором, что может снижать удобство эксплуатации, особенно в ситуациях загрязнения или при повреждениях кожи пальцев. Сканирование радужной оболочки глаза обеспечивает ещё более высокую точность идентификации за счёт уникальности рисунка радужки, но требует специализированного оборудования и точного позиционирования пользователя относительно камеры. Это ограничивает быстроту и комфорт применения этого метода в реальных условиях.

Голосовая биометрия не требует визуального контакта и проста в использовании – достаточно произнести определённую фразу для идентификации. Однако качество распознавания голоса чувствительно к шуму окружающей среды, а также может меняться при заболеваниях

голосового аппарата у пользователя. Кроме того, голос легко записать и воспроизвести, что снижает надёжность метода без дополнительных мер защиты. Существуют и другие способы биометрической идентификации, например геометрия ладони или рисунок кровеносных сосудов на руке, но они требуют специфического оборудования и реже используются в массовых приложениях. В целом геометрия лица оказывается наиболее универсальным и сбалансированным решением.

Метод аутентификации по геометрии лица сочетает несколько важных преимуществ, а именно надёжность идентификации и простоту использования. Распознавание лица осуществляется бесконтактно – достаточно взглянуть в камеру, что особенно удобно в повседневной практике. Современные алгоритмы могут корректировать изменения в кадре: они учитывают различные условия освещённости, повороты головы и незначительные изменения внешности (например, причёску или очки), что позволяет системе адаптироваться к нестандартным условиям эксплуатации. Уникальное сочетание множества признаков лица обеспечивает высокий уровень безопасности: подделать или воспроизвести чужое лицо довольно сложно. Таким образом, геометрия лица обеспечивает баланс между высоким уровнем защиты и комфортом пользователя, делая её предпочтительным методом для интеграции в многофакторные системы аутентификации.

2. Реализация решения

Для реализации системы биометрической аутентификации по лицу был выбран язык программирования Python, так как он предоставляет широкие возможности для обработки изображений и имеет мощные библиотеки для машинного обучения. В качестве системы управления базами данных (СУБД) использовалась PostgreSQL, обеспечивающая надёжное хранение и эффективную обработку биометрических данных. Для реализации алгоритмов распознавания лиц была выбрана библиотека dlib, которая зарекомендовала себя как высокоточный и производительный инструмент в данной области.

Библиотека dlib предоставляет современные средства для детектирования и распознавания лиц, включая глубокие нейронные сети на основе архитектуры ResNet-34[4]. Она предоставляет уже обученную модель примерно на 3 млн лиц, которая демонстрирует очень высокую точность. На стандартном бенчмарке LFW (Labeled Faces in the Wild) она достигает примерно 99.38% правильного распознавания [5]. Это делает хорошим решение для поставленной задачи.

В приложении было предусмотрено сохранение всех настроек в JSON-файле, что обеспечивает гибкость и простоту конфигурации системы. Это позволяет быстро настраивать параметры для множества

машин, что особенно ценно в корпоративных средах с большим числом рабочих станций. Использование JSON также упрощает интеграцию с другими инструментами управления, повышая универсальность системы.

Для обеспечения безопасности, масштабируемости и надёжности была выбрана клиент-серверная архитектура приложения. Такой подход позволяет разделить функции между клиентской и серверной частями, обеспечивая эффективное распределение нагрузки и централизованное управление данными.

Клиентское приложение работает следующим образом: периодически захватывает изображения с камеры пользователя и анализирует найденные лица. Общую логику работы можно описать следующим образом:

– Нет распознанных лиц: система проверяет активность пользователя (движение мыши, нажатия клавиш и т.п.). Если активности нет, приложение переходит в режим ожидания возвращения пользователя, после чего начинает распознавание лиц снова. Если же пользователь активен и ещё не истёк отведённый интервал для распознавания, производится повторная попытка захвата кадра и анализа.

– Распознано более N лиц: по соображениям безопасности система считает ситуацию подозрительной и выполняет выход в окно входа Windows (выходит из активной сессии пользователя). Порог обнаружения (количество лиц, после которого происходит сброс) может настраиваться, исходя из условий эксплуатации.

– Распознано менее N лиц: приложение для каждого пользователя формирует 128-мерный дескриптор лица (вектор признаков) и отправляет его на сервер для проверки доступа. Сервер сверяет присланный дескриптор с базой разрешённых пользователей. Если сервер подтверждает доступ этому лицу, клиент сбрасывает таймер распознавания и продолжает работу. Если же сервер сообщает, что ни одному из лиц доступ не разрешён, клиент завершает сеанс и переводит систему к экрану входа Windows.

– Истечение времени: если за отведённый интервал ни одно лицо так и не было надёжно распознано, а активность за устройством присутствует – клиент также выполняет выход из активной сессии пользователя. Таким образом, при любом «исключительном» состоянии (многочисленность пользователей, неизвестное лицо) приложение автоматически блокирует сеанс и переходит к экрану входа.

На рис. 1 представлена общая блок-схема работы клиентской части приложения.

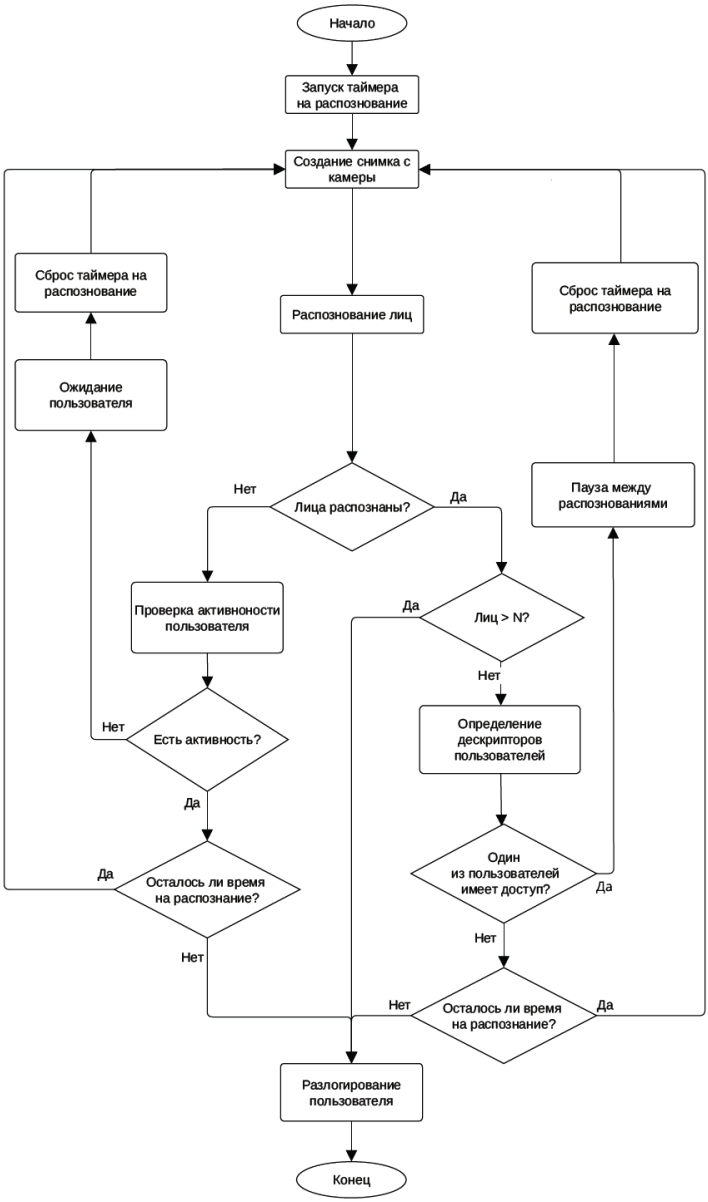


Рис. 1. Общая блок-схема клиентской части приложения

В свою очередь, серверная часть приложения выполняет:

– хранение дескрипторов в базе данных в зашифрованном виде;
– добавление и удаление дескрипторов пользователя;
– сравнение полученных от клиента дескрипторов с имеющимися в базе;

Для обеспечения безопасности дескрипторов пользователей, передача данных между клиентом и сервером осуществляется по защищённому протоколу HTTPS, который обеспечивает шифрование канала связи и защищает данные от перехвата, подмены или несанкционированного доступа.

3. Интерфейс клиентской части приложения

Клиентское приложение снабжено удобным графическим интерфейсом, который обеспечивает быстрый доступ ко всем основным функциям и параметрам системы распознавания.

Интерфейс представлен на рис. 2 и в левой части окна располагается вертикальная панель с кнопками для управления приложением.

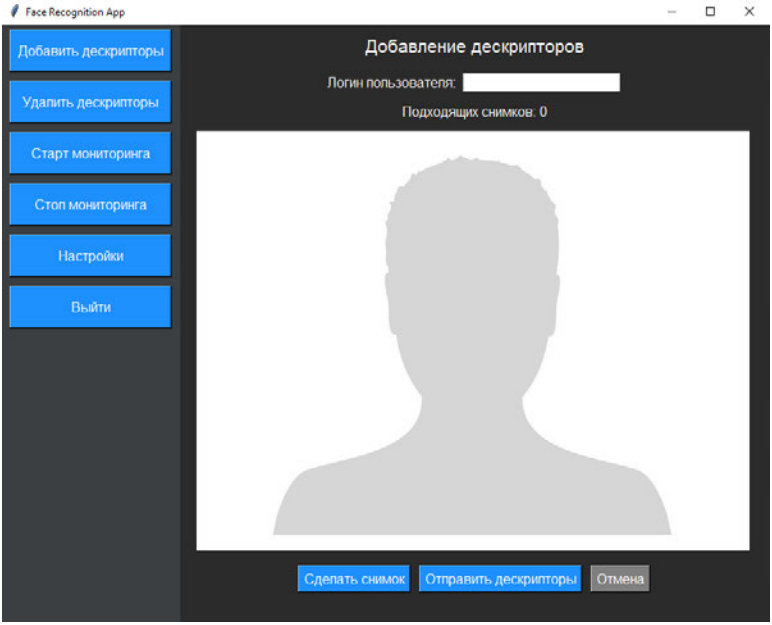


Рис. 2. Окно добавления дескрипторов пользователю

Меню содержит следующие функции:

– Добавить дескрипторы — переход в меню добавления дескрипторов пользователя, которое отображено на рис. 2. Данная операция осуществляется через логин пользователя, а максимальное количество снимков ограничивается в настройках.

– Удалить дескрипторы — удаление всех, хранимых в базе данных, дескрипторов пользователя по указанному логину.

– Старт и стоп мониторинга — запуск и остановка работы программы по мониторингу лица пользователя за компьютером.

– Настройки — открытие окна для конфигурации параметров работы клиента.

В настройках представлен обширный набор параметров для конфигурации работы приложения под необходимые условия. Окно настроек представлено на рис. 3.

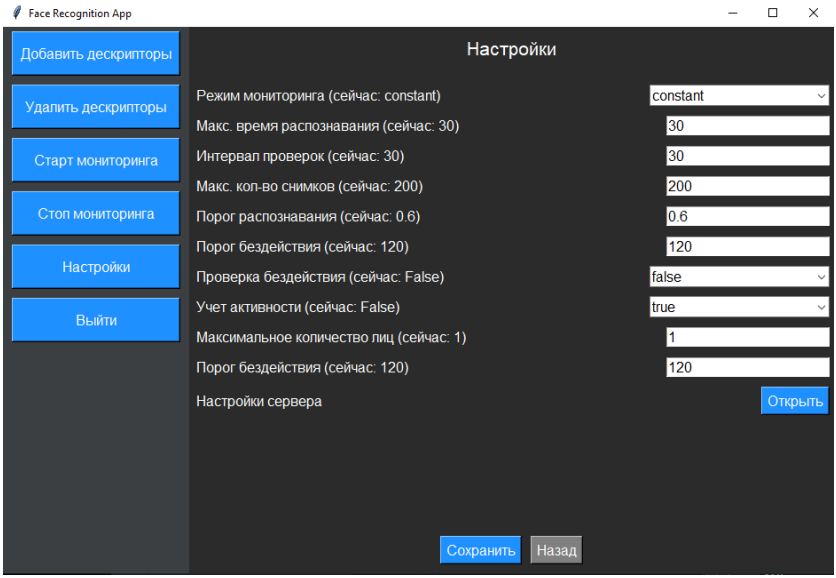


Рис. 3. Окно настройки приложения

Заключение

В результате работы было разработано клиент-серверное приложение для распознавания лиц в режиме реального времени, которое обеспечивает баланс между высоким уровнем безопасности и удобством использования. Архитектура приложения построена по клиент-серверному принципу, что обеспечивает масштабируемость,

безопасность и удобство в эксплуатации. Все данные, хранимые на сервере и передаваемые между клиентом и сервером, находятся в зашифрованном виде для обеспечения их безопасности.

Разработанный графический интерфейс клиента интуитивен и содержит все необходимые элементы управления. Гибкость конфигурации параметров позволяет адаптировать систему под конкретные условия эксплуатации.

Таким образом, реализованное решение представляет собой перспективный подход, который может стать хорошим дополнением биометрической идентификации пользователей к существующим методам.

Список литературы

1. «Лаборатория Касперского» проверила 193 миллиона паролей на устойчивость к взлому: результат неутешительный [Электронный ресурс] : Kaspersky. – Режим доступа: https://www.kaspersky.ru/about/press-releases/laboratoriya-kasperskogo-proverila-193-milliona-parolej-na-ustojchivost-k-vzlomu-rezultat-neuteshitelnyj

2. Фахреева, Д. Р. Информационная безопасность с использованием биометрической аутентификации в цифровой цивилизации / Д. Р. Фахреева, Т. В. Новозенко, Л. Н. Сафина // Мировая культура на переломе эпох и цивилизаций, проблемы глобализации 2.0 : Сборник научных статей. – Санкт-Петербург : Общество с ограниченной ответственностью «Издательство «СПбКО», 2023. – С. 96-101.

3. Никифоров, Д. А. Как устроено распознование лиц / Д. А. Никифоров // XXVI Туполевские чтения (школа молодых ученых) : Материалы Международной молодёжной научной конференции (Казань, 09–10 ноября 2023 г.). – Казань: ИП Сагиев А.Р., 2023. – С. 2510-2513.

4. Каляшов, Е. В. Сравнительный анализ систем распознавания лиц, построенных с использованием блоков стандартных архитектур / Е. В. Каляшов // Информационные технологии и телекоммуникации. – 2020. – Т. 8. – № 3. – С. 94-101.

5. High Quality Face Recognition with Deep Metric Learning [Электронный ресурс] : Dlib C++ Library. – Режим доступа: https://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html

Исследование устойчивости нейронной сети Inception-ResNet-v1 к атакам уклонения

П. С. Лыгина

Студент бакалавр

А. В. Швырева

Старший преподаватель

Введение

В настоящее время нейронные сети используются для решения многих задач. Одной из таких задач является задача распознавания лиц, рассматриваемая в данной работе. Определение личности человека по лицу находит применение в системах аутентификации и подтверждения личности, в поиске правонарушителей и других людей на фото и видео. Это позволяет экономить время на обработку большого объема данных и избавляет от человеческого фактора при опознании человека по внешности. Однако, несмотря на преимущества, такие системы могут подвергаться атакам. Наиболее часто встречающимся видом атак являются атаки уклонения, целью которых является незаметное изменение входных данных таким образом, чтобы вызвать ошибку распознавания. Многие нейронные сети, показывающие высокие результаты на чистых данных, оказываются уязвимыми к атакам подобного рода. Это оказывает влияние на системы аутентификации, где злоумышленник имеет возможность выдать себя за другого человека, изменив свое изображение, и получить несанкционированный доступ к чужим данным. Данная работа посвящена определению влияния атак уклонения на работу нейронной сети распознавания лиц Inception-ResNet-v1 и сравнению методов атак.

1. Атаки уклонения на модели машинного обучения

Атаки уклонения производятся злоумышленниками на те модели машинного обучения, которые успешно прошли обучение на достоверных данных и достигли высокой точности. Задача атаки ставится следующим образом. Есть задача классификации, которая заключается в нахождении отображения $f : X \to Y$, где X – исходный набор данных, а Y – конечное множество меток классов. Тогда

© Лыгина П. С., Швырева А. В., 2025

отображение f уязвимо для атаки тогда, когда существует отображение A такое, что для любого $x \in X$ найдется $\tilde{x} = A(x)$, для которого $f(\tilde{x}) \neq y$ при том, что $f(x) = y$ [1]. Примерная схема атаки для распознавания лиц изображена на рисунке.

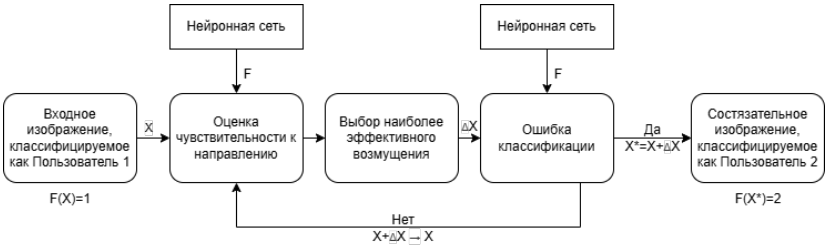


Рис. 1. Схема атаки уклонения на модель распознавания лиц

Рассмотрим методы атак уклонения, которые будут применяться к выбранной модели и набору данных.

FGSM (Fast Gradient Sign Method).

Пусть θ – параметры модели, x – входные данные, y – выходные данные, связанные с x, а $J(\theta, x, y)$ – функция, используемая для обучения нейронной сети. Тогда оптимальное изменение входных данных можно выразить следующим образом [2]

$$\eta = \varepsilon sign(\nabla_x J(\theta, x, y)) \quad (1)$$

Требуемый градиент может быть эффективно вычислен с помощью метода обратного распространения ошибки. В результате входные данные модифицируются следующим образом

$$x_{adv} = x + \eta \quad (2)$$

Basic Iterative Method.

Данный метод является расширением предыдущего быстрого градиентного метода. Он применяется несколько раз с небольшим шагом и значения пикселей промежуточных результатов обрезаются после каждой итерации, чтобы они отличались от исходного изображения не более чем на ε и оставались в пределах допустимого диапазона [3].

Пусть X – входное изображение, X_i^{adv} – измененное изображение на i-ой итерации, y_{true} – истинный класс, $J(X, y)$ – функция ошибки, α – коэффициент, определяющий шаг в направлении градиента функции

ошибки, ε – максимальное отклонение от исходного изображения. Тогда изменение изображения можно представить следующим образом

$$X_0^{adv} = X, \quad X_{N+1}^{adv} = Clip_{X,\varepsilon}\left\{X_N^{adv} + \alpha sign\left(\nabla_X J\left(X_N^{adv}, y_{true}\right)\right)\right\} \tag{3}$$

Здесь

$$Clip_{X,\varepsilon}\{X'\}(x,y,z) = \min\left(255, X(x,y,z) + \varepsilon, \max\left(0, X(x,y,z) - \varepsilon\right.\right. \tag{4}$$

Carlini and Wagner's attack.

Это итеративная атака, которая позволяет получить изображения, противодействующие многим средствам защиты, устойчивым к другим атакам. Использует оптимизатор Adam и специально подобранную функцию потерь для поиска примеров с меньшим искажением, чем при других атаках. За это приходится платить временем, так как эта атака работает намного медленнее, чем другие.

Рассмотрим вариант этой атаки, который минимизирует L_2 расстояние между состязательным примером и исходным изображением. Для конкретного изображения x выбирается целевой класс t, не равный истинному классу, и ищется ω, которое минимизирует следующую функцию [4]

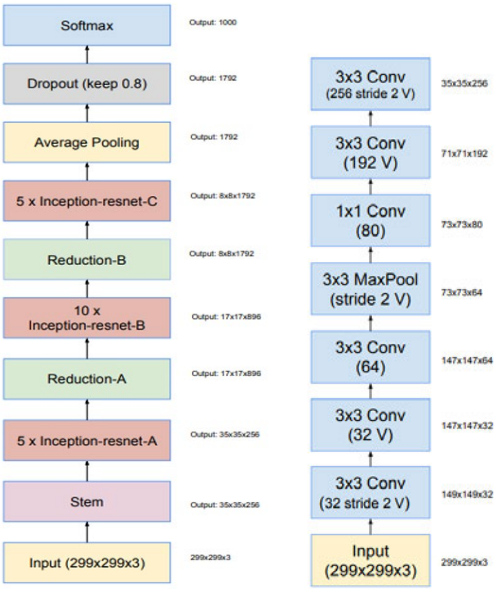
$$\left\|\frac{1}{2}\left(\tanh(\omega)+1\right) - x\right\|_2^2 + cf\left(\frac{1}{2}\left(\tanh(\omega)+1\right)\right) \tag{5}$$

Здесь первое слагаемое – это L_2 расстояние между входными данными и состязательным примером, оно отвечает за то, насколько похожи изображения, второе слагаемое отвечает за то, насколько полученный пример соответствует целевому классу. f – функция, подобранная таким образом, что измененное изображение $x+\delta$ относится к классу t, только когда $f(x+\delta) \le 0$. В данной формуле измененное изображение записано как $\frac{1}{2}\left(\tanh(\omega)+1\right)$. Это позволяет всегда удовлетворять ограничению $0 \le x+\delta \le 1$, чтобы значения пикселей находились в допустимых пределах. Коэффициент c выбирается так, чтобы оба слагаемых имели равный вклад и оба были минимизированы.

2. Архитектура модели и набор данных

Для исследования была выбрана архитектура Inception-ResNet-v1 [5]. Ее основная идея заключается в объединении структуры и

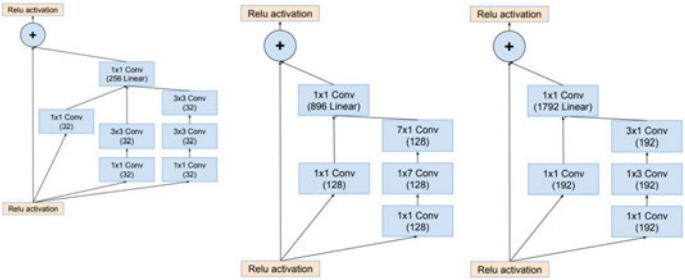
эффективности архитектуры Inception со свойствами регуляризации ResNet. Архитектура модели представлена на рисунках



а б

а – архитектура модели, б – блок Stem

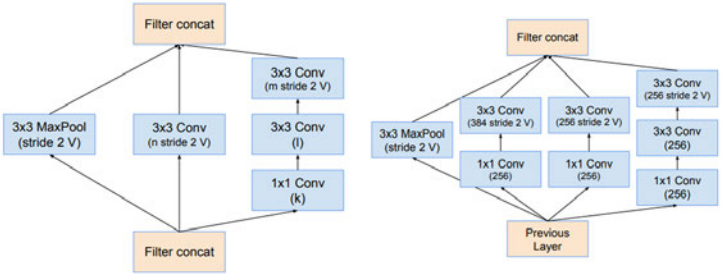
Рис. 2. Архитектура модели Inception-ResNet-v1:



а б в

а – Inception-resnet-A, б – Inception-resnet-B, в – Inception-resnet-C

Рис. 3. Блоки Inception-resnet



а б

а – Reduction-A, б – Reduction-B

Рис. 4. Блоки Reduction

Перед слоем Softmax были добавлены следующие слои для классификации

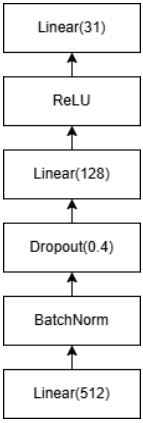


Рис. 5. Слои, добавленные перед Softmax

В качестве набора данных был выбран датасет Face Recognition Dataset, в котором 2562 изображений, 31 класс, от 30 до 110 изображений на один класс. Модель была предобучена на датасете VGGFace2. Слои классификации были дообучены на описанном выше Face Recognition Dataset.

3. Влияние атак на точность модели

Нейросеть можно считать устойчивой к атакам уклонения, если она показывает высокую точность распознавания, несмотря на изменения

изображения. Для исследования устойчивости выбранной сети атаки, описанные в разделе 1, были применены к тестовому набору данных (20% от исходного датасета) с разными значениями параметров. Результаты представлены в таблице

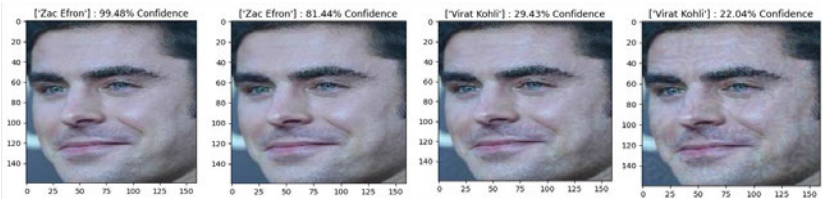
Таблица

Точность распознавания на исходных тестовых данных и на измененных данных

Атака/ Метрика	Accuracy	Precision: micro-averaging	Precision: macro-averaging	Recall: micro-averaging	Recall: macro-averaging
Исходные данные	0.9941	0.9991	0.9913	0.9941	0.9923
FGSM $\varepsilon = 0.01$	0.9375	0.9867	0.9860	0.9375	0.9237
FGSM $\varepsilon = 0.05$	0.4609	0.3356	0.4280	0.4609	0.4428
FGSM $\varepsilon = 0.1$	0.1953	0.1332	0.2175	0.1953	0.1890
BIM $\varepsilon = 0.01$	0.8965	0.9547	0.9587	0.8965	0.8913
BIM $\varepsilon = 0.02$	0.5547	0.4625	0.5434	0.5547	0.5516
BIM $\varepsilon = 0.05$	0.0098	0.0592	0.0857	0.0098	0.0096

Отсюда видно, что при использовании FGSM для существенного падения точности нужно вносить большие изменения в изображения, а BIM позволяет добиться большего падения точности.

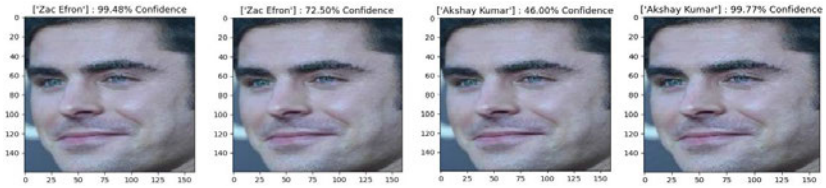
На примере одного изображения можно увидеть вносимые изменения.



а б в г

а – исходное изображение, б – результат FGSM с $\varepsilon = 0.01$, в – результат FGSM с $\varepsilon = 0.05$, г – результат FGSM с $\varepsilon = 0.1$

Рис. 6. Результаты атаки FGSM

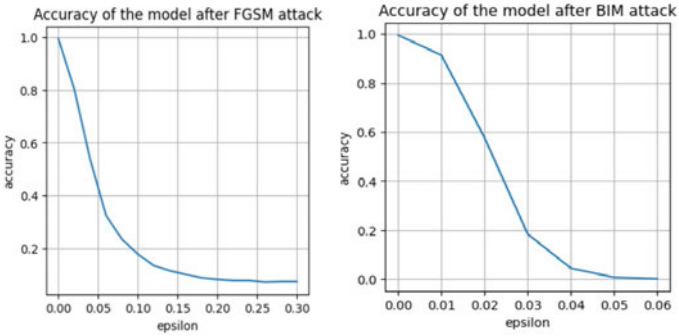


а б в г

а – исходное изображение, б – результат BIM с $\varepsilon = 0.01$*,*
в – результат BIM с $\varepsilon = 0.02$*, г – результат BIM с* $\varepsilon = 0.05$

Рис. 7. Результаты атаки BIM

На следующих графиках видно, что для обоих методов при маленьких значениях параметра точность падает быстрее. Значит, чаще всего не требуется вносить слишком большие изменения для получения желаемого результата.

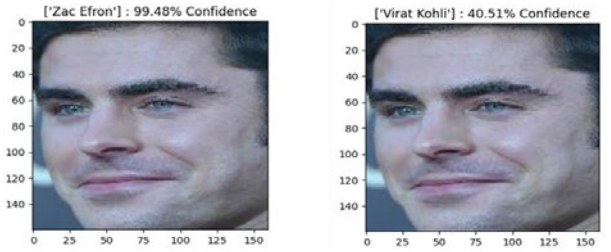


а б

а – для FGSM, б – для BIM

Рис. 8. Зависимость точности модели от параметра ε

Для Carlini and Wagner’s attack, в силу большого времени работы, было получено одно изображение, практически неотличимое от исходного.



а б

а – исходное изображение, б – результат атаки

Рис. 9. Результат Carlini and Wagner’s attack

Заключение

В данной статье были рассмотрены три вида атак уклонения на нейронные сети и их влияние на точность распознавания модели Inception-ResNet-v1. Данная сеть не является устойчивой к атакам, так как даже небольшие изменения существенно снижают ее точность. При сравнении методов атак выявлено, что метод FGSM работает быстрее всего и часто оказывается достаточным для получения ошибки распознавания, однако изменения, полученные при помощи данного метода, легко заметны человеческому глазу. Basic Iterative Method позволяет получить изображение, более близкое к исходному, с большой ошибкой классификации. Carlini and Wagner’s attack с большей вероятностью реализует успешную атаку, однако работает большое количество времени.

Список литературы

1. Намиот, Д. Е. Атаки на системы машинного обучения - общие проблемы и методы / Д. Е. Намиот, Е. А. Ильюшин, И. В. Чижов // International journal of open information technologies. – 2022. – Т. 10 – № 3. – С. 17-22.

2. Goodfellow, I. J. Explaining and harnessing adversarial examples / I. J. Goodfellow, J. Shlens, C. Szegedy. – 2015. – 11 p.

3. Kurakin, A. Adversarial examples in the physical world / A. Kurakin, I. J. Goodfellow, S. Bengio; Workshop track – ICLR, 2017. – 14 p.

4. Carlini, N. Towards Evaluating the Robustness of Neural Networks / N. Carlini, D. Wagner. – 2017. – 19 p.

5. Szegedy, C. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning / C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi. – 2016. – 12 p.

Сравнение производительности реализаций языка Python в задачах ресурсоёмких вычислений

С. Э. Масыч

Студент бакалавр

С. В. Борзунов

Доцент

Введение

Язык программирования Python завоевал широкое признание благодаря простоте синтаксиса и обширной базе библиотек, став стандартом в области анализа данных и машинного обучения. Тем не менее, его основная реализация – CPython – ограничена по производительности из-за отсутствия JIT-компиляции и наличия глобальной блокировки интерпретатора (GIL). Для преодоления этих ограничений созданы альтернативные реализации, такие как PyPy, IronPython, Jython и др. [1, 2].

Современные методы математического моделирования и анализа больших данных, как правило, базируются на алгоритмах, требующих значительных вычислительных ресурсов. Развитие высокопроизводительных вычислений позволило реализовать сложные алгоритмы за счёт использования параллельных и распределённых архитектур. Однако эффективность таких алгоритмов определяется не только выбором аппаратной платформы, но также и соответствующими программными средствами. Особенно ярко это проявляется в реализации алгоритмов с использованием языка Python, который, несмотря на широкую популярность, изначально не был ориентирован на ресурсоёмкие вычисления.

1. Методика эксперимента

Для оценки производительности различных реализаций языка Python была разработана методика, основанная на воспроизводимых вычислительных экспериментах. В качестве тестовых задач были выбраны два класса вычислений, предназначенных для научных и инженерных расчётов: операция умножения квадратных матриц и задача табулирования функции.

© Масыч С. Э., Борзунов С. В., 2025

Эти задачи являются типичными для прикладной линейной алгебры и численного анализа соответственно и хорошо поддаются масштабированию.

Алгоритм умножения матриц реализован с использованием трёх вложенных циклов, что приводит к кубической вычислительной сложности. Размер матриц варьировался от 1000×1000 до 2000×2000, что позволило выявить закономерности при увеличении объёма вычислений. Второй класс задач предполагал табулирование функции вида

$$f(x) = \sin(x) \cdot J_0(x), \tag{1}$$

где $J_0(x)$ — функция Бесселя нулевого порядка. Расчёт проводился на равномерной сетке от 0 до 10^4 с шагом 10^{-4}, что эквивалентно 10^8 операций вычисления функции в одной точке.

Для сравнения были выбраны следующие реализации языка Python:

- **CPython** (стандартная реализация),
- **PyPy** (использующая JIT-компиляцию),
- **IronPython** (работающая на платформе .NET),
- **Jython** (интегрированная с JVM).

Для сравнения использовалась реализация алгоритмов на языке **C**, распараллеливаемая с помощью директив технологии параллельного програмирования **OpenMP**. Время исполнения измерялось следующим образом: в C использовалась функция `omp_get_wtime()`, в Python — `time.perf_counter()`.

Тестирование проводилось на двух аппаратных платформах:

– персональный компьютер с процессором AMD Ryzen 5 2600 (12 потоков), ОС Windows 10;

– сервер на базе двух процессоров Intel Xeon E5-2620v3 (12 потоков), ОС CentOS 7.

Каждый эксперимент повторялся по 5 раз, результаты усреднялись. Измерения проводились для различного количества потоков от 1 до 12 с целью оценки масштабируемости.

Ключевым показателем эффективности многопоточного исполнения служит ускорение $S_p(N)$, рассчитываемое по формуле:

$$S_p(N) = \frac{T_1(N)}{T_p(N)}, \tag{2}$$

где $T_1(N)$ — время выполнения задачи в однопоточном режиме, $T_p(N)$ — время при использовании p потоков. Этот показатель позволяет

количественно оценить производительность при параллельной обработке и сопоставить масштабируемость различных реализаций [3–5].

Такая постановка эксперимента позволила обеспечить сопоставимость результатов и получить объективную картину эффективности каждой реализации в условиях, близких к реальной вычислительной практике.

2. Результаты и анализ

Задача умножения матриц: увеличение размера матриц (от 1000 до 2000) позволило выявить устойчивость и эффективность масштабируемости. При переходе от одного потока к двенадцати ускорение составило:

- **C + OpenMP**: с 1× до 6.5–7.2×;
- **CPython**: с 1× до ~5×;
- **IronPython**: с 1× до ~5.2×;
- **PyPy**: с 1× до ~3.1–3.4×;
- **Jython**: с 1× до ~3.8–4×.

На рис. 1 и 2 наблюдается рост ускорения у **IronPython** и **CPython**, тогда как **PyPy** быстро выходит на плато. **C + OpenMP** демонстрирует близкий к линейному рост за счёт эффективного распределения нагрузки по потокам.

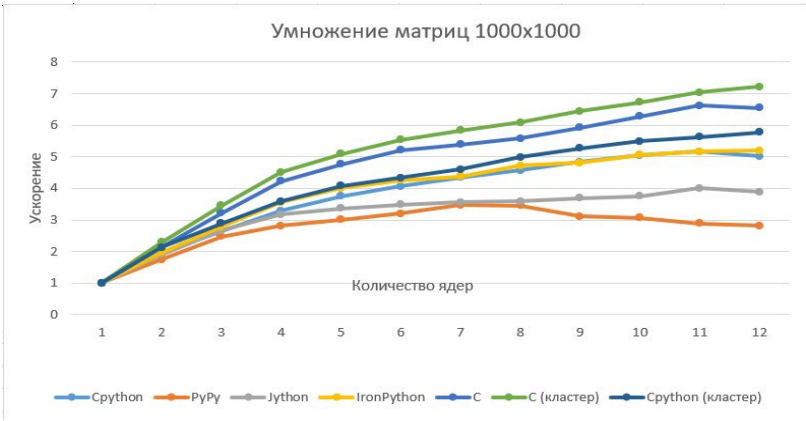


Рис. 1. Умножение матриц 1000x1000

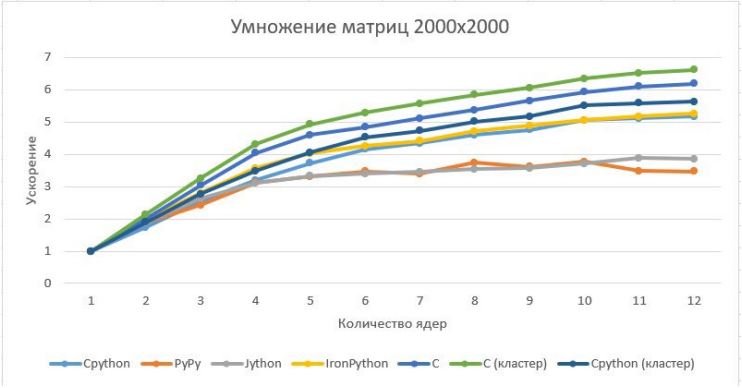


Рис. 2. Умножение 2000x2000

Задача табулирования функции: при табулировании специальной функции ускорение показало аналогичную тенденцию. В частности:

- **COpenMP** достигает ~4.4×;
- **IronPython** даёт ускорение до 5× благодаря отсутствию GIL;
- **CPython** – до ~4.5×,
- **PyPy** – около 2.7×;
- **Jython** – не более 3.8×.

На рис. 3 видно, что именно **IronPython** демонстрирует наилучшее поведение среди Python решений, особенно при увеличении числа потоков, в то время как **PyPy** значительно теряет эффективность.

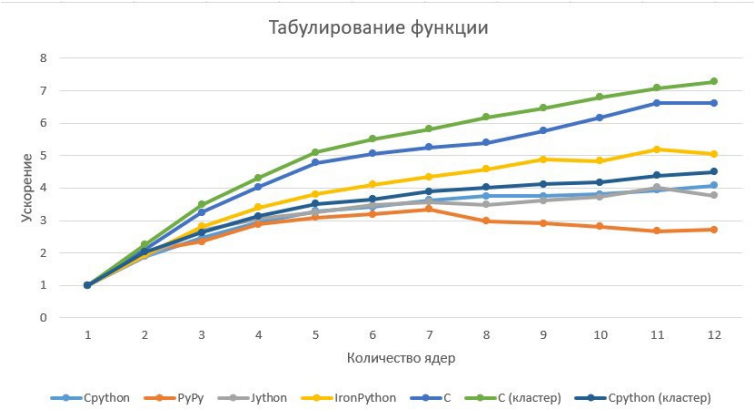


Рис. 3. Задача табулирования функции

Выполнение вычислений на суперкомпьютере ВГУ показало ускорение на 6-10% выше, чем на настольном персональном компьютере. Это связано с более эффективным распределением ресурсов и большим объёмом оперативной памяти.

Заключение

Проведённое исследование показало, что эффективность реализации Python в задачах высокопроизводительных вычислений существенно зависит от архитектуры интерпретатора и применяемых механизмов параллелизма. PyPy обеспечивает наилучшую производительность в однопоточном режиме за счёт JIT-компиляции, но сильно ограничен в масштабируемости. CPython с использованием `multiprocessing` демонстрирует хорошую масштабируемость и универсальность. IronPython, из-за интеграции с .NET, оказался эффективным в многопоточных сценариях. Jython существенно уступает по скорости и, таким образом, не рекомендуется для задач высокопроизводительных вычислений. Реализация на C + OpenMP по-прежнему обеспечивает наилучшие результаты, но требует более высокой квалификации и времени на разработку.

Таким образом, выбор конкретной реализации должен основываться на типе задачи, требуемом уровне масштабируемости и доступной вычислительной инфраструктуре.

Список литературы

1. Zaccone, G. Python Parallel Programming Cookbook. – Packt Publishing, 2015. – 286 p. – ISBN 978-1785883758.

2. Gorelick, M. High Performance Python: Practical Performant Programming for Humans / M. Gorelick, I. Ozsvald. – O'Reilly Media, 2020. – 500 p. – ISBN 978-1492055020.

3. Борзунов, С. В. Суперкомпьютерные вычисления: практический подход / С. В. Борзунов, С. Д. Кургалин. – Санкт-Петербург : БХВ-Петербург, 2019. – 256 с. – ISBN 978-5-9909805-2-5.

4. Биллиг, В. А. Параллельные вычисления и многопоточное программирование : учебник / В. А. Биллиг. – Москва : ИНТУИТ, 2016. – 310 с. – ISBN 978-5-4497-0936-3.

5. Гергель, В. П. Высокопроизводительные вычисления для многопроцессорных многоядерных систем / В. П. Гергель. – Москва : Изд-во МГУ, 2010. – 544 с. – ISBN 978-5-211-05937-5.

Реактивная реализация паттерна сага в условиях легаси архитектуры

М. С. Медведев

Студент бакалавр

Е. Д. Проскуряков

Ассистент

Введение

Микросервисная архитектура в современных системах сталкивается с проблемой обеспечения транзакционной целостности данных при увеличении числа взаимодействующих сервисов. Задача обеспечения межсервисной транзакционности особенно актуальна для систем, в которых изначально эта функциональность не была запланирована, но с ростом сложности стала необходима для предотвращения рассогласованности данных. В частности, возникла необходимость в механизме обеспечения консистентности данных в рамках развития крупной, устоявшейся системы, где обработка одного запроса пользователя может включать в себя взаимодействие с S3 хранилищем, распределёнными блокировками, реляционными и нереляционными базами данных, брокером сообщений и сторонними системами через REST API.

Существуют различные подходы к решению проблемы обеспечения межсервисной транзакционности, например, двухфазная фиксация и паттерн сага. Однако реализация этих паттернов может оказаться сложной задачей, особенно в условиях существующей легаси-архитектуры. Интеграция новых транзакционных механизмов в устоявшуюся систему требует значительных усилий по рефакторингу и адаптации.

В этой работе рассматриваются паттерны для обеспечения межсервисной транзакционности, а также создание и внедрение решения для обеспечения консистентности данных.

1. Паттерны для обеспечения межсервисной транзакционности

Рассмотрим паттерн двухфазной фиксации. Данный паттерн предполагает два этапа: подготовку и фиксацию. На этапе подготовки все микросервисы, необходимые для выполнения транзакции, сообщают

© Медведев М. С., Проскуряков Е. Д., 2025

координатору о готовности завершить транзакцию. На этапе фиксации микросервисы выполняют требуемые от них действия [1-2].

Пример выполнения двухфазной фиксации представлен на рис. 1.

Рис. 1. Пример выполнения двухфазной фиксации

Преимуществами данного подхода являются:

– атомарность транзакции. Все микросервисы сработают успешно или не внесут никаких изменений;

– изолированное чтение от записи. Изменения в объектах не видны пока координатор транзакций не зафиксирует эти изменения.

Недостатками данного подхода являются:

– координатор транзакций может значительно замедлять работу системы;

– координатор транзакций является единой точкой отказа.

Рассмотрим паттерн сага. Паттерн сага является шаблоном проектирования микросервисов, обеспечивающий согласованность данных в распределённых системах. Существуют два подхода к реализации паттерна сага: хореография и оркестрация [3].

В подходе с хореографией каждый микросервис, участвующий в распределенной транзакции, ответственен за управление сагой, и нет необходимости в центральном координаторе.

Рассмотрим пример на рис. 2. Микросервисы с помощью брокера сообщений передают друг другу обработку запроса. В случае неудачи, например при возникновении ошибки в одном из микросервисов, он отправит другое сообщение (или в другую очередь), которое прочитает предыдущий микросервис и компенсирует внесенные изменения.

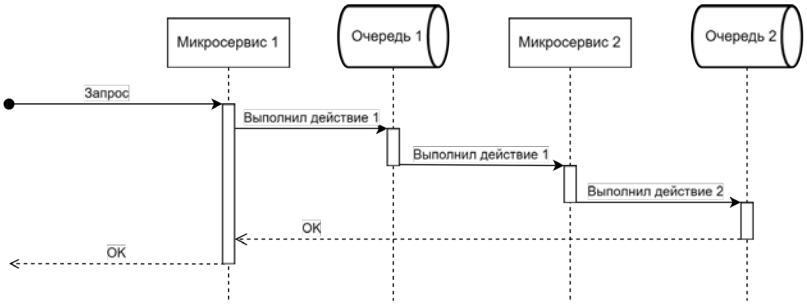


Рис. 2. Пример обработки запроса для хореографии

Преимуществами данного подхода являются:

– отсутствие единой точки отказа. Каждый микросервис сосредотачивается на собственной атомарной транзакции;

– работа микросервисов не блокируется. При помощи такого подхода можно обеспечить масштабируемость системы.

Недостатками данного подхода являются:

– не обеспечивается изоляция при чтении, изменения в микросервисах видны до завершения всей распределённой транзакции;

– при большом количестве связей между микросервисами усложняется их отладка и поддержка.

В подходе с оркестрацией существует центральный координатор. Сервис оркестрации координирует процесс выполнения межсервисной транзакции, а в случае ошибки запускает компенсационные операции для предыдущих шагов [3]. Сервис оркестрации, например, может быть реализован в Camunda процессе. В примере на рис. 3. показан процесс, состоящий из двух действий, а в случае ошибки запускаются компенсации.

Преимуществами данного подхода являются:

– централизованное управление. При помощи такого подхода можно легче организовывать сложную бизнес-логику;

– снижение связанности между сервисами. Изменения в порядок выполнения шагов или в логику саги требуется вносить только в оркестраторе, не затрагивая сами микросервисы, которые в свою очередь могут ничего не знать друг о друге.

Недостатками данного подхода являются:

– не обеспечивает изоляцию;

– оркестратор является единой точкой отказа.

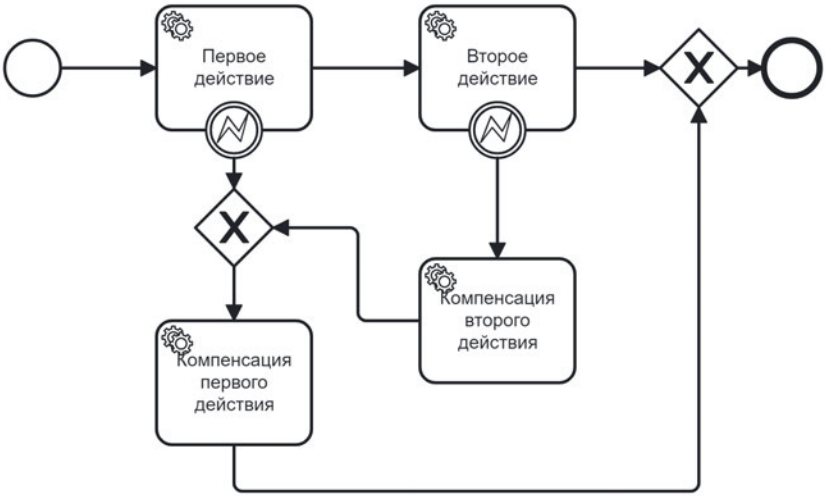


Рис. 3. Пример обработки запроса центральным координатором

2. Реализация и внедрение собственного решения

На данный момент существует несколько фреймворков для обеспечения согласованности данных в микросервисной архитектуре. Например, фреймворки: Axon, Apache Saga Actuator, Eventuate Tram.

Существует несколько проблем внедрения готового решения:

– много существующих связей между микросервисами;

– фреймворки требуют много изменений, например понадобиться сильно декомпозировать легаси код, чтобы бизнес-логика могла быть обёрнута в аннотации фреймворков или реализовывала их интерфейсы;

– отсутствие документации на проекте. Любое изменение кода для соответствия фреймворкам может быть тяжело протестировать, поскольку потеряны требования к функциональности приложения;

– ответственных людей, проектировавших или реализовывавших некоторую функциональность, уже может не быть на проекте, а внесение изменений в большое количество легаси-кода станет трудной задачей для новых разработчиков.

Учитывая описанные ограничения, возникла необходимость в разработке собственного решения для обеспечения межсервисной транзакционности. Это решение должно было позволить внедрять логику саги с минимальным воздействием на существующий код, обеспечивая при этом консистентность данных в распределенной системе. Процесс внедрения включал в себя несколько этапов: анализ существующих бизнес-процессов, выявление ключевых точек

взаимодействия между сервисами, создание функций компенсации, разработку и внедрение сервиса координатора, а также настройку существующих сервисов для взаимодействия с ним. Важно отметить, что внедрение производилось постепенно, с минимальным влиянием на существующую функциональность. В качестве средств реализации используются язык программирования Java с фреймворком Spring Boot и нереляционная СУБД Redis.

Создан отдельный сервис, называемый координатором, который предоставляет клиентам определённое API для взаимодействия с ним. Для взаимодействия с координатором с помощью Java реализован отдельный модуль, который можно подключить как библиотеку.

Основные конечные точки API координатора это:

– начало межсервисной транзакции;
– добавление шага в межсервисную транзакцию;
– завершение распределённой транзакции.

Микросервис, участвующий в межсервисной транзакции, перед изменением данных сообщает об этом сервису координатора, предоставляя ему некоторые данные, необходимые для компенсации изменений, а также способ запуска компенсации. Вместо декомпозиции легаси-кода, добавляется вызов в координатор перед выполнением бизнес-логики, а после всех требуемых действий делается вызов в координатор об успешном или не успешном завершении транзакции.

Координатор хранит данные о текущих распределённых транзакциях в СУБД Redis. Каждая отдельная транзакция хранит в себе данные о шагах в ней, шаг состоит из следующих атрибутов:

– номер шага в межсервисной транзакции;
– ожидаемая дата, успешного завершения;
– URL для вызова компенсирующей операции;
– данные, необходимые для компенсации;
– статус шага (в процессе, выполнен, компенсирован).

Связь между шагами и транзакцией показана на рис. 4.

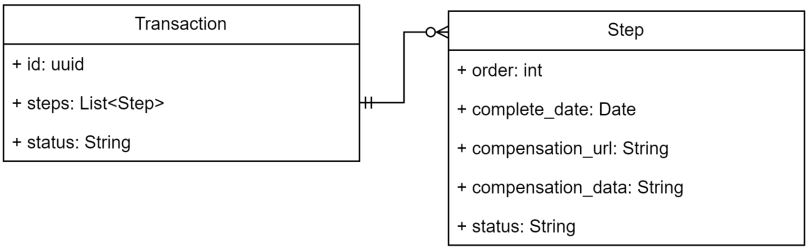


Рис. 4. Сущности координатора

Взаимодействие микросервисов с координатором, в случае если в процессе обработки запроса произошла ошибка, показано на рис. 5. Микросервисы берут на себя ответственность за:

– уведомление координатора о некотором действии;
– уведомление координатора об ошибке;
– предоставление конечной точки для вызова компенсирующей операции;
– уведомление о завершении межсервисной транзакции.

Вызовы к координатору, кроме вызова для создания межсервисной транзакции выполняются асинхронно, с помощью реактивного Java клиента WebClient. Сам координатор обрабатывает запросы в реактивном режиме и реализован с использованием Spring WebFlux.

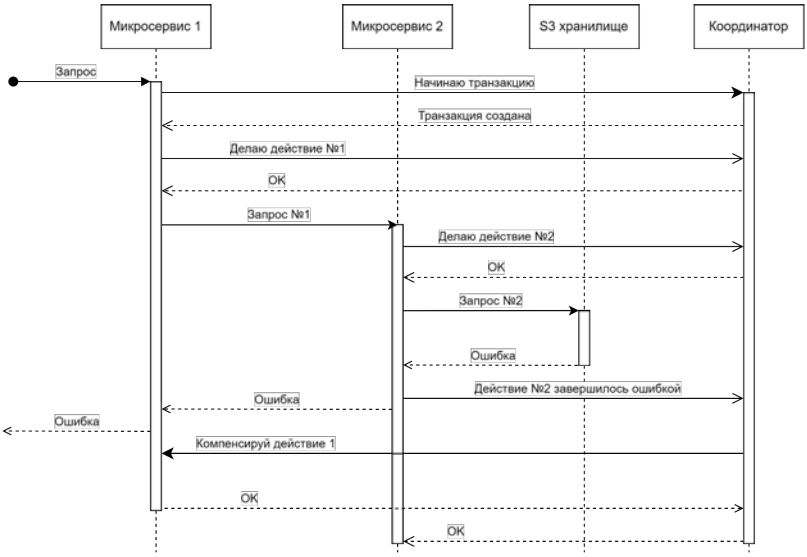


Рис. 5. Взаимодействие микросервисов с сервисом координатора

3. Масштабируемость и отказоустойчивость созданного решения

Координатор хранит состояние транзакции в БД, а значит не потеряет эти данные в случае перезапуска.

В случае, если координатор не смог компенсировать некоторую транзакцию, может быть вызвана конечная точка для поиска и компенсации незавершенных шагов с истекшей датой ожидаемого успешного завершения. Вызов этой точки может выполняться любым

способом. В конкретной реализации вызов происходит автоматически по расписанию, для этого используется CronJob на уровне kubernetes кластера.

Таким образом, координатор может быть развернут в нескольких репликах, поскольку хранит состояние сущностей в БД, а задание по расписанию вынесено из самого сервиса. Используя кластерное и шардированное решение Redis, работа с БД не становится узким местом этой системы, благодаря высокой производительности такого варианта развертывания Redis [4].

4. Тестирование системы с учётом отказов

Производилось хаос тестирование для проверки устойчивости программной системы к неожиданным ситуациям или сбоям [5]. Для этого отключались отдельные компоненты системы.

По внутренним метрикам компании, для которой было реализовано описанное решение, количество жалоб пользователей на рассогласованность данных снизилось на 90%.

Заключение

В результате разработки создано и внедрено решение, обеспечивающее согласованность данных между микросервисами, для использования которого не требуется декомпозиция существующего кода бизнес-логики. Описано взаимодействие микросервисов с сервисом, контролирующим выполнение межсервисной транзакции. В дальнейшем предполагается изменение созданного сервиса путём добавления API на основе очередей в целях повышения отказоустойчивости.

Список литературы

1. Kleppmann, M. Designing Data-Intensive Applications / M. Kleppmann. – Sebastopol, California : O'Reilly Media, 2017. – 590 p.

2. Distributed Systems: Concepts and Design / G. Coulouris [et al.] ; exec. editor M. Goldstein. – 5th edition – Boston : Addison-Wesley, 2004. – 1049 p.

3. Richardson, C. Microservices Patterns / C. Richardson. – Shelter Island : Manning Publications, 2019. – 491 p.

4. Nelson, J. Mastering Redis / J. Nelson. – Birmingham : Packt, 2016. – 366 p.

5. Miles, R. Learning Chaos Engineering / R. Miles. – Sebastopol, California : O'Reilly Media, 2019. – 160 p.

Разработка программного комплекса для контроля защищенности внешнего периметра информационной системы

А. Е. Меркулов

Студент бакалавр

А. Ю. Иванков

Доцент

Введение

Вопросы обеспечения кибербезопасности в условиях стремительной цифровизации организаций, переход на отечественное программное обеспечение приобретают критическое значение.

Бизнесу важно видеть полную картину организации: потеря конфиденциальных данных, какие цифровые активы уязвимы, где могут появиться фишинговые страницы и т.д.

Для обеспечения безопасности информации в текущих реалиях недостаточно рассчитывать на антивирусные программы в борьбе с современным вредоносным программным обеспечением. Уязвимые службы и системы могут стать легкой добычей для злоумышленников, что приведет к непоправимым последствиям для репутации организации.

Целью этой работы является разработка программного комплекса для мониторинга и анализа защищенности внешнего периметра информационной системы, обеспечивающего автоматизированное сканирование, выявление уязвимостей и формирование отчетов для повышения уровня безопасности информационной системы.

По данным исследования эксплуатация уязвимостей стала одним из основных методов атаки на организации в 2023 году [1]. Для минимизации риска эксплуатации уязвимостей необходим комплексный подход к защите:

– Выстраивание процесса оперативного устранения уязвимостей;

– Проведение регулярной инвентаризации и категоризации активов;

– Систематическую оценку уровня защищенности ИТ-инфраструктуры и приложений.

© Меркулов А. Е., Иванков А. Ю., 2025

1. Банк данных угроз безопасности информации ФСТЭК России

Согласно Указу Президента РФ от 1 мая 2022 г. N 250 "О дополнительных мерах по обеспечению информационной безопасности Российской Федерации" с 1 января 2025 года запрещено использовать средства защиты информации, произведенные в «недружественных» странах или компаниями под их юрисдикцией.

В данной работе используется база знаний уязвимостей из БДУ ФСТЭК России. На сегодняшний день в банке описано более тридцать тысяч уязвимостей.

Банк данных угроз ФСТЭК России – это электронная база, содержащая информацию о выявленных угрозах, их классификации, методах эксплуатации и мерах нейтрализации. Используется в задачах специалистов по информационной безопасности для обеспечения актуальными данными для анализа рисков и проектирования защитных мер.

2. Требования к разрабатываемому комплексу

При разработке программного комплекса учитывались следующие требования:

- Обнаружение открытых портов;
- Определение сетевых протоколов;
- Определение программного обеспечения;
- Поиск системных уязвимостей;
- Поиск поддоменов;
- Интеграция с БДУ ФСТЭК;
- Непрерывное исследование защищенности всех активов;
- Использование микросервисной архитектуры и технологии контейнеризации;
- Создание открытого API для возможного подключения других сканирующих приложений.

3. Реализация микросервисов

Для реализации программного комплекса была выбрана микросервисная архитектура. Она организует комплекс как набор небольших независимых сервисов, каждый из которых решает конкретную бизнес-задачу и функционирует в собственной среде для обеспечения гибкости и масштабируемости.

Для контейнеризации приложений используется Docker, который упаковывает код, зависимости и настройки в изолированные среды - их называют Docker Images.

Взаимодействие между микросервисами строится по принципу REST API посредством обмена HTTP-запросов. Общая структура системы представлена на рис. 1.

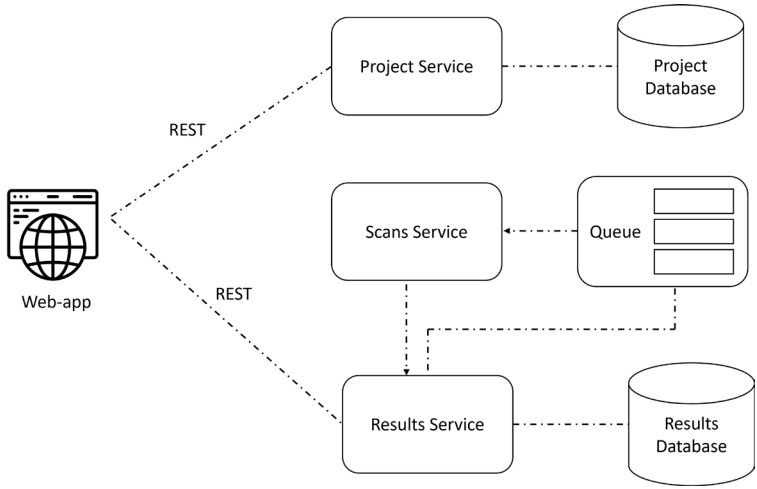


Рис. 1. Структура системы

Серверная часть реализована на языке программирования Python. Используемый фреймворк FastAPI в сочетании с Python становится оптимальным выбором для микросервисной архитектуры благодаря ряду преимуществ: гибкость, скорость разработки и эффективность.

Для простоты реализации была выбрана система управления базами данных PostgreSQL. Она обеспечивает надежное хранение результатов сканирования: поддоменов, открытых портов, версий ПО. Его выбор обусловлен поддержкой JSONB для неструктурированных данных. В дальнейшем возможно использования NoSQL решений в одном из приложений для гибкого способа организации и хранения данных.

В роли брокера сообщений используется RabbitMQ. Он основан на протоколе AMQP. Брокер был выбран как легкая и быстро настраиваемая система для разработки программного комплекса.

Для web-приложения выбран фреймворк Flask, как удобное средство для интеграции с REST API серверной части, удобство рендеринга статических страниц с помощью Jinja2. Упоминания о реализации этого приложения в статье нет.

Стоит упомянуть специфику реализации каждого приложения:

– Project Service;

– Results Service;
– Scans Service.

Задача Project Service - создание и агрегация проектов для последующего сканирования, привязка доменов сканирования к проекту. ER-диаграмма представлена на рис. 2.

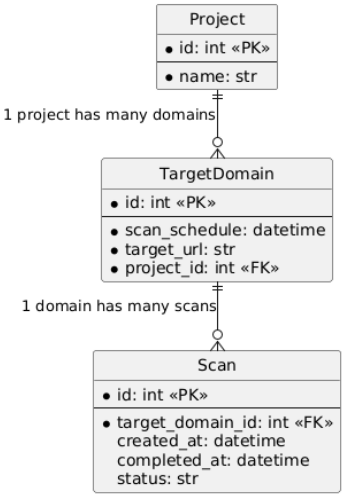


Рис. 2. ER-диаграмма базы данных Project Service

API сервиса организован вокруг двух ключевых сущностей: управления проектами сканирования (project) и обработки задач сканирования доменов (scan). Все эндпоинты соответствуют RESTful-принципам, используют HTTP-методы (POST, GET, PUT) для операций CRUD и возвращают данные в формате JSON. Документация автоматически генерируется через OpenAPI (Swagger UI), что обеспечивает прозрачность и удобство тестирования. Представленную Swagger документацию можно увидеть на рис. 3.

Results Service представляет из себя приложение, которое хранит и создает сканирования результатов внешнего периметра. ER-диаграмма выглядит следующим образом см. рис. 4. Используется основная сущность в базе данных ScanResult, в которую записываются все найденные результаты сканирования в независимости от типа сканера. Для некоторых сканеров были созданы дополнительные отношения для удобства хранения данных, об используемых сканерах далее пойдет речь.

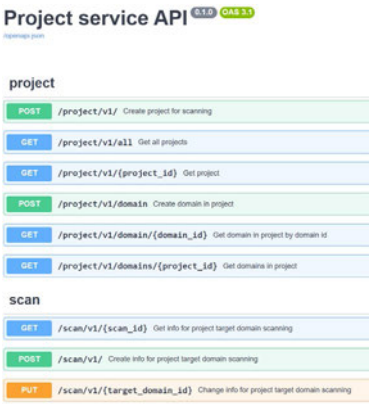


Рис. 3. Swagger-документация для Project Service

API предоставляет эндпоинты для сохранения и получения результатов сканирования сетевых ресурсов. Представленную Swagger документацию можно увидеть на рис. 5.

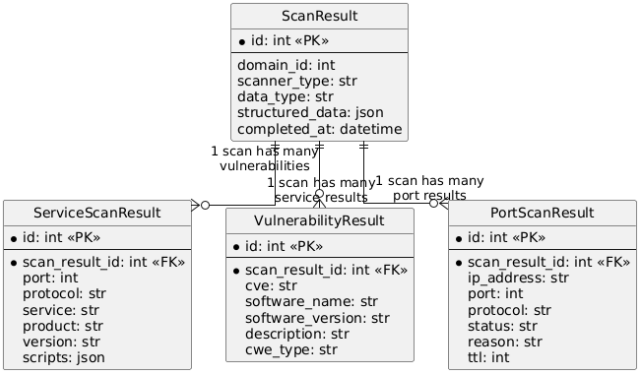


Рис. 4. ER-диаграмма базы данных Results Service

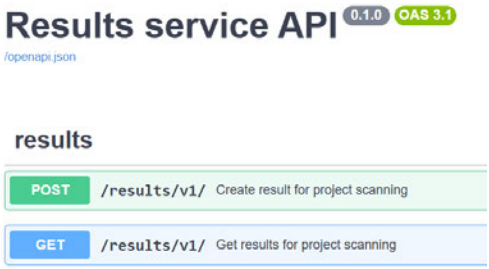


Рис. 5. Swagger-документация для Results Service

Scans Service представляет из себя приложение, которое непосредственно включает в себе различные сканеры уязвимостей:

– Sublist3r для поиска поддоменов через публичные источники;
– Masscan для поиск открытых портов;
– Nmap для определение открытых сервисов и ПО.

Общая логика работы всех сканеров уязвимостей выглядит следующим образом см. рис. 6. В ходе работы алгоритма на основе предыдущих данных сканирования находятся данные из БДУ ФСТЭК России.

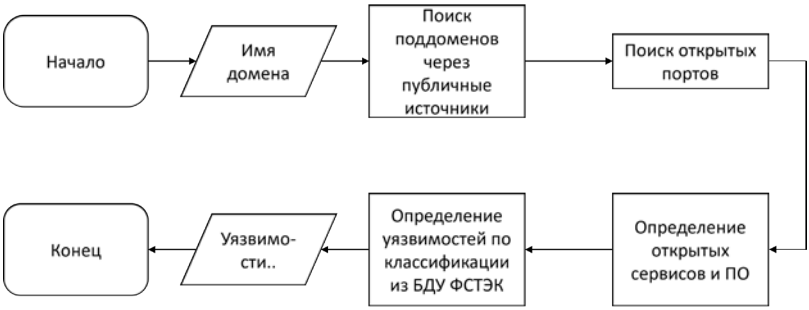


Рис. 6. Общая структура сканирования

4. Тестирование разработанного программного комплекса

Предварительное тестирование разработанного программного комплекса производилось с использованием реального IP адреса 188.246.226.191, в котором работают некоторые службы. Часть результатов представлена в таблицах ниже.

Таблица 1

Открытые порты на IP адресе

№ п/п	1	2	3	4
Порт	443	21	22	80
Протокол	tcp	tcp	tcp	tcp
Статус	open	open	open	open
TTL	52	63	52	52

Интерес в тестировании представляет столбец под номером 4 с портом 80 для того, чтобы пройти всю цепочку алгоритма из рис. 6.

В нем открытые порты выглядят следующим образом см. таблица 2. А уязвимости служб найденные в разделе уязвимости из БДУ ФСТЭК России показаны в таблице 3.

Таблица 2

Работающие сетевые службы на 80 порту

№ п/п	**Порт**	**Протокол**	**Сервис**	**Продукт**	**Версия**
1	80	tcp	http	nginx	1.18.0

Таблица 3

Уязвимости служб, работающие на 80 порту по БДУ ФСТЭК России

№ п/п	1	2	3
Идентификатор в БДУ ФСТЭК России	BDU: 2019-02957	BDU:2019-02994	…
Идентификатор CVE	CVE-2019-9518	CVE-2019-9511	…
Название ПО	nginx	nginx	…
Тип ошибки CWE	CWE-400	CWE-400	…

Заключение

Ключевое преимущество разработанного программного комплекса для мониторинга и анализа защищенности внешнего периметра — автоматизация процессов сканирования, выявления уязвимостей и формирования отчетов, что позволяет минимизировать человеческий фактор и ускорить реагирование на угрозы. Микросервисная архитектура обеспечивает легкость масштабирования системы, а

универсальное API позволяет гибко интегрировать службе безопасности новые сканеры и инструменты анализа, адаптируя комплекс под меняющиеся требования инфраструктуры.

Внедрение подобного решения не только снижает риски эксплуатации слабых мест во внешнем периметре информационной системы, но и способствует сохранению репутации организации за счет предотвращения инцидентов.

Таким образом, предложенный комплекс становится элементом стратегии устойчивого развития бизнеса в цифровом киберпространстве.

Список литературы

1. Актуальные киберугрозы для организаций: итоги 2023 года [Электронный ресурс] / PT Security. — 2023. — Режим доступа: https://ptsecurity.com/ru-ru/research/analytics/aktualnye-kiberugrozy-dlya-organizacij-itogi-2023-goda/ (дата обращения: 16.05.2025).

2. Указ Президента РФ от 1 мая 2022 г. N 250 "О дополнительных мерах по обеспечению информационной безопасности Российской Федерации" (с изменениями и дополнениями) [Электронный ресурс] : Указ Президента РФ от 1 мая 2022 г. N 250 "О дополнительных мерах по обеспечению информационной безопасности Российской Федерации" // СПС «Гарант». — Режим доступа: https://base.garant.ru/404561984/ (дата обращения: 16.05.2025).

3. База данных угроз безопасности информации (БДУ ФСТЭК России) [Электронный ресурс] / ФСТЭК России. – Режим доступа: https://bdu.fstec.ru/ (дата обращения: 16.05.2025).

4. Руководство по веб-фреймворку FastAPI [Электронный ресурс] / Metanit.com. – Режим доступа: https://metanit.com/python/fastapi/ (дата обращения: 16.05.2025).

5. Дурденко, В. А. Разработка классификации и архитектуры построения интегрированных систем безопасности / В. А. Дурденко, А. А. Рогожин // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2013. – № 1. – С. 61-70. – EDN QZBXXL.

Разработка гибридного нейроинтерфейса с имитацией P300-реакции

А. Ю. Минакова

Студент бакалавр

А. Ю. Минакова

Студент бакалавр

Я. А. Туровский

Доцент

Введение

Современное развитие нейротехнологий всё активнее направлено на создание эффективных интерфейсов взаимодействия человека и машины, основанных на анализе мозговой активности. Такие интерфейсы находят широкое применение в медицине, нейрореабилитации, системах вспомогательного управления, а также в виртуальной и дополненной реальности. Одним из наиболее изученных и применимых в задачах BCI компонентов электроэнцефалографической (ЭЭГ) активности является вызванный потенциал P300, возникающий в ответ на значимый для пользователя стимул. Его высокая специфичность и сравнительная устойчивость делают P300 надёжной основой для создания систем выбора и управления [1].

Однако однокомпонентные интерфейсы, опирающиеся исключительно на ЭЭГ-сигналы, сталкиваются с рядом ограничений: высокой чувствительностью к шумам, вариативностью индивидуальных реакций и снижением точности при низкой концентрации внимания. Для преодоления этих проблем всё большую популярность приобретают гибридные нейроинтерфейсы. Одним из перспективных подходов является объединение регистрации P300 и пространственного анализа внимания пользователя через положение головы, что позволяет более точно интерпретировать намерения и фокус интереса [2].

Цель настоящей работы — разработка и тестирование модели гибридного нейроинтерфейса, сочетающего регистрацию P300 и контроль направления взгляда посредством анализа положения головы. В качестве первого этапа реализована техническая модель,

© Минакова А. Ю., Минакова А. Ю., Туровский Я. А., 2025

имитирующая мозговые ответы на визуальные стимулы без участия человека. Такая методика позволяет в контролируемых условиях отладить алгоритмы регистрации, обработки и интерпретации сигналов, а также оценить точность различных подходов к распознаванию целевых стимулов.

1. Описание тестирования системы

На начальном этапе разработки интерфейса было принято решение предварительно протестировать работу алгоритмов на технической модели, имитирующей потенциал P300 в ответ на световую вспышку. В качестве модели использовалась тестовая установка, снабжённая светочувствительным датчиком.

При включении конкретной плашки фотодатчик фиксировал световую вспышку, и устройство генерировало тестовый сигнал с пиком в заданный момент времени, имитируя отклик типа P300 [3]. Блок-схема генерации сигнала представлена на рис. 1.

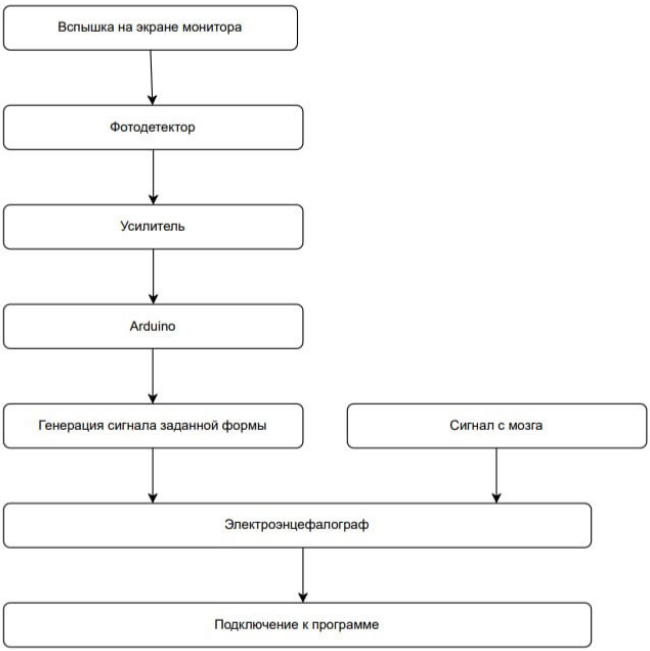


Рис. 1. Блок-схема генерации сигнала

Такой отклик регистрировался системой как «реакция на целевую плашку». При вспышках остальных плашек, на которые не был направлен датчик, реакция не формировалась, что соответствовало ситуации, когда пользователь не фокусирует внимание. Это позволило достоверно воспроизвести условия выбора плашки в реальной задаче, но в контролируемом виде без участия человека. Вид генерируемого сигнала представлен на рис. 2.

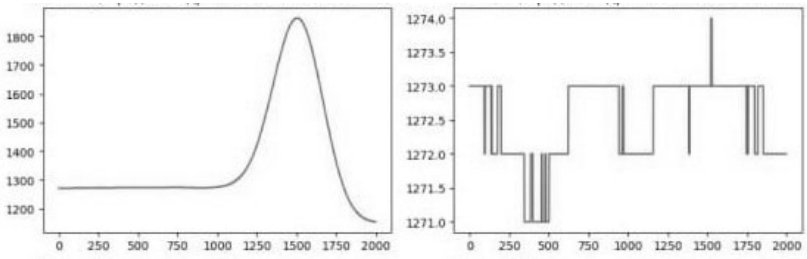


Рис. 2. Модельные сигналы, генерируемые датчиком при срабатывании на вспышку (левый график) и при её отсутствии (правый график)

Амплитуда модельного пика настраивалась вручную в диапазоне от 1 мкВ до 50 мкВ, что имитировало различную выраженность нейрофизиологической реакции [4].

Реализация гибридного нейроинтерфейса была выполнена с использованием двух взаимосвязанных программных компонентов: клиентской части на языке C#, отвечающей за работу с ЭЭГ-данными, и визуального интерфейса языке Python. Обмен данными между ними реализован по TCP-протоколу.

2. Процедура тестирования

Работа тестовой модели основывалась на последовательной регистрации откликов, формируемых в ответ на световой раздражитель, и дальнейшем анализе точности распознавания.

Процедура тестирования включала следующие шаги:

1. Формирование эталонного сигнала. К светочувствительному датчику подключалась одна из плашек. При её вспышке датчик срабатывал и генерировал одиночный импульс, имитирующий компоненту P300. Полученный сигнал фиксировался и сохранялся как эталонный отклик (аналог реакции мозга на выбранный объект).

2. Регистрация тестовых сигналов. Выбиралась плашка, которую предполагалось "отслеживать". Во время последовательного загорания

всех плашек многократно записывались сигналы: как с целевой плашки (где стоял датчик), так и с остальных (где датчик не срабатывал). Таким образом, формировались сигналы, содержащие как имитацию P300, так и фоновую активность.

3. Когерентное накопление. Для каждой плашки, независимо от наличия реакции, производилось когерентное накопление сигналов, соответствующих всем её вспышкам. Это позволяло повысить отношение сигнал/шум и структурировать данные.

4. Сравнение с эталоном. Полученные сигналы от каждой плашки подавались вместе с эталонным откликом на вход алгоритмам анализа (свёртка с гауссовыми окнами, каноническая корреляция, MSI). Каждый алгоритм возвращал индекс наиболее схожего сигнала, на основе которого принималось решение о распознавании.

Анализ точности. Эксперименты повторялись для разных значений амплитуды срабатывания (от 1 до 50 мкВ), и для каждой комбинации алгоритм–амплитуда рассчитывался процент правильно определённых плашек. Результаты экспериментов представлены таблице.

Таблица

Точность определения плашки каждым из алгоритмов при различных амплитудах модельного сигнала

Амплитуда, мкВ	**Алгоритм**	**Номер плашки**					
		1	2	3	4	5	6
1	Свёртка	100%	100%	100%	100%	100%	100%
	Корр. анализ	100%	95%	100%	95%	100%	100%
	MSI	10%	10%	0%	0%	10%	10%
3	Свёртка	100%	100%	100%	100%	100%	100%
	Корр. анализ	100%	100%	100%	100%	95%	100%
	MSI	90%	100%	95%	100%	100%	95%
5	Свёртка	100%	100%	100%	100%	100%	100%
	Корр. анализ	100%	100%	100%	100%	100%	100%
	MSI	100%	100%	100%	100%	100%	100%
10	Свёртка	100%	100%	100%	100%	100%	100%
	Корр. анализ	100%	100%	100%	100%	100%	100%
	MSI	100%	100%	100%	100%	100%	100%

Окончание таблицы

15	Свёртка	100%	100%	100%	100%	100%	100%
	Корр. анализ	100%	100%	100%	100%	100%	100%
	MSI	100%	100%	100%	100%	100%	100%
20	Свёртка	100%	100%	100%	100%	100%	100%
	Корр. анализ	100%	100%	100%	100%	100%	100%
	MSI	100%	100%	100%	100%	100%	100%
25	Свёртка	100%	100%	100%	100%	100%	100%
	Корр. анализ	100%	100%	100%	100%	100%	100%
	MSI	100%	100%	100%	100%	100%	100%
30	Свёртка	100%	100%	100%	100%	100%	100%
	Корр. анализ	100%	100%	100%	100%	100%	100%
	MSI	100%	100%	100%	100%	100%	100%
40	Свёртка	100%	100%	100%	100%	100%	100%
	Корр. анализ	100%	100%	100%	100%	100%	100%
	MSI	100%	100%	100%	100%	100%	100%
50	Свёртка	100%	100%	100%	100%	100%	100%
	Корр. анализ	100%	100%	100%	100%	100%	100%
	MSI	100%	100%	100%	100%	100%	100%

Из результатов следует, что:

– Метод свёртки показал стабильно высокую точность (100%) при всех амплитудах сигнала. Это указывает на его высокую чувствительность к слабовыраженным компонентам P300.

– Кросс-корреляционный анализ также продемонстрировал высокую точность, схожую со свёрткой, особенно при амплитудах от 3 мкВ и выше.

– Алгоритм MSI показал низкую точность на амплитуде 1 мкВ, однако при её увеличении демонстрировал рост точности. Это может свидетельствовать о его требовательности к выраженности паттерна сигнала и необходимости в более сильном ответе P300 для эффективного выявления.

Таким образом, тестирование на модельных данных подтвердило работоспособность предложенных методов. Полученные результаты позволили перейти к анализу реальных электроэнцефалографических записей, предполагая, что в условиях, приближенных к идеальным, модель уверенно определяет требуемый стимул [5].

Заключение

В данной работе была реализована и протестирована модель гибридного нейроинтерфейса для распознавания потенциала P300 в ответ на световой стимул. Проведённое моделирование позволило оценить эффективность различных алгоритмов обработки сигналов при имитации нейрофизиологического отклика на визуальные стимулы.

Разработанная система показала высокую точность определения целевого стимула в контролируемых условиях, что свидетельствует о её потенциальной применимости для дальнейшего использования в реальных задачах человеко-машинного взаимодействия. Следующим этапом предполагается проведение экспериментов с участием пользователей и адаптация алгоритмов под реальные ЭЭГ-сигналы. Это позволит оценить устойчивость системы в условиях биологической и поведенческой вариативности, а также расширить её функциональность за счёт интеграции дополнительных каналов анализа внимания.

Предложенный гибридный подход может стать основой для создания более надёжных и интуитивных нейроинтерфейсов, применимых в нейрореабилитации, виртуальной реальности и других сферах, где важно учитывать как когнитивное состояние, так и физическое направление внимания пользователя.

Список литературы

1. High frequency SSVEPs for BCI applications G. Garcia [Электронный ресурс]. – Режим доступа: http://hmi.ewi.utwente.nl/chi2008/chi2008_files/ garcia.pdf

2. Wang Yu., Wang Yi., Cheng Ch., Jung T. Developing Stimulus Presentation on Mobile Devices for a Truly Portable SSVEP-based BCI [Электронный ресурс]. – Режим доступа: http://ieeexplore.ieee.org/document/ 6610738/?section=abstract.

3. Comparison of the influence of stimuli color on Steady-State Visual Evoked Potentials / R. Tello [et al.] // Bastos Res. Biomed. Eng. – 2015. – Vol. 31. – Iss. 3. – P. 218-231.

4. Туровский, Я.А. Частотная модуляция энцефалограмма при фотостимуляции / Борзунов С.В., Алексеев А.В., Карпова М.А. // Биофизика. М.: ФИЦ "Пущинский научный центр биологических исследований РАН" Российская академия наук. – 2021. – №3. – С. 583-589.

5. Туровский, Я.А. Алгоритм коррекции статистического оценивания с учетом эффекта множественных сравнений на основе группировки результатов тестов / Борзунов С.В., Вахтин А.А. // Программная инженерия. – 2022. – №3. – С. 148-152.

Разработка картографического приложения PhotoMap

Е.С.Воронежская

Студент бакалавр

Е. А. Михайлюк

Доцент

Введение

Внешние социально-экономические и культурно-политические факторы значительно влияют на структуру внутреннего туристического рынка. Рост внутреннего и внутрирегионального туризма способствует экономическому развитию, занятости и наполнению бюджета. В этих условиях актуально создание цифровых решений для персонализированной маршрутизации и изучения территорий.

Цель исследования — обоснование и разработка веб-приложения «PhotoMap» для планирования маршрутов. В задачи входят анализ картографических сервисов, формулировка целей автоматизированной системы, изучение методических подходов и описание программной реализации.

1. Обоснование и актуальность разработки приложения

Структура внутреннего туристического рынка формируется под воздействием внешних экономических, политических и культурных факторов. Внутренний и внутрирегиональный туризм демонстрируют активный рост, а сама отрасль стала одним из наиболее динамично развивающихся сегментов мировой экономики, обладая высокой адаптивностью к макроэкономическим изменениям.

Туризм выполняет важные функции: пополнение бюджета, стимулирование инвестиций, создание рабочих мест (по прогнозу WTTC, к 2033 году — более 430 млн), а также развитие смежных отраслей и депрессивных территорий. Он способствует межкультурному взаимодействию, его вклад в глобальный ВВП составляет 7,6% [1].

В России развитие туризма определяется Стратегией до 2035 года, включающей создание системы туристской навигации. Наличие богатого природного и культурного потенциала усиливает потребность в точных веб-картографических сервисах для маршрутизации и доступа к информации о достопримечательностях.

© Воронежская Е. С., Михайлюк Е. А., 2025

В настоящее время актуальна разработка интеллектуальных решений, позволяющих строить персонализированные маршруты с фильтрацией по параметрам. Такие цифровые инструменты повысят удобство пользователей и будут способствовать развитию туризма, особенно в регионах с ограниченной инфраструктурой.

2. Сравнительный анализ картографических веб-сервисов

Интерактивные картографические приложения, доступные через браузер или API, обеспечивают отображение карт, построение маршрутов, интеграцию с данными о пробках, погоде и инфраструктуре.

Платформы делятся на ориентированные на пользователей сервисы и инструменты для разработчиков, такие как Mapbox, Leaflet и OpenLayers. Сравнение платформ приведено в таблице.

Современные решения переходят к динамическим веб-картам с настройкой визуального стиля, добавлением данных и аналитикой.

Таблица

Сравнение картографических платформ

Свойство	Google Maps	Яндекс Карты	Mapbox Maps
Целевая аудитория	Широкая (массовый пользователь)	Локальный пользователь (Россия, СНГ)	Разработчики
Уровень детализации	Высокий (глобальный охват)	Высокий в пределах РФ и СНГ	Зависит от источников
Интерактивность	Средняя	Средняя	Высокая
Визуальная кастомизация	Ограниченная	Ограниченная	Полная
API/SDK	Да (ограничено бесплатно)	Да (базовые возможности)	Да (расширенные возможности)

В условиях растущего спроса на адаптированные и кастомные решения в сфере геоинформатики, актуальными становятся платформы, сочетающие высокое качество визуализации, гибкость настройки и мощные аналитические инструменты [2].

3. Реализация

В процессе анализа предметной области была составлена диаграмма бизнес-процессов, представленная на рис. 1.

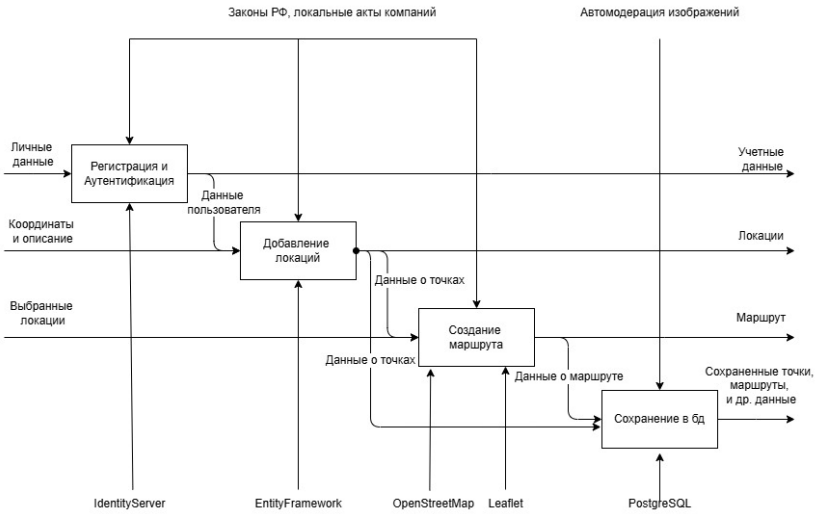


Рис. 1. Диаграмма бизнес-процессов.

На начальном этапе пользователь взаимодействует с системой посредством процесса регистрации, в рамках которого он предоставляет базовую информацию, необходимую для создания учетной записи. Параллельно реализуются механизмы безопасной аутентификации, направленные на защиту персональных данных и обеспечение конфиденциальности доступа к учетной записи. Аутентификация предполагает использование современных методов идентификации (OAuth 2.0), что минимизирует риск несанкционированного доступа.

После успешной регистрации пользователи получают возможность добавлять на карту новые географические объекты (локации). Каждая добавленная локация содержит ряд атрибутов, включая название, текстовое описание, категорию, изображения и отзывы, что позволяет формировать полную и содержательную характеристику объекта. Такой подход способствует созданию богатой информационной среды, в которой каждый участник может не только получать информацию, но и вносить вклад в ее развитие.

Следующим этапом пользовательского взаимодействия является формирование маршрутов. Пользователь может выбирать на карте несколько интересующих его точек, представляющих собой ранее добавленные локации, и на их основе формировать персонализированные маршруты. Интерфейс системы предусматривает возможность гибкой настройки маршрута — в частности, пользователь

имеет право добавлять или удалять промежуточные точки, в результате чего отображаемый маршрут автоматически адаптируется к внесённым изменениям.

Все введённые данные, включая локации и маршруты, сохраняются в базе данных. Это обеспечивает не только возможность повторного обращения к информации со стороны её автора, но и её доступность для других пользователей. Таким образом, реализуется механизм коллективного использования и хранения пространственных данных, который, при необходимости, может быть дополнительно задействован для аналитических целей или формирования отчётной документации.

В работе [3] содержится описание инструментов разработки приложения PhotoMap.

Работа с базами данных реализована через Entity Framework, позволяющий использовать объектно-ориентированный подход. Для валидации данных применяется FluentValidation, отличающийся выразительным синтаксисом. Отправка email-уведомлений осуществляется через SMTP-сервер Mail.ru.

Архитектура бэкенда организована по слоям: Data отвечает за работу с базой данных, Services — за бизнес-логику, Shared содержит общие компоненты и конфигурации, а Systems делится на слои API и Identity, реализующие взаимодействие с клиентом и процессы аутентификации соответственно. Каждый слой представлен подпроектами, соблюдающими принцип разделения ответственности.

Система рекомендаций в приложении PhotoMap основана на алгоритме матричной факторизации, позволяющем выявлять скрытые предпочтения пользователей на основе их взаимодействия с точками интереса. Исходной является матрица взаимодействий «пользователь — точка», где отметка «нравится» обозначается единицей. Модель предсказывает недостающие значения на основе латентных признаков.

Для реализации системы использован ML.NET — библиотека машинного обучения от Microsoft, глубоко интегрированная в .NET-экосистему. Она позволяет обучать модель на типизированных данных, сохранять её, и использовать без переобучения. Модель обучается на действиях пользователей (отметках «нравится»). После обучения осуществляется предсказание наиболее релевантных точек, которые пользователь ещё не оценил, и формируется топ-10 рекомендаций.

Техническая архитектура системы рекомендаций в PhotoMap включает несколько ключевых этапов: подготовку данных, обучение модели, генерацию рекомендаций и управление моделью.

На этапе подготовки обучающей выборки из базы извлекаются данные о лайках пользователей. Каждое взаимодействие преобразуется

в объект с числовыми идентификаторами пользователя и точки, полученными через SHA-256 хеширование Guid. Поскольку отметка «нравится» трактуется как положительная оценка, рейтинг фиксируется на уровне 1. Эти данные формируют основу для обучения.

Данные загружаются в модель в виде перечисления, после чего запускается pipeline обработки: идентификаторы преобразуются в категориальные признаки, и применяется алгоритм матричной факторизации с 20 итерациями и размерностью латентного пространства 100. Результатом является обученная модель, сохраняемая на диск или используемая немедленно.

Для генерации рекомендаций система либо загружает ранее обученную модель, либо обучает её заново. Затем извлекаются точки, с которыми пользователь не взаимодействовал, и создаётся тестовая выборка. Модель предсказывает вероятные рейтинги, после чего выбираются топ-10 наиболее релевантных точек.

Идентификаторы преобразуются из Guid в uint с помощью SHA-256, что снижает вероятность коллизий и обеспечивает равномерное распределение значений. Это критично для стабильной работы алгоритма.

Модель может быть сохранена и загружена повторно, что снижает вычислительные издержки. Выбор матричной факторизации обусловлен её эффективностью при работе с разреженными данными, характерными для PhotoMap. Такой подход позволяет выявлять скрытые предпочтения пользователей без опоры на контентные характеристики точек и масштабировать систему без существенного роста затрат. Использование ML.NET обеспечивает автономность и отказоустойчивость решения [4].

Для фронтенда использован фреймворк React и язык JavaScript, обеспечивающие модульность, переиспользуемость компонентов и высокую производительность за счёт виртуального DOM и одностороннего потока данных. Благодаря широкому сообществу и экосистеме, React предоставляет множество готовых решений для быстрой и удобной разработки.

Для работы с картой и построения маршрутов применяются библиотеки Leaflet и OpenStreetMap, позволяющие отображать точки на карте, строить маршруты и визуализировать их с детализацией. На рис. 2 представлено отображение маршрута между несколькими точками в приложении PhotoMap.

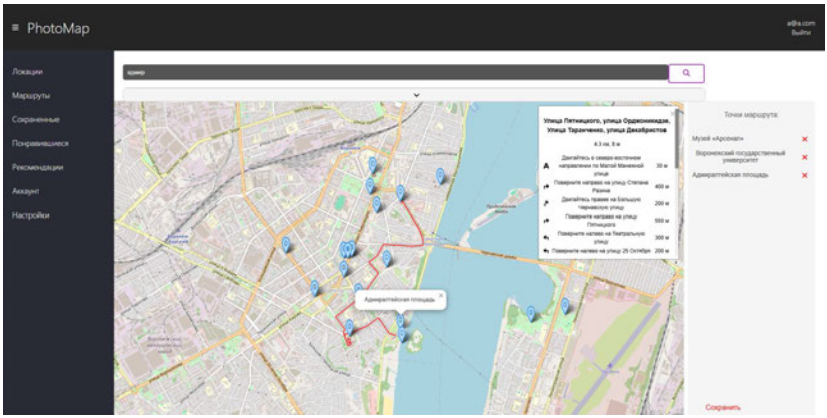


Рис. 2. Отображение маршрута в приложении PhotoMap.

Созданы основные страницы: Account для отображения данных пользователя, Dashboard для взаимодействия с точками интереса (добавление фото, отзывов, фильтрация по категориям), RoutePage для построения мультимаршрутов по интересующим локациям, Recommendations для отображения рекомендаций, Saved Routs для отображения и управления сохраненными маршрутами, Settings для изменения темы и настройки уведомлений. Интерфейс дополнен верхней панелью с навигационными элементами и боковым меню, переключающимся по клику.

Для обеспечения безопасности контента и соблюдения нормативных требований в системе реализована автоматическая модерация изображений с использованием внешнего сервиса машинного зрения SightEngine. Этот сервис анализирует изображения на предмет запрещённого содержания через API-интерфейс [5].

Процесс модерации начинается с отправки изображения на сервер SightEngine, где оно проходит проверку по заданным категориям. Анализ выполняется с помощью глубокой нейронной сети, которая определяет вероятность наличия нежелательного контента. В ответ система возвращает вероятностные оценки по каждой категории риска.

Если значение по одной из категорий превышает установленный порог, загрузка изображения автоматически блокируется. В противном случае оно сохраняется в облачном хранилище.

Применение SightEngine обеспечивает высокую точность, адаптивность к различным форматам изображений и возможность гибкой настройки порогов прохождения модерации.

Заключение

В ходе написания работы был проведен анализ предметной области картографических веб-сервисов, рассмотрены аналоги, описаны программные средства, а также подробно рассмотрена реализация веб-приложения PhotoMap.

Список литературы

1. Доходы глобального туризма достигнут $15,5 трлн к 2033 году. [Электронный ресурс]: Интерфакс. 21.08.2023. – Режим доступа: https://www.interfax.ru/business/917117
2. Веб-картография и интерактивные карты: конструктор карт онлайн [Электронный ресурс] : Молодой ученый. – 2023. – № 16 (463). – Режим доступа: https://moluch.ru/archive/463/102388/
3. Михайлюк, Е.А. Обоснование создания приложения для планирования маршрутов в сфере туризма / Е. А. Михайлюк, Е. С. Воронежская // Информатика: проблемы, методы, технологии. – 2024. – С. 88-92
4. Документация по ML.NET [Электронный ресурс] : техническая документация. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/machine-learning/
5. Документация по SightEngine [Электронный ресурс] : техническая документация. – Режим доступа: https://sightengine.com/docs/

Модули «Курсовые работы и проектные семинары» информационной системы факультета

В. Б. Новичихин

Студент бакалавр

А. И. Чекмарев

Старший преподаватель

Введение

В условиях современной образовательной системы, эффективность управления учебным процессом и контроль над выполнением курсовых и выпускных квалификационных работ становятся важными аспектами, влияющими на качество образования. Одним из ключевых элементов, обеспечивающих организацию учебного процесса, является балльно-рейтинговая система [1], которая позволяет объективно оценивать достижения студентов и стимулировать их к повышению учебной активности.

Одним из главных модулей этой системы должен быть модуль распределения курсовых и выпускных квалификационных работ. Данный модуль предназначается для автоматизации процесса назначения научных руководителей студентам, распределения тем работ и управления выполнением курсовых и выпускных проектов [2]. Это позволит существенно упростить работу как студентов, так и преподавателей, обеспечивая прозрачность и упорядоченность всех этапов подготовки и сдачи работ.

Особую значимость приобретает модуль «Курсовые работы и проектные семинары», интегрированный в информационную систему факультета. Данный модуль предназначен для автоматизации [3] процессов назначения научных руководителей, распределения тем, контроля сроков выполнения и оценки работ. Его внедрение позволит студентам упростить выбор тем и взаимодействие с преподавателями, преподавателям – снизить административную нагрузку, а администрации факультета – получать актуальную аналитику и равномерно распределять учебные нагрузки.

© Новичихин В. Б., Чекмарев А. И., 2025

1. Изменения в базе данных

В рамках разработки модуля была проведена модернизация структуры базы данных, включающая добавление новых полей в три ключевые таблицы. Изменения направлены на реализацию функциональности распределения научных работ и включают 3 существенные модификации.

В таблицу Curriculum_unit было введено поле use_topic, определяющее тип связи учебной единицы с научным руководством. Поле принимает 3 значения:

- 0 – не относится к работам с научным руководителем;
- 1 – курсовая/выпускная/научная работа;
- 2 – проектный семинар.

Данная модификация позволяет классифицировать виды учебной нагрузки и дифференцировать процессы назначения руководителей.

В таблицу Att_mark были добавлены 2 критически важных поля:

- teacher_id – внешний ключ, связывающий работу с научным руководителем из таблицы Teacher;
- work_theme – текстовое поле для хранения формулировки темы работы.

Эти изменения обеспечивают основу для системы назначения и контроля научного руководства.

В таблицу Att_mark_hist для сохранения истории изменений были добавлены аналогичные поля:

- teacher_id – для фиксации смены научного руководителя;
- work_theme – для отслеживания изменений темы работы.

Модификация позволяет сохранять полный аудит всех изменений, связанных с научным руководством, что особенно важно для:

- контроля качества образования;
- разрешения спорных ситуаций;
- анализа динамики работы студентов.

Реализация изменений была выполнена в СУБД MySQL 8.0 [4], что обеспечило:

- совместимость с существующей инфраструктурой факультета;
- поддержку транзакций при обработке связанных данных;
- механизмы блокировок для конкурентного доступа.

Модифицированная структура данных, включающая все описанные изменения, представлена в виде ER-диаграммы [5] (рис. 1), где красным выделены добавленные поля.

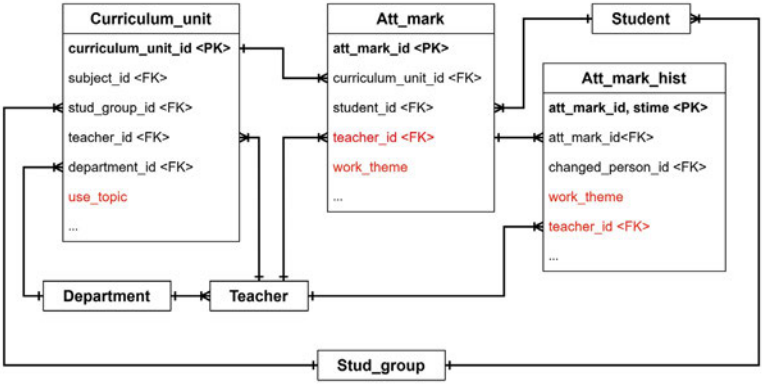


Рис. 1. ER-диаграмма

2. Модификация интерфейса создания учебных единиц

В рамках разработки модуля был модифицирован интерфейс страницы создания единицы учебного плана. Ключевым нововведением стало добавление селектора типа работы, позволяющего указать отношение единицы учебного плана к работе с руководителем:

– “Не относится к работам с руководителем” (значение по умолчанию);
– “Курсовая/выпускная/научная работа”;
– “Проектный семинар”.

При выборе варианта “Курсовая/выпускная/научная работа” в ведомости автоматически добавляются:

– столбец “Научный руководитель” (с выпадающим списком преподавателей);
– столбец “Тема работы” (текстовое поле для ввода).

При выборе “Проектный семинар” добавляется единственный столбец “Тема семинара” (текстовое поле).

Интерфейс страницы с нововведениями представлен на рис. 2.

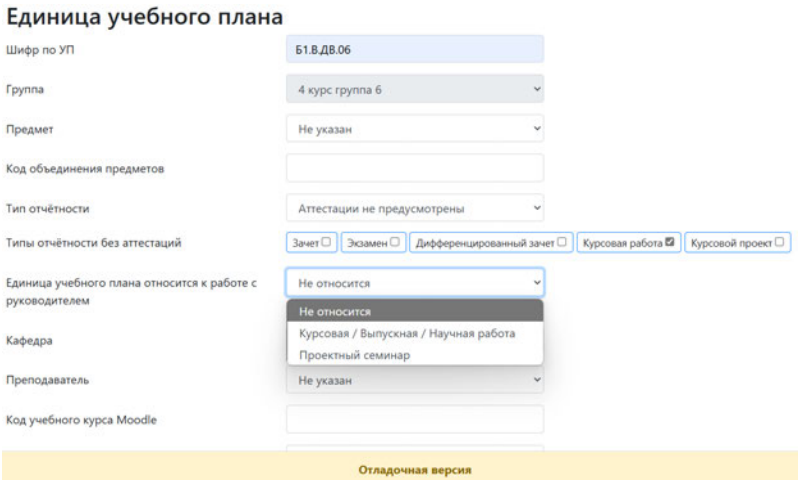


Рис. 2. Страница создания единицы учебного плана

3. Функциональность модуля для преподавателей

Интерфейс модуля предоставляет преподавателям следующие возможности:

– выбор студента, у которого еще нет научного руководителя, при этом в выпадающем списке доступен только текущий преподаватель (рис. 3);

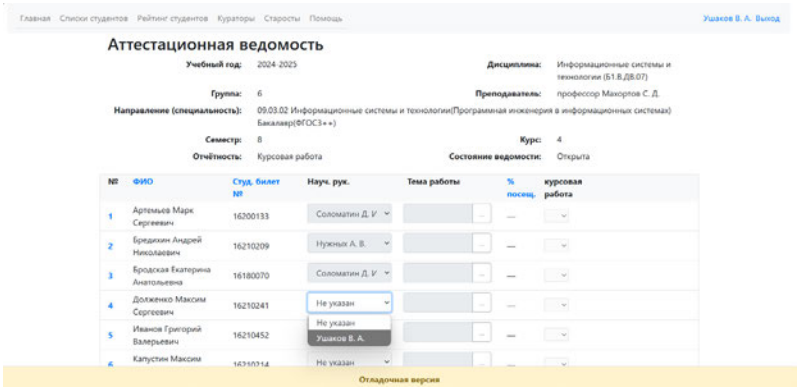


Рис. 3. Выбор научного руководителя

– написание и редактирование темы работы у тех студентов, для которых он является научным руководителем (рис. 4).

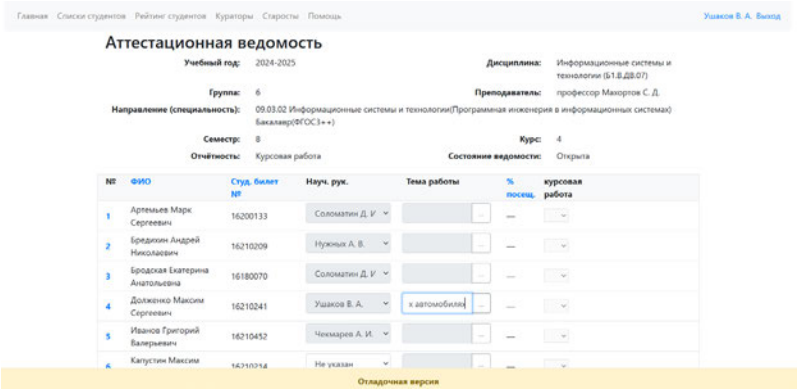


Рис. 4. Написание темы работы

Поскольку тема работы может быть гораздо длиннее поля ввода, у преподавателя есть возможность нажать на кнопку с многоточием справа от поля ввода, после чего откроется окно редактирования темы работы (рис. 5).

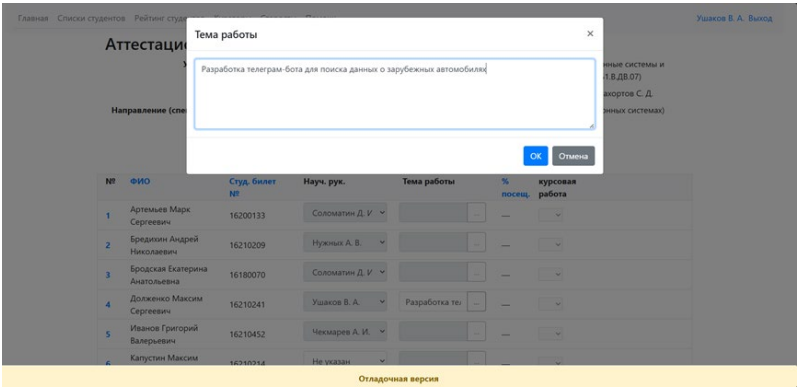


Рис. 5. Редактирование темы работы

Стоит отметить, что все действия, которые были описаны ранее, сохраняются и их легко можно посмотреть, нажав на номер студента по списку в аттестационной ведомости, после чего откроется полная история изменений аттестационной оценки с указанием времени и автора каждого изменения (рис. 6).

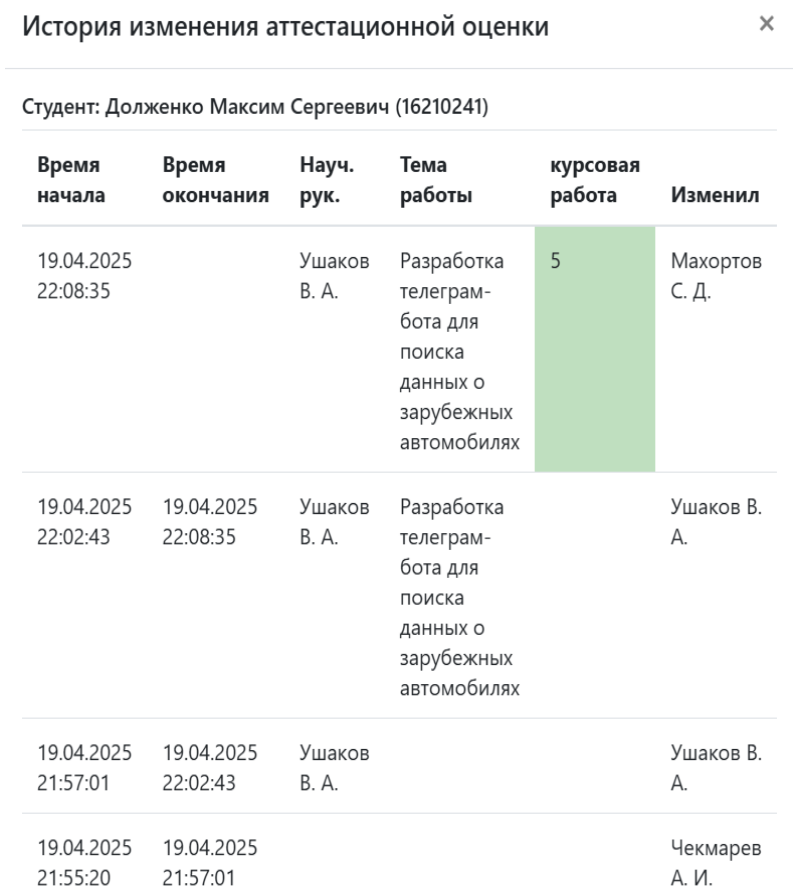
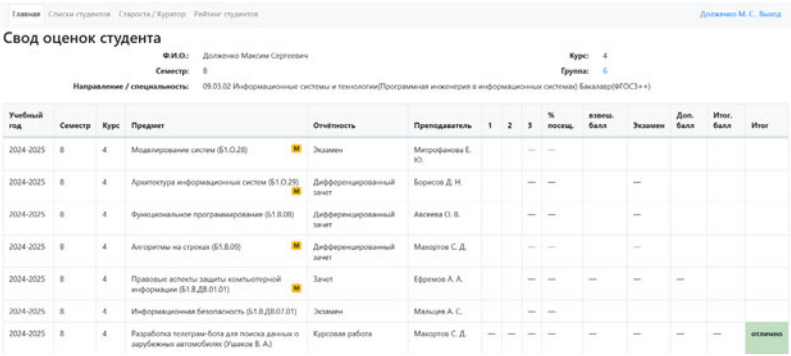


Рис. 6. История изменения аттестационной оценки

4. Функциональность модуля для студентов

Интерфейс модуля предоставляет студентам возможность просмотра информации о его работе, в том числе темы работы и научного руководителя (рис. 7).


Главная | Списки студентов | Староста / Куратор | Рейтинг студентов | Должненко М. С. | Выход

Свод оценок студента

ФИО.: Должненко Максим Сергеевич | Курс: 4

Семестр: 8 | Группа: 6

Направление / специальность: 09.03.02 Информационные системы и технологии(Программная инженерия в информационных системах) Бакалавр(ФГОС3++)

Учебный год	Семестр	Курс	Предмет	Отчётность	Преподаватель	1	2	3	% посещ.	взвеш. балл	Экзамен	Доп. балл	Итог. балл	Итог
2024-2025	8	4	Моделирование систем (Б1.О.28) М	Экзамен	Митрофанова Е. Ю.			—	—					
2024-2025	8	4	Архитектура информационных систем (Б1.О.29) М	Дифференцированный зачет	Борисов Д. Н.			—	—		—			
2024-2025	8	4	Функциональное программирование (Б1.В.08)	Дифференцированный зачет	Авсеева О. В.			—	—		—			
2024-2025	8	4	Алгоритмы на строках (Б1.В.09) М	Дифференцированный зачет	Макортов С. Д.			—	—		—			
2024-2025	8	4	Правовые аспекты защиты компьютерной информации (Б1.В_ДВ.01.01) М	Зачет	Ефремов А. А.			—	—	—	—	—		
2024-2025	8	4	Информационная безопасность (Б1.В_ДВ.07.01)	Экзамен	Мальцев А. С.			—	—					
2024-2025	8	4	Разработка телеграм-бота для поиска данных о зарубежных автомобилях (Ушаков В. А.)	Курсовая работа	Макортов С. Д.	—	—	—	—	—	—	—	—	отлично

Рис. 7. Информация о работах студента

Заключение

Данная статья посвящена обзору результатов разработки модуля «Курсовые работы и проектные семинары» информационной системы факультета. Описаны изменения в базе данных, а также интерфейс модуля для преподавателей и студентов. В результате проделанной работы получился полностью рабочий модуль, интегрируемый в балльно-рейтинговую систему факультета.

Список литературы

1. Балльно-рейтинговая система знаний [Электронный ресурс] : ЛаЛаЛань. – Режим доступа : https://lala.lanbook.com/ballno-rejtingovaya-sistema-ocenki-znanij
2. Отличие курсовой работы и проекта [Электронный ресурс] : work5. – Режим доступа : https://www.work5.ru/article/otlichie_kursovoj_raboty_i_proekta
3. Как я вуз автоматизировал [Электронный ресурс] : Habr. – Режим доступа : https://habr.com/ru/articles/899334/
4. MySQL 8.0 Reference Manual [Электронный ресурс] : Oracle. – Режим доступа : https://docs.oracle.com/cd/E17952_01/mysql-8.0-en/
5. What Is the Difference Between ER Diagrams and Database Schema? [Электронный ресурс] : Baeldung. – Режим доступа : https://www.baeldung.com/sql/entity-relationship-diagram-vs-database-schema

Оптимизация управления контрагентами в 1С: анализ проблем и пути решения

Е. С. Паневина

Студент бакалавр

С. В. Бебнева

Ст. преподаватель

Введение

В условиях современной экономики, характеризующейся высокой конкуренцией и динамичностью бизнес-процессов, эффективное управление контрагентами становится критически важным для успеха любой компании. Контрагенты, будь то поставщики, клиенты или партнеры, играют ключевую роль в обеспечении бесперебойной работы предприятия и достижении его стратегических целей. В этом контексте автоматизация процессов управления контрагентами приобретает особое значение, позволяя компаниям повышать эффективность, снижать затраты и минимизировать риски. В современном бизнесе эффективное управление контрагентами является ключевым фактором успеха. Платформа «1С: Предприятие» предоставляет мощные инструменты для автоматизации и оптимизации работы компаний. Однако, несмотря на обширные возможности, существуют недостатки, которые могут затруднить взаимодействие с контрагентами. В данной статье рассматриваются основные проблемы и предполагаются пути их решения для улучшения управления контрагентами в 1С.

1. Теоретические основы управления контрагентами в 1С: Предприятие

«1С: Предприятие» представляет собой единую платформу, предназначенную для автоматизации бизнес-процессов. Она использует собственный язык программирования для разработки прикладных решений, которые работают в рамках единой технологической платформы. Контрагент в 1С – это любое юридическое или физическое лицо, с которым организация вступает в деловые отношения, будь то поставщик, покупатель, подрядчик или партнер.

Система 1С позволяет формировать и хранить полную базу данных о контрагентах, включая юридическое и фактическое наименование,

© Паневина Е. С., Бебнева С. В., 2025

ИНН, КПП, ОГРН, прочие реквизиты, контакты, адреса, историю взаимодействий и операций. Это обеспечивает оперативный доступ к информации для всех участников процесса. Ключевым аспектом для успешного управления бизнесом является работа с контрагентами, и в системе 1С автоматизировано множество процессов, связанных с их учетом, что облегчает работу бухгалтерам и менеджерам.

2. Функциональные возможности для работы с контрагентами

Для грамотного управления данными о контрагентах в 1С предусмотрены функциональные возможности, которые значительно упрощают работу:

– Регистрация контрагента: система позволяет легко создать карточку контрагента, где фиксируются основные данные, такие как наименование, юридический адрес, контактные лица, банковские счета и др.;

– Классификация и сегментация: можно разделять контрагентов по категориям: покупатели, поставщики, партнеры. Это помогает организовать учет и взаимодействие с каждым типом контрагентов на основе их значимости и роли в бизнесе;

– Хранение истории взаимодействий: в карточке контрагента хранится вся история операций, что упрощает ведение отчетности и помогает анализировать долгосрочные отношения;

– Интеграция с другими модулями: данные о контрагентах в 1С можно использовать для формирования отчётности, ведения бухгалтерского учёта, управления заказами и для финансового анализа.

Ведение учета и проверка контрагентов в 1С дает бизнесу определенные преимущества, такие как снижение рисков, экономия времени и интеграция с бухгалтерским учетом. Благодаря удобной системе классификации и истории взаимодействий, работа с партнерами становится прозрачнее [1].

3. Недостатки управления контрагентами в 1С: Предприятие

Система 1С широко используется для автоматизации бухгалтерского учёта и управления бизнесом. Однако, несмотря на обширные возможности, существуют недостатки, которые могут затруднить эффективное взаимодействие с контрагентами.

Основные недостатки:

– Неполное заполнение обязательных полей: это может привести к недостатку информации, необходимой для корректного учета и взаимодействия с контрагентами;

– Отсутствие комментариев и примечаний: информация, которая не подходит под стандартные поля, может затеряться вследствие их отсутствия для комментариев и примечаний;

– Недостаток информации о контактных лицах: может быть необходима подробная информация о контактных лицах контрагента, включая должности, контактные данные (телефон, email), полномочия и т. д.;

– Устаревшая информация: отсутствие актуальных данных затрудняет связь с контрагентами. Это особенно критично в случае срочных вопросов или изменений в условиях сотрудничества;

– Отсутствие коммуникации с контрагентом: для получения дополнительной информации будут полезны ссылки на профили в социальных сетях и веб-сайты [2].

4. Методы решения выявленных проблем управления контрагентами

Рассмотренные недостатки могут влиять на эффективность управления контрагентами и требуют внимания для их устранения или минимизации. Они могут быть устранены с помощью простых, но эффективных мер.

– Неполное заполнение обязательных полей: решением этой проблемы будет являться настройка обязательных полей. Все критически важные поля (например, ИНН, КПП, юридический адрес) должны быть отмечены как обязательные для заполнения. Это потребует анализа бизнес-процессов и определения, какие именно поля являются критически важными;

– Отсутствие комментариев и примечаний: для решения данной проблемы стоит включить в карточки контрагентов поля для ввода комментариев и примечаний. Если присутствует некоторое количество повторяющихся комментариев и примечаний, можно создать шаблоны комментариев, чтобы ускорить процесс ввода данных;

– Недостаток информации о контактных лицах: чтобы избежать этой проблемы, необходимо расширить карточки контрагента добавлением поля для ввода информации о контактных лицах;

– Отсутствие ссылок на профили в социальных сетях и веб-сайты: для их добавления нужно включить в карточку контрагента поля для ввода ссылок на профили в социальных сетях и веб-сайты. Возможно использование разных типов полей, а также интеграция 1С с API социальных сетей для автоматического обновления информации о контрагентах. Если используется CRM-система, можно настроить синхронизацию данных между 1С и CRM для обмена информацией о контрагентах.

Внедрение предложенных решений позволит значительно улучшить управление контрагентами в 1С. Настройка обязательных полей, добавление возможностей для ввода комментариев и примечаний, а также расширение информации о контактных лицах и интеграция с внешними источниками данных помогут создать более полное и актуальное представление о контрагентах. Это, в свою очередь, положительно скажется на эффективности бизнес-процессов, снижении рисков и улучшении коммуникации с партнерами. Реализация этих мер требует не только технической настройки системы, но и обучения пользователей, а также регулярного мониторинга и обновления данных. В результате компания сможет оперативно реагировать на изменения, поддерживать высокий уровень взаимодействия с контрагентами и повышать общую производительность бизнеса. В условиях быстро меняющейся деловой среды, такие улучшения становятся особенно актуальными и могут служить основой для дальнейшего развития и оптимизации бизнес-процессов [3].

Заключение

Данная статья посвящена анализу проблем и разработке решений для оптимизации управления контрагентами в системе «1С: Предприятие». В работе были рассмотрены ключевые аспекты взаимодействия с контрагентами, выявлены основные проблемы, с которыми сталкиваются пользователи при работе с платформой, и предложены эффективные пути их решения. В ходе исследования были проанализированы такие проблемы, как неполное заполнение обязательных полей, отсутствие полей для комментариев и примечаний, недостаток информации о контактных лицах, отсутствие коммуникации с контрагентом.

Для каждой из этих проблем были предложены конкретные решения, включая использование специализированных модулей и API для интеграции, применение готовых решений и шаблонов для упрощения настройки, внедрение дополнительных инструментов анализа данных и усиление механизмов безопасности.

В дальнейшем предполагается развитие предложенных решений, включая более глубокую интеграцию с внешними системами, расширение аналитических возможностей и усиление механизмов безопасности. Это позволит компаниям еще более эффективно управлять взаимодействием с контрагентами и достигать стратегических целей в условиях современной бизнес-среды.

Список литературы

1. Хрусталева Е. Ю. Технологии интеграции 1С: Предприятия 8.3 / Е.Ю. Хрусталева. – М. : ООО «1С-Паблишинг», 2023. – 503 с.

2. Филатова О. А. Автоматизация бизнес-процессов в 1С: Предприятие / О. А. Филатова. – М. : Инфра-Инженерия, 2022. – 320 с.

3. Габец А. П., Гончаров Д. И. 1С: Предприятие 8.3. Программирование и визуальная разработка на примерах / А. П. Габец, Д.И. Гончаров. – М. : БХВ-Петербург, 2020. – 544 с.

Генеративная аугментация данных для задач детектирования летательных аппаратов

А. С. Пекшев

Студент бакалавр

А. В. Акимов

Старший преподаватель

Введение

Детектирование летательных аппаратов: самолетов, дронов, вертолетов является актуальной задачей в различных областях, включая системы безопасности, мониторинг воздушного пространства и автономную навигацию. Обучение высокоточных моделей детектирования, таких как YOLOv8, требует значительных объемов разнообразных и качественно размеченных данных. Однако сбор и разметка таких датасетов часто сопряжены с большими временными и ресурсными затратами, что приводит к проблеме дефицита данных. Стандартные методы аугментации такие как геометрические или цветовые преобразования не всегда способны создать достаточное разнообразие для повышения робастности моделей в сложных условиях. В связи с этим, генеративные методы аугментации, основанные на диффузионных моделях, представляют значительный интерес для создания синтетических данных, обогащающих обучающие выборки.

Целью данной работы является исследование и оценка эффективности методов генеративной аугментации данных для повышения качества модели детектирования воздушных объектов на базе YOLOv8 в условиях ограниченного набора реальных данных [1].

1. Обзор методов аугментации данных

Аугментация данных – это процесс искусственного увеличения размера и разнообразия обучающего набора данных путем создания модифицированных копий существующих данных или генерации полностью новых синтетических данных.

К стандартным методам, широко применяемым в задачах детектирования, относятся.

– Геометрические преобразования: Изменение масштаба, случайные повороты, сдвиги, отражения и обрезка (cropping). При их

© Пекшев А.С., Акимов А.В., 2025

применении важно корректно трансформировать координаты bounding boxes – ограничивающих рамок объектов.

– Фотометрические преобразования: изменение яркости, контрастности, насыщенности, цветового тона изображений. Эти преобразования обычно не требуют изменения аннотаций.

– Комбинированные методы: Современные фреймворки, такие как YOLOv8, часто используют сложные комбинации аугментаций, например, Mosaic (объединение четырех разных изображений и их объектов в одно) или MixUp (линейная интерполяция между двумя изображениями и их метками) [1].

Генеративные подходы предлагают более глубокий уровень модификации данных. Среди них можно выделить:

– Генеративно-состязательные сети (GAN): Способны генерировать новые изображения, переносить стиль (например, CycleGAN для смены времени суток или сезона). Однако обучение GAN может быть нестабильным, а сохранение деталей объектов и валидности аннотаций – сложной задачей [2].

– Диффузионные модели (например, Stable Diffusion): Современный класс моделей, демонстрирующий впечатляющие результаты в генерации высокореалистичных изображений.

Для аугментации в задачах детекции интерес представляют следующие подходы:

– Image-to-Image (Img2Img): Модификация исходного изображения на основе текстового промпта. Высокий риск искажения или удаления исходного объекта делает этот метод сложным для задач детекции, требующих сохранения аннотаций.

– Inpainting: Заполнение определенной области изображения, заданной маской, на основе текстового промпта. Этот метод был выбран как основной для генерации синтетических данных в данной работе. Принцип заключается в том, чтобы вырезать объект из исходного изображения, сохранить его, сгенерировать новый фон для исходного изображения (замаскировав место, где был объект), а затем вклеить сохраненный объект на новый фон. При этом сам объект может быть дополнительно аугментирован стандартными методами перед вклейкой [3].

На рис. 1 представлено схематичное представление процесса работы с использованием inpainting подхода.

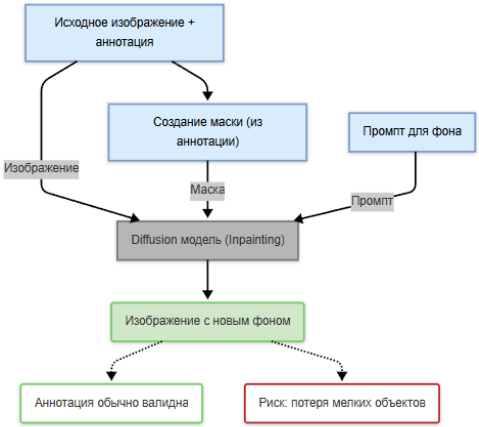


Рис. 1. Схематическое изображение процесса inpainting для аугментации

2. Методология эксперимента

Для оценки эффективности генеративной аугментации в условиях дефицита данных был проведен следующий эксперимент. В качестве основного набора данных использовался датасет "Drone Detection" с платформы Kaggle, содержащий около 12 тысяч изображений с тремя классами объектов: 'AirPlane', 'Drone', 'Helicopter' [4].

Основной эксперимент включал следующие шаги.

Отбор 100 реальных изображений из обучающей выборки стратифицировано по классам и количеству объектов.

Генерация 900 новых изображений с использованием метода Inpainting (Stable Diffusion InpaintPipeline). Процесс включал вырезание объектов с оригинальных изображений, генерацию разнообразных фонов по текстовым промптам, например, "чистое голубое небо" или "пасмурное серое небо" (в коде они применяются на английском языке, чтобы они были понятны модели). После этого вклеиваются объекты на новые фоны с сохранением их аннотаций. При этом для аугментации самих вырезанных объектов перед вклейкой применялись стандартные преобразования из библиотеки Albumentations [5].

Обучение модели YOLOv8n на полученном датасете из 1000 изображений (100 реальных + 900 сгенерированных).

Обучение контрольных моделей YOLOv8n на 100 реальных изображениях и на 1000 реальных изображениях без стандартных аугментаций YOLO для чистоты сравнения [6].

На рис. 2 представлен пример результата аугментации изображения с дроном.

а б

Рис. 2. Пример преобразования изображения: а – исходное изображение, б – аугментированное изображение

Работа выполнялась на языке Python с использованием фреймворков PyTorch, Ultralytics YOLOv8 и библиотеки Hugging Face Diffusers для работы с диффузионными моделями. Для обработки изображений применялись OpenCV и Pillow [5-6].

Для оценки качества моделей детектирования использовались стандартные метрики.

– Precision (P, точность): Доля правильно идентифицированных положительных объектов (True Positives, TP) среди всех объектов, предсказанных моделью как положительные (количество истинно положительных – True Positives, TP и ложноположительных срабатываний – False Positives, FP).

$$P = \frac{TP}{TP + FP}, \quad (1)$$

– Recall (R, полнота): Доля правильно идентифицированных положительных объектов (True Positives, TP) среди всех реально существующих положительных объектов (сумма истинно положительных – True Positives и ложноотрицательных – False Negatives, FN).

$$R = \frac{TP}{TP + FN}, \quad (2)$$

– mAP@0,5 (mean Average Precision при IoU=0,5): Средняя точность для всех классов при пороге пересечения над объединением (Intersection over Union, IoU) равном 0,5.

– mAP@0,5:0,95 (mean Average Precision): Средняя точность, усредненная по различным порогам IoU (от 0,5 до 0,95 с шагом 0,05).

– Fitness: Комбинированная метрика, часто используемая в YOLO, обычно взвешенная сумма mAP@0,5 и mAP@0,5:0,95 [7].

3. Результаты экспериментов и их анализ

Оценка моделей проводилась на стандартной тестовой выборке датасета. Ключевые метрики производительности моделей представлены в таблице.

Таблица

Результаты сравнения моделей

Датасет	mAP 0,5	mAP 0,5:0,95	Точность	Полнота	Fitness
Реальные 100 изображений	0,5231	0,2314	0,5768	0,4805	0,2606
1000 изображений полученные из 100 исходных	0,5721	0,2614	0,7227	0,4907	0,2926
Реальные 1000 изображений	0,9033	0,5225	0,8991	0,8855	0,5605

Анализ результатов.

– Средняя точность при пороге IoU=0,5 (mAP@0,5) увеличилась с 0,5231 до 0,5721, что составляет прирост примерно на 9,37%. Это указывает на то, что модель стала значительно лучше локализовывать и классифицировать объекты при умеренно строгом критерии перекрытия.

– Более строгая метрика mAP@0,5:0,95, усредняющая точность по диапазону порогов IoU, также продемонстрировала рост – с 0,2314 до 0,2614 (прирост около 12,96%). Это свидетельствует об улучшении способности модели к более точной локализации объектов.

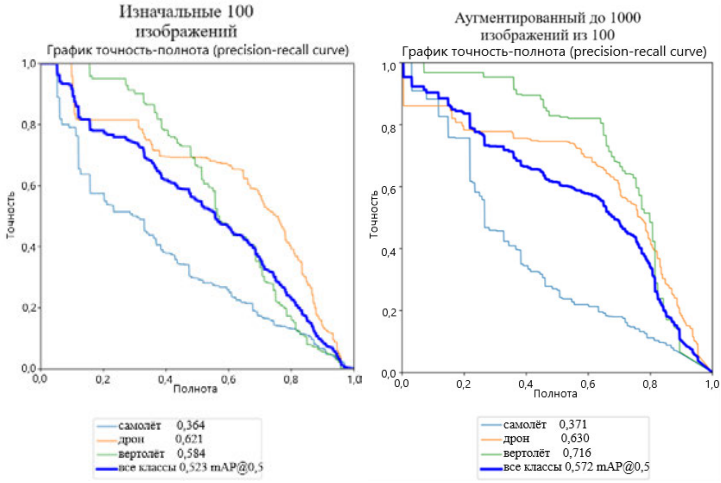
– Точность (Precision) показала наиболее значительный рост: с 0,5768 до 0,7227 (прирост примерно 25,29%). Такой результат говорит о том, что модель, обученная на дополненных данных, стала гораздо реже делать ложные срабатывания (предсказывать объекты там, где их нет, или неверно классифицировать).

– Полнота (Recall) также незначительно улучшилась с 0,4805 до 0,4907 (прирост около 2,12%). Это означает, что модель стала обнаруживать немного большую долю реально существующих объектов.

– Интегральная метрика Fitness увеличилась с 0,2606 до 0,2926, подтверждая общее улучшение производительности модели.

Несмотря на существенный прогресс, достигнутый за счет генеративной аугментации, производительность модели обученной на аугментированном датасете все еще значительно уступает модели, обученной на 1000 реальных изображениях. Эталонная модель демонстрирует mAP@0,5 равный 0,9033 и mAP@0,5:0,95 равный 0,5225. Это подчеркивает, что качество и разнообразие реальных данных пока остаются непревзойденными для обучения высокопроизводительных моделей. Синтетические данные, сгенерированные текущим методом, хотя и полезны, не могут полностью компенсировать недостаток реальных примеров.

На рис. 3 представлены графики точность-полнота (Precision-Recall curve), на которых более детально можно отследить изменения по классам.



а б

Рис. 3. Кривые Precission-Recall для сравниваемых моделей: а – обученная на выборке 100 исходных изображений, б – обученная на выборке 1000 изображений, полученной из 100 исходных применением аугментации

Наибольшее улучшение после генеративной аугментации наблюдается для класса «вертолёт» — mAP@0,5 вырос с 0,584 до 0,716, а PR-кривая показывает явный прирост точности при средних и высоких значениях полноты, что указывает на успешное расширение разнообразия обучающих данных для этого класса. Класс «дрон», который изначально имел лучший результат 0,621, получил лишь незначительное улучшение до 0,630, что может свидетельствовать о хорошей начальной представленности в датасете и ограниченном эффекте дополнительной аугментации. Для класса «самолёт» наблюдается минимальный прирост — с 0,364 до 0,371, при этом точность остаётся низкой на большинстве уровней полноты, что может быть связано со сложностью аугментации изображений самолётов, у них в отличие от дронов и вертолётов, есть сложность в отрисовке и потере на изображениях конденсационного следа, что может усложнять работу для генеративной модели.

4. Заключение

В данной статье проведено исследование эффективности генеративной аугментации данных для задачи детектирования летательных аппаратов с использованием модели YOLOv8. Разработана и апробирована методология, основанная на методе Inpainting с применением Stable Diffusion для обогащения ограниченного набора обучающих данных.

Экспериментальные результаты подтвердили, что использование данного решения способно улучшить качество модели детектирования по сравнению с обучением только на малом объеме реальных данных, в частности, повышая метрики mAP и Precision. Однако достигнутый уровень производительности все еще ниже, чем при обучении на эквивалентном объеме реальных данных.

Были выявлены некоторые проблемы, связанные с применением генеративных методов: Inpainting может испытывать трудности с сохранением очень мелких объектов, а также с сохранением более сложного контекста. Это указывает на необходимость дальнейших исследований в области контроля процесса генерации и валидации синтетических данных.

Тем не менее, полученные результаты демонстрируют перспективность использования Inpainting-подхода для генеративной аугментации в задачах компьютерного зрения при дефиците данных.

Список литературы

1. Redmon, J. You Only Look Once: Unified, Real-Time Object Detection / J. Redmon, S. Divvala, R. Girshick, A. Farhadi // Proceedings of

the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2016.

2. Okunev, S. V. Application of augmentation and a generative-adversarial neural network to increase data sets / S. V. Okunev // Молодежь. Общество. Современная наука, техника и инновации. – 2020. – № 19. – С. 215–217.

3. Rombach, R. High-Resolution Image Synthesis with Latent Diffusion Models / R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – 2022.

4. Датасет "Drone Detection". Kaggle [Электронный ресурс]. – Режим доступа: https://www.kaggle.com/datasets/cybersimar08/drone-detection

5. Документация Hugging Face Diffusers [Электронный ресурс]. – Режим доступа: https://huggingface.co/docs/diffusers

6. Документация Ultralytics YOLOv8 [Электронный ресурс]. – Режим доступа: https://docs.ultralytics.com/

7. Документация Ultralytics YOLOv8 metrics [Электронный ресурс]. – Режим доступа: https://docs.ultralytics.com/reference/utils/metrics/

Анализ и исследование некоторых методов прогнозирования нестационарных временных рядов

М. С. Петренко

Студент бакалавр

Е. А. Сирота

Доцент

Введение

Анализ и обработка временных рядов (ВР) является важной задачей в различных прикладных областях, например, в экономических, промышленных, метеорологических, и т. д.

Большинство процессов, которые описываются нестационарными ВР, включают изменения свойств или переходы между состояниями. Переходы между состояниями характеризуются изменениями в поведенческих закономерностях процесса и могут происходить в произвольные моменты времени под влиянием внешних факторов. Невыявленные изменения часто приводят к ошибочным выводам при анализе процесса.

Анализ процесса в условиях возможных изменений отличается от анализа в условиях их отсутствия. Поэтому очень важно перед применением методов прогнозирования определить тип процесса, а после определяться с технологией обработки ВР.

Таким образом, целью данной работы является анализ и исследование некоторых методов прогнозирования (AR, ARIMA, модель Хольта-Винтерса), реализация методов определения типа процесса, разработка библиотеки, определяющей тип процесса и предоставляющей рекомендации по дальнейшему анализу ВР.

1. Типы ВР

Математическое описание временного ряда может представлять собой сумму нескольких компонент или только одну из них: тренд или систематическое движение - устойчивое систематическое изменение направления движения ряда в течение длительного периода, колебания относительно тренда, сезонные изменения и «нерегулярная» «случайная» составляющая [1].

© Петренко М. С., Сирота Е. А., 2025

При моделировании реальных процессов возникает задача: имеется реализация ряда, и необходимо подобрать модель, способную точно генерировать такие данные.

Стационарность временного ряда — это свойство, при котором статистические характеристики ряда, остаются неизменными с течением времени. Значения автокорреляционной функции уменьшаются с увеличением времени между наблюдениями.

В случае нестационарности, напротив, такие характеристики как среднее и дисперсия зависят от времени, дисперсия стремится к бесконечности, автокорреляционная функция не уменьшается с увеличением времени между наблюдениями, и в конечных выборках выборочная автокорреляционная функция затухает медленно. Нестационарность временных рядов может быть двух типов: тренд-стационарный и разностно-стационарный процессы. Понимание конкретного типа нестационарности критически важно для выбора правильного метода преобразования данных и дальнейшего прогнозирования временного ряда [2].

2. Тренд-стационарный (TS) процесс

Тренд-стационарный временной ряд имеет детерминированную тенденцию роста или снижения, то есть является стационарным относительно детерминированного тренда. Математически это можно представить как

$$y_t = f(t) + z_t, \tag{1}$$

где $f(t)$ - чисто детерминированная составляющая, z_t - случайный стационарный процесс дисперсия и математическое ожидание «белого шума» равны 0. Влияние случайных воздействий на значения ВР затухает с течением времени [1].

Пример тренд-стационарного процесса представлен на рис. 1.

3. Разностно-стационарный (DS) процесс

Интегрированный, разностно-стационарный процесс – это процесс, в котором тренд может быть и стохастическим, и детерминированным. Затухание отсутствует [1]. Всякое случайное воздействие влияет на ряд одинаково. Процесс описывается моделями авторегрессии с «единичным корнем»:

$$y_t = \mu + \rho y_{t-1} + z_t \tag{2}$$

где z_t - случайный стационарный процесс, $\rho = 1$ - константа, которая определяет величину дрейфа. [3].

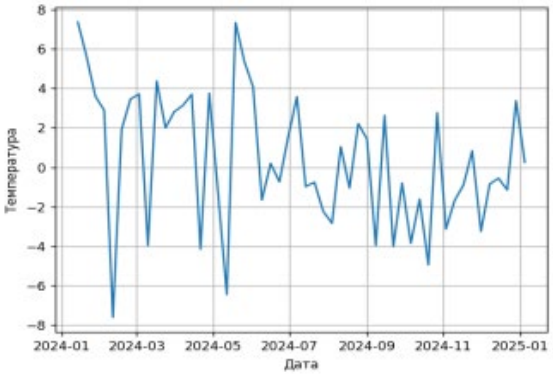


Рис. 1. Пример TS-ряда

Пример разностно-стационарного процесса представлен на рис. 2.

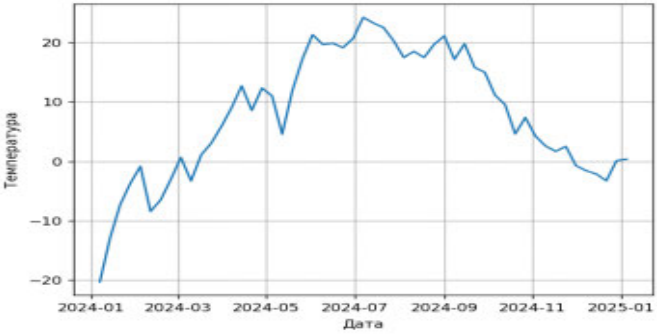


Рис. 2. Пример DS-ряда

4. Необходимость определения типа процесса

Понимание конкретного типа временного ряда критически важно по следующим причинам:

1. Наличие адекватной модели временных рядов наблюдений, описывающих поведение системы, является необходимым условием для проведения анализа.

2. Если в процессе возможны изменения свойств, то для успешного прогнозирования и мониторинга, после построения модели на интервале без изменений, необходимо регулярно проверять её адекватность по мере поступления новых данных.

3. Мощность тестов на проверку типа процесса значительно снижается при наличии изменений в процессах.

Методика проверки временных рядов на стационарность включает в себя целый ряд тестов, направленных на выявление «единичного корня», базовым из которых является тест Дики–Фуллера [1, 2].

5. Гипотеза единичного корня

Понятие «единичного корня» используется в анализе временных рядов как характеризующее свойство некоторых нестационарных временных рядов. Наличие единичных корней в авторегрегрессионной модели временного ряда эквивалентно понятию интегрированности временного ряда [3].

$$\Delta y_t = \mu + \alpha y_{t-1} + \sum_{i=1}^{n} \alpha_i y_{t-i} + z_t. \quad (3)$$

В тесте Дики–Фуллера для (4) проверяется гипотеза $H_0 : \alpha = 0$ против $H_1 : \alpha \neq 0$. Для этого используется t-статистика, подчинённая распределению Дики–Фуллера. Гипотеза H_0 отвергается, если $|t| > t_{кр}$.

6. Удаление тренда

После определения типа нестационарности временного ряда необходимо применить соответствующие методы преобразования для достижения стационарности [2]. Как правило, необходимо применять разные методы удаления тренда в зависимости от типа временного ряда: для ряда тренд-стационарного типа применяется операция вычитания компоненты тренда из исходного ряда, а для разностно-стационарного ряда применяется дифференцирование. Пример результата выполнения процедуры приведения временного ряда к стационарному процессу представлен на рис. 3.

Рис. 3. Пример процесса удаления тренда

7. Авторегрессионная модель AR

Авторегрессионная модель (AR(p)) — это статистическая модель временных рядов, при которой текущее значение переменной объясняется её предыдущими значениями и случайным отклонением (ошибкой):

$$x_t = \phi_1 x_{t-1} + ... + \phi_p x_{t-p} + e_t \,, \tag{4}$$

при этом текущее значение не коррелирует с лагами [3].

8. Авторегрессионная скользящая средняя ARMA

Авторегрессионная модель со скользящим средним (ARMA) объединяет преимущества моделей авторегресссии и скользящей средней: учитывает и влияние прошлых значений переменной (авторегрессия), и ошибки предыдущих прогнозов (скользящее среднее) [2].

$$x_t = \phi_1 x_{t-1} + ... + \phi_p x_{t-p} + e_t - \theta_1 e_1 - ... - \theta_q e_{t-q} \,. \tag{5}$$

Частными случаями являются модели ARIMA (авторегрессионная интегрированная скользящая средняя), SARIMA (сезонная авторегрессионная интегрированная скользящая средняя) [4].

9. Модель Хольта-Винтерса

В модели Хольта-Винтерса используется тройное экспоненциальное сглаживание: учитывается сглаженный экспоненциальный ряд, тренд, сезонность [4].

Расчёт сглаженного экспоненциального ряда:

$$E_t = \alpha \frac{Y_t}{S_{t-s}} + (1-\alpha)(E_{t-1} - T_{t-1}) \,. \tag{6}$$

Определение значений тренда:

$$T_t = \beta(E_t - E_{t-1}) + (1-\beta)T_{t-1} \tag{7}$$

Оценка сезонности:

$$S_t = \gamma \frac{Y_t}{E_t} + (1-\gamma)S_{t-s} \,. \tag{8}$$

Прогнозирование на период:

$$Y_t = \beta(E_t - T_t t)S_{t-s} \,. \tag{9}$$

10. Численный эксперимент. Сравнение результатов прогнозирования ВР некоторых моделей

С целью определения типа процесса, а также выявления модели с наилучшим качеством был проведён численный эксперимент на модельных данных ежедневных показателей температуры воздуха в г. Москва за период 1.01.2024–1.01.2025. Данные взяты с сайта http://meteo.ru/data/temperature-precipitation/. В рамках эксперимента были реализованы модели авторегресии, авторегрессионной скользящей средней, Хольта–Винтерса. На первом этапе проводилось определение типа процесса с помощью теста Дики–Фуллера и дальнейший процесс удаления тренда, отображённый на рис. 3.

Следующий этап - применение вышеуказанных моделей для задачи прогнозирования модельных данных и оценка качества прогноза.

Пример прогнозирования значений временного ряда моделями авторегресии, авторегрессионной скользящей средней, Хольта–Винтерса при правильном определении типа процесса с помощью разработанного программного модуля представлен на рис. 7.

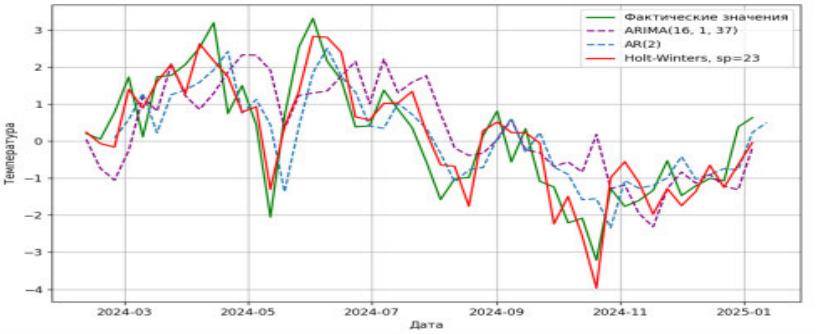


Рис. 4. Пример прогнозирования значений ВР с помощью некоторых моделей при правильном определении типа процесса

Показатели функций потерь MAE, MSE, MAPE для оценки адекватности моделей авторегрессии, авторегрессионной скользящей средней, Хольта–Винтерса в задаче прогнозирования временного ряда с правильным определением типа процесса приведены в табл. 1.

Таблица 1

Показатели ошибок

Метод прогнозирования	MSE	MAE	MAE
ARMA	1.83	1.07	1.68
AR	0.95	0.75	1.2
Модель Хольта–Винтерса	0.43	0.56	0.78

Из табл. 1 видно, что наиболее эффективной по сравнению с моделями авторегресии (AR) и авторегрессионной скользящей средней (ARMA) является модель Хольта–Винтерса.

Пример прогнозирования значений временного ряда моделями авторегресии, авторегрессионной скользящей средней, Хольта–Винтерса при некорректном определении типа процесса с помощью разработанного программного модуля представлен на рис. 5.

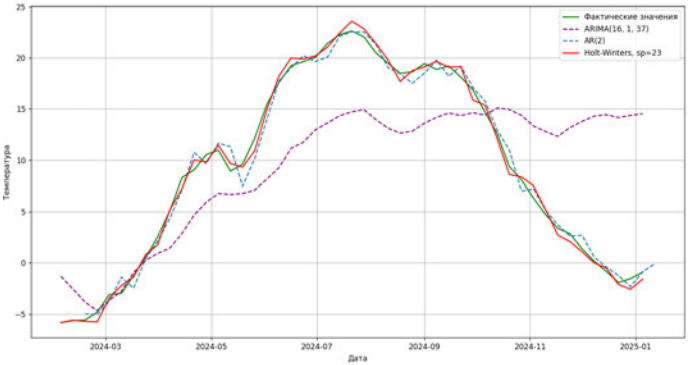


Рис. 5. Пример прогнозирования значений ВР с помощью некоторых моделей при неправильном определении типа процесса

Показатели функций потерь MAE, MSE, MAPE для моделей авторегрессии, авторегрессионной скользящей средней, Хольта-Винтерса в задаче прогнозирования временного ряда с некорректным определением типа процесса приведены в табл. 2:

Таблица 2

Показатели ошибок

Метод прогнозирования	MSE	MAE	MAE
ARMA	54.57	6.08	3.48
AR	7.06	2.22	0.73
Модель Хольта–Винтерса	0.42	0.55	0.14

Из табл. 2 видно, что показатели ошибок значительно ухудшились в связи с неправильным определением типа процесса, но тем не менее, модель Хольта–Винтерса корректно рассчитала моделируемые значения.

Заключение

Таким образом, полученные результаты эксперимента ещё раз доказывают необходимость определения типа процесса перед применением различных моделей прогнозирования. Также следует отметить, что модель Хольта–Винтерса показала наиболее точные показатели в обоих случаях. Это связано с тем, что модель Хольта–Винтерса относится к адаптивным методам прогнозирования. Сущность таких методов прогнозирования заключается в построении самокорректирующихся моделей, способных оперативно реагировать на изменение условий, что для конкретных данных численного эксперимента, которые представлены нестационарным временным рядом, оказалось наилучшим вариантом. Адаптивные методы учитывают различную информационную ценность уровней временного ряда, степень «устаревания» данных.

Процесс адаптации осуществляется итеративно с получением каждой новой фактической точки ряда. Модель постоянно «впитывает» новую информацию, приспосабливается к ней и поэтому наиболее точно отражает тенденцию развития, существующую в данный момент.

Список литературы

1. Гребенюк, Е. А. Методы анализа нестационарных временных рядов с неявными изменениями свойств / Е. А. Гребенюк // Автомат. и телемех. – 2005. – №12. – С. 3–30.

2. Носко, В. Н. Эконометрика. Кн. 2 Ч. 3, 4 : учебник / В. Н. Носко. – М. : Дело, 2011. – 259 с.

3. Моделирование динамики атмосферных процессов на основе анализа многомерных временных рядов / М. Г. Матвеев [и др.] // Воздушно-космические силы. Теория и практика. Электронное периодическое издание. – 2017. – №1 (1). – С. 98–110.

4. Открытый курс машинного обучения. Тема 9. Анализ временных рядов с помощью Python [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/ods/articles/327242/

Разработка системы автоматизированного парсинга расписания факультета с интеграцией в Telegram-бота

Д. К. Платов

Студент бакалавр

Е. Д. Проскуряков

Ассистент

Введение

В настоящее время автоматизация процессов сбора, обработки и представления данных является одним из ключевых направлений развития информационных технологий. Это обусловлено возрастающими объемами информации, необходимостью оперативного доступа к актуальным данным, а также требованиями к удобству их использования. Особую значимость приобретают системы, способные автоматически извлекать структурированные данные из облачных источников, таких как Google Таблицы, и интегрировать их в пользовательские сервисы.

Современные системы организации учебного процесса сталкиваются с проблемой эффективного представления данных для конечных пользователей. При использовании облачных таблиц для публикации расписаний возникает существенная сложность – необходимость работы с избыточными объемами информации. Студент, получая доступ к общему расписанию, вынужден самостоятельно вычленять из массива данных, содержащего информацию для всех групп и курсов, только те сведения, которые относятся лично к нему. Такой подход не только увеличивает временные затраты на поиск нужной информации, но и создает риск ошибок при интерпретации данных. Данная работа направлена на создание системы, которая не просто извлекает данные из таблиц, но и трансформирует их в удобный для студентов формат, устраняя информационный шум и обеспечивая быстрый доступ к релевантным учебным данным.

1. Выбор средств реализации

Для разработки системы был выбран язык программирования Python благодаря его эффективности при обработке данных, богатой экосистеме библиотек и доступности учебных материалов [1]. Язык

© Платов Д. К., Проскуряков Е. Д., 2025

обеспечивает простую интеграцию с различными API и быстрое прототипирование.

Хранение данных организовано на SQLite – легковесной файловой СУБД, не требующей отдельного сервера. Это решение обеспечило надежное хранение информации при минимальных накладных расходах.

Для работы с Telegram API использовалась библиотека telebot [2], предоставляющая удобную абстракцию для создания ботов. Ее выбор обусловлен простотой интеграции, хорошей документацией и поддержкой всех необходимых функций.

Такая комбинация технологий позволила создать оптимальное по производительности и простоте поддержки решение.

2. Разработка архитектуры системы

Система реализована как монолитное приложение. Несмотря на единую кодовую базу, архитектура организована вокруг трёх ключевых логических сервисов, каждый из которых отвечает за определённую функциональную область:

1. Bot Service выступает центральным узлом системы, отвечающим за взаимодействие с конечными пользователями через Telegram API. Этот сервис реализует полный цикл обработки пользовательских взаимодействий. Сервис выполняет следующие ключевые операции: прием и маршрутизацию запросов к специализированным обработчикам на основе анализа контекста; динамическое формирование ответов и финальную отправку подготовленных сообщений через Telegram API. Для обработки составных запросов реализован механизм обработки. Все компоненты сервиса интегрированы в единый конвейер обработки, что обеспечивает высокую отзывчивость и стабильность работы системы.

2. Parser Service представляет собой специализированный модуль для работы с данными расписания, обеспечивающий взаимодействие с Google API для своевременного обновления информации и извлечения необходимых данных из таблиц. Сервис поддерживает работу с Excel-таблицами Google Sheets, осуществляя их загрузку и последующую обработку.

Основная функция модуля – поиск данных по заданным параметрам, включая номер курса, конкретную учебную группу и номер подгруппы. После получения сырых данных сервис выполняет их нормализацию, группирует занятия по дням недели и преобразует информацию в структурированный формат, удобный для дальнейшего использования.

3. Database Service выполняет роль промежуточного слоя между основными компонентами системы и хранилищем данных, обеспечивая

надежное и безопасное взаимодействие с базой данных SQLite. Этот сервис инкапсулирует всю логику работы с данными, предоставляя другим модулям системы простой и понятный интерфейс для выполнения операций CRUD (Create, Read, Update, Delete). Для его реализации была использована библиотека sqlite3 [3].

Для обеспечения надежности модуль использует параметризованные запросы, защищающие от SQL-инъекций. Простота и эффективность Database Service позволяют другим компонентам системы быстро получать необходимую информацию без прямого взаимодействия с базой данных.

Таким образом, архитектура системы, представленная на рис. 1, обеспечивает четыре основных преимущества: высокую скорость разработки, простоту тестирования, эффективное использование ресурсов и лёгкость развёртывания.

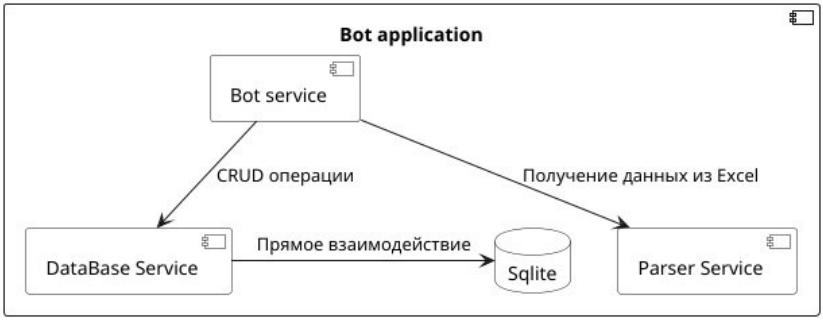


Рис. 1. Архитектура системы

3. Получение данных из таблиц

Для обеспечения высокой производительности обработки большого количества запросов Parser Service реализует двухэтапный алгоритм [4] обработки учебного расписания. На первом этапе система определяет положение целевых данных в табличной структуре, анализируя заголовки столбцов с учётом заданных параметров выборки. Для корректной работы с подгруппами предусмотрено смещение при поиске нужного столбца.

На втором этапе осуществляется извлечение информации по конкретному дню. Алгоритм использует фиксированный шаблон расположения данных, где каждый учебный день представлен определённым блоком строк. Обработка учитывает чередование недель и формирует структурированный результат, связывающий временные интервалы с соответствующими занятиями.

Особенностью реализации является адаптация под таблицы с объединёнными ячейками и жёстко заданной структурой. Система предполагает определённое форматирование исходных данных, включая:

– Фиксированное положение заголовков;
– Стандартизированное распределение дней недели;
– Чёткое разделение информации по неделям;

Модуль генерирует исключения при невозможности найти запрошенные данные, что позволяет внешним системам корректно обрабатывать случаи отсутствия расписания.

Ключевой технической сложностью при реализации парсера стала обработка объединённых ячеек в таблицах. Используемая библиотека openpyxl [5] сохраняет данные только в первую ячейку объединённого диапазона, оставляя остальные элементы пустыми. Для корректного чтения информации был разработан специальный механизм проверки.

Алгоритм работы с объединениями реализует следующий подход:

1. При анализе каждой ячейки система сначала определяет её принадлежность к объединённому диапазону через предоставляемый openpyxl двумерный массив слияний;
2. Если ячейка является частью объединения, модуль автоматически извлекает значение из первичной (корневой) ячейки диапазона;
3. Для обычных ячеек применяется стандартное чтение данных;

Такой подход обеспечивает:

– Полноту получаемой информации независимо от позиции ячейки в объединении;
– Сохранение целостности данных при любом форматировании таблицы;
– Единообразие обработки как объединённых, так и одиночных ячеек;

Такое решение полностью устраняет проблему "пустых" ячеек в объединённых диапазонах и обеспечивает точное чтение данных.

4. Бизнес-логика обработки запросов

Telegram-бот использует механизм polling для постоянного мониторинга входящих сообщений через Telegram API. Когда поступает новое сообщение, бот анализирует его содержание и ищет соответствующий обработчик среди зарегистрированных команд. Если подходящий обработчик не обнаружен, система генерирует и отправляет пользователю подробное руководство, содержащее список всех доступных команд с примерами их использования. В случае успешного распознавания команды, бот переходит к ее обработке.

На рис. 2 представлена архитектура процесса формирования ответов на запросы расписания, включающая следующие взаимосвязанные этапы обработки данных:

1. Сбор пользовательских данных: бот обращается к сервису Database для получения необходимой информации о пользователе и типе недели. Запрашиваются основные идентификационные данные: номер курса, группы, подгруппы пользователя.

2. Извлечение расписания: полученные параметры передаются в Parser Service для дальнейшей обработки. Данный сервис сначала определяет точное расположение нужных данных в таблице расписания. После этого выполняется точная фильтрация данных согласно заданным критериям, и в результате формируется структурированный набор занятий, готовый к последующему форматированию.

3. Подготовка и отправка ответа: на этом этапе полученные данные проходят окончательное форматирование. Выполняется логическая группировка занятий по дням недели для удобства восприятия. В завершение добавляется вспомогательная информация, такая как тип текущей учебной недели, что позволяет пользователю лучше ориентироваться в предоставленных данных.

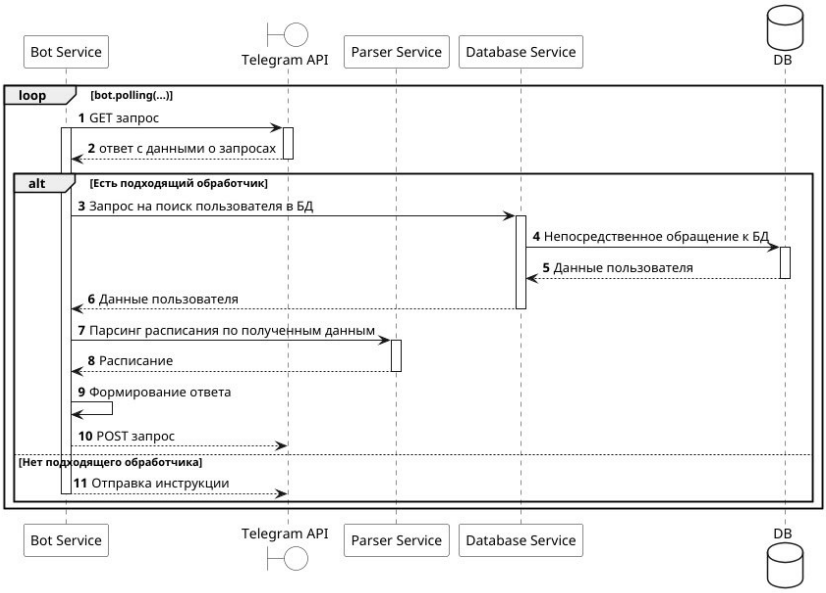


Рис. 2. Последовательность обработки запроса на расписание

Заключение

Данная статья представляет результаты разработки программного комплекса для автоматизированного предоставления учебного расписания через Telegram-бота. В работе подробно рассмотрены принципы проектирования системы, включая архитектурные решения и алгоритмы обработки запросов. Основное внимание уделено созданию эффективного инструмента для парсинга и распределения актуальной информации о занятиях.

Практическая ценность разработки заключается в значительном упрощении доступа к учебному расписанию. Система исключает необходимость ручного поиска информации в общих таблицах, предоставляя каждому пользователю только релевантные данные в удобном формате. Интеграция с популярным мессенджером обеспечивает мгновенный доступ к расписанию с любого устройства, что особенно важно для мобильных пользователей.

Практическая реализация проекта была осуществлена на факультете компьютерных наук Воронежского государственного университета (ВГУ). В настоящее время система успешно функционирует, обслуживая преимущественно студентов первого курса.

Ключевые показатели внедрения:

- Среднесуточная активность: 100+ пользователей;
- Время бесперебойной работы: 2 месяца (на момент публикации);
- Среднее время отклика: около 1 секунды;

Список литературы

1. Добрый, добрый Python с Сергеем Балакиревым [Электронный ресурс] : сайт. – Режим доступа: https://stepik.org/course/100707/syllabus

2. Pytba: Python Telegram Bot API [Электронный ресурс] : сайт. – Режим доступа: https://pytba.readthedocs.io/ru/latest/index.html

3. SQLite3: DB-API 2.0 interface for SQLite databases [Электронный ресурс] : сайт. – Режим доступа: https://docs.python.org/3/library/sqlite3.html

4. Бхаргава, А. Грокаем алгоритмы : иллюстрированное пособие для программистов и любопытствующих / А. Бхаргава ; 2-е изд., испр. и доп. – Санкт-Петербург : Питер, 2025. – 288 с.

5. Openpyxl: Python library to read/write Excel files [Электронный ресурс] : сайт. – Режим доступа: https://openpyxl.readthedocs.io/en/stable

Информационная система 3D-сканирования раневых поверхностей

И. А. Плотников

Студент бакалавр

Е. А. Деревянко

Студент бакалавр

Н. А. Ерофеева

Студент бакалавр

А. В. Максимов

Старший преподаватель

Введение

Современная медицина уделяет значительное внимание вопросам эффективного лечения ран, включая хронические, послеоперационные и травматические поражения кожи. Одним из ключевых аспектов успешного лечения является объективный мониторинг регенеративных процессов, позволяющий оценивать динамику заживления, корректировать терапию и прогнозировать сроки восстановления тканей. На сегодняшний день в клинической практике для измерения площади раневой поверхности наиболее широко применяются простые и доступные методы, такие как использование градуированной линейки или прозрачных трафаретов. Эти подходы обеспечивают быстроту измерений, однако обладают рядом существенных недостатков: низкой точностью, субъективностью интерпретации данных, а также невозможностью корректно оценивать площадь ран сложной геометрической формы.

Целью данной работы является разработка системы визуализации изображений со стереокамер после обработки моделями машинного обучения. Информационная система, объединяющая в себе функционал для сканирования, обработки и предоставления готовых данных об изменении площади поверхности с раной, значительно упростит жизнь медицинских работников.

© Плотников И. А., Деревянко Е. А., Ерофеева Н. А., Максимов А. В., 2025

1. Постановка задачи

Постановка задачи в рамках данной статьи звучит следующим образом: требуется разработать систему для вычисления и фиксирования площади поверхности раны. Были выдвинуты следующие требования:

- Разработать функционал для получения снимков со стереокамеры в формате PLY;
- Разработать функционал удаления со снимка областей, не являющихся раневыми;
- Разработать функционал восстановления поверхности трехмерных объектов;
- Разработать систему визуализации трехмерных объектов, полученных со стереокамеры.

2. Средства реализации программного продукта

Система для 3D сканирования раневых поверхностей написана на языке Python, с использованием следующих технологий:

- Open3D – это библиотека для работы с 3D-данными, поддерживающая визуализацию, обработку и анализ трехмерных облаков точек, мешей и объемных данных.
- PyRealsense2 – Python-обёртка для Intel Realsense SDK.
- U-net для сегментации раневых поверхностей
- OpenGL – спецификация, определяющая независимый от языка программирования программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику.

3. Реализация логики программного продукта

Разработка системы для 3D сканирования раневых поверхностей может быть условно разделена на три части:

1. работа со стереокамерами и PLY файлами;
2. сегментация полученных изображений;
3. разработка пользовательского интерфейса для взаимодействия с конечным результатом обработки PLY файлов и преобразования их в формат 3D-объектов.

На рис. 1 представлен пример получаемого 3D-объекта с камеры d430l от компании Intel.

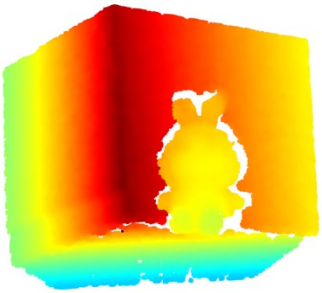


Рис. 1. Объект, полученный после снимка на камеру Intel d430l

Полученный документ является воксельной моделью. Его необходимо преобразовать в полигональный, для более удобного подсчета площади. Для начала переведем его в формат PLY.

main.py 1.ply 3z16.ply

```
ply
format ascii 1.0
comment pointcloud saved from Realsense Viewer
element vertex 246961
property float32 x
property float32 y
property float32 z
property float32 nx
property float32 ny
property float32 nz
property uchar red
property uchar green
property uchar blue
element face 475426
property list uchar int vertex_indices
end_header
-1.87219 1.74933 -4.246
0 0 0
63 0 0
-1.81365 1.70113 -4.129
0 0 0
65 0 0
```

Рис. 2. Пример PLY-файла

Следующим этапом обучается модель и сегментируется изображение, как показано в [1–4].

1. Подготовка данных

– Сбор и аннотация данных: создается датасет изображений с раневыми поверхностями. Каждое изображение сопровождается бинарной маской:

– Маска: Пиксели раны (класс 1) vs. фон (класс 0).

– Предобработка: Нормализация значений пикселей, аугментация данных (сдвиги, повороты, изменения яркости) для увеличения разнообразия обучающей выборки

2. Обучение
– Разделение данных: 70% – тренировочный набор, 15% – валидационный, 15% – тестовый.
– Мониторинг: Отслеживание метрик (IoU, Dice score) для каждой задачи отдельно.
3. Визуализация
– Наложение маски на исходное изображение (цветовое выделение раны).

В результате получаем изображение, представленное на рис. 3, с выделенной областью раны.

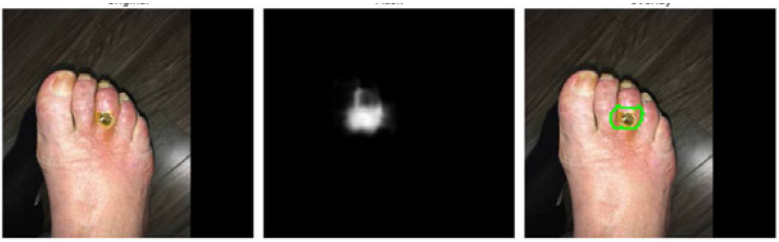


Рис. 3. Результат сегментации (1 изображение – оригинал, 2 – маска, 3 – результат наложения)

Параллельно разрабатывалась система визуализации файлов в PLY-формате. Для этого использовался модуль PyOpenGL, предоставляющий инструментарий технологии OpenGL, отраженный в [5], для языка Python. В отличие от во многом более удобного для данной задачи модуля Open3D, он позволяет создавать пользовательский интерфейс и работать не только с воксельным изображением, но и с облаком точек, а также создавать из него полноценную полигональную 3D модель. На рис. 4 представлен промежуточный вариант интерфейса в сравнении с Open3D.

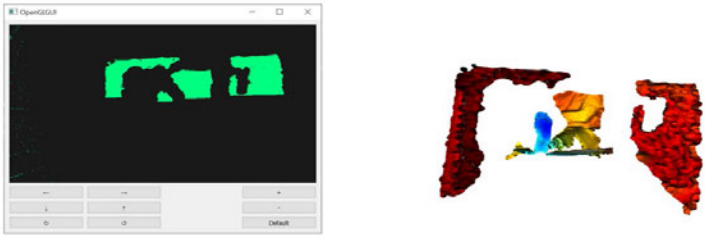


Рис. 4. Сравнение интерфейсов OpenGL (слева) и Open 3D (справа)

Можно обратить внимание, что воксельное изображение, с наложенными на него разными цветами и находящимся на сцене источником освещения, выглядит выигрышнее на фоне плоского облака точек OpenGL, однако следует заметить, что именно он дает возможность наложить оригинальное изображение на полигональную модель как текстуру, что повысит информативность изображения в разы.

Важно отметить, что при отображении объекта, с использованием средств Open 3D, функционал вращения, изменения масштаба и тому подобного доступен по умолчанию. Для нашей системы вращения объекта были реализованы при помощи модуля SciPy и кватернионов. Благодаря их плавной интерполяции система может быть портирована на устройства с сенсорным экраном. А также впоследствии можно полностью избавиться от интерфейсных кнопок для взаимодействия с объектом.

В полученных сегментированных трехмерных изображениях необходимо избавиться от пропущенных вершин. В работе использовались два способа восстановления поверхности трехмерных объектов – Poisson Surface Reconstruction и Alpha Shapes. Примеры работы этих алгоритмов представлены на рис. 5.

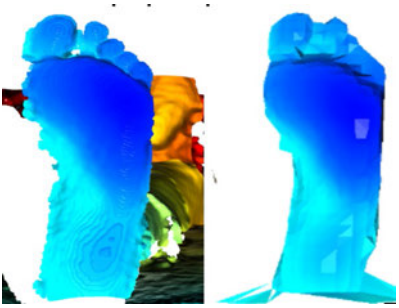


Рис. 5. Примеры работы алгоритмов восстановления трехмерных изображений – Poisson Surface Reconstruction слева и Alpha Shapes справа.

Заключение

В данной статье была рассмотрена реализация информационной системы 3D сканирования раневых поверхностей. Разработанные функциональные возможности соответствуют всем поставленным требованиям. В будущем планируется расширение функциональности данного программного продукта.

Список литературы

1. Ма, К. Трехмерное глубокое обучение на пайтон / К. Ма, В. Хегде, Л. Йольан. – ДМК-Пресс, 2023. – 226 с.

2. Learning for Medical Image Analysis / ed. by S. K. Zhou, H. Greenspan, D. Shen. – Academic Press, 2017.

3. Рашка, С. Машинное обучение с PyTorch и Scikit-Leam : пер. с англ. / С. Рашка, Ю. Лю, В. Мирджалили. – Астана : Фолиант, 2024. - 688 с.: ил. – ISBN 978-601-11-0034-2

4. Ronneberger, O. U-Net: Convolutional Networks for Biomedical Image Segmentation. / O. Ronneberger, P. Fisher, T. Brox // Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. – 2015. – Vol. 9351 – ISBN: 978-3-319-24573-7

5. Вольф, Д. OpenGL 4. Язык шейдеров. Книга рецептов / Д. Вольф ; пер. с англ. А. Н. Киселева. – Москва : ДМК Пресс, 2015. – 368 с.: ил. – ISBN 978-5-97060-255-3

Инструмент импорта задач из Polygon в Moodle

П. А. Путин

Студент бакалавр

Д. И. Соломатин

Старший преподаватель

Введение

В рамках дисциплин, посвященных изучению программирования, наиболее распространенной формой практических занятий являются лабораторные работы, в ходе которых студенты выполняют задачи, связанные с реализацией алгоритмов и структур данных. На факультете компьютерных наук Воронежского государственного университета существует ряд учебных дисциплин, использующих данную форму практического обучения, в частности, «Введение в программирование», «Язык программирования Си», «Алгоритмы и структуры данных», «Языки и системы программирования», «Язык программирования C++», «Информационные технологии» [1]. Проверка лабораторных работ по указанным дисциплинам предполагает проверку правильности выполнения программ путем последовательного тестирования на заранее подготовленных контрольных примерах.

В системе управления образовательными электронными курсами Moodle, используемой ВГУ, данная проверка может осуществляться с помощью плагина CodeRunner [2]. Но процесс подготовки тестовых данных в этом плагине является трудоёмким и требует значительных временных затрат из-за необходимости ручного заполнения.

На факультете функционирует факультатив по спортивному программированию, в рамках которого было создано более 200 обучающих задач по алгоритмам и структурам данных. Данные задачи были подготовлены на платформе Polygon – специализированном веб-приложении для подготовки задач для онлайн площадки решения задач и проведения соревнований по спортивному программированию Codeforces. Но задачи, созданные с помощью Polygon не совместимы с CodeRunner, что создает дополнительные трудности в их использовании в учебном процессе, так как требует доступа к Codeforces для запуска этих задач.

© Путин П. А., Соломатин Д. И., 2025

Таким образом, существует проблема интеграции задач, разработанных на платформе Polygon, в учебный процесс с использованием системы Moodle и плагина CodeRunner. В данной работе рассматривается задача разработки инструмента импорта задач из Polygon в Moodle.

1. Постановка цели

На данный момент для того, чтобы перенести задачу из системы Polygon в Moodle необходимо вручную переписать условие, решение и тестовые примеры в плагин CodeRunner. Для оценки эффективности ручного подхода было проведено испытание, результаты которого приведены в табл. 1. Расчёт времени, затрачиваемого на подготовку теста производился с учётом того, что на обработку одного теста в среднем уходит 15 секунд. Согласно приведенным данным, рассмотренный процесс занял в среднем 24 минуты с доверительным интервалом 8 минут, что было признано чрезмерно длительным.

Таблица 1

Оценка эффективности ручного преобразования задач

Задача	**Время, мин**	**Время на условие, мин**	**Количество тестовых примеров**
atomic-heart-puzzle	19,12	11,62	30
curtains2023	59,88	4,88	220
data-format	6,58	2,33	17
dummycheckersjan2023	30,34	2,84	110
integers-sum	19,93	2,43	70
linked-list	12,21	3,71	34
rockerarm2023	31,17	3,42	111
sort-table	15,59	3,09	50
Итог	**$24{,}35 \pm 8{,}22$ мин**		

Целью проекта стало сокращение времени переноса задач из платформы Polygon в систему Moodle таким образом, чтобы среднее время составляло менее одной минуты. Данный критерий проверяется по заранее предоставленному набору задач в формате Polygon.

Разработанный инструмент позволит снизить нагрузку на преподавателей, связанную с проверкой лабораторных работ, а студентам быстрее получать обратную связь по их решению. Задачи на продвинутые темы алгоритмов и структур данных могут быть использованы для углубленного обучения способных студентов, претендующих на высокий балл. Также использование банка задача,

предоставленного факультативом по спортивному программированию снизит нагрузку, связанную с подготовкой тестов и аттестаций. На рис. 1 представлена контекстная диаграмма проекта.

Рис. 1. Контекстная диаграмма

2. Постановка задачи

Процесс конвертации можно автоматизировать благодаря тому, что имеются возможности экспорта задач из Polygon в формате ZIP-архива [3] и импорта задач в Moodle из XML файла [4].

Внутренняя структура задачи Polygon описана в файле problem.xml, содержащем метаинформацию (см. табл. 2).

Таблица 2

Используемое содержимое файла problem.xml

Название поля	Описание
/problem/names/name[value]	название задачи
/problem/judging/testset/time-limit	ограничение на время выполнения решения
problem/judging/testset/memory-limit	ограничение на используемую решением оперативную память
/problem/statements/statement [@type='application/x-tex'] [@language='russian']	условие задачи на русском языке
/problem/assets/solutions/solution [@tag='main']/source	путь до исходного кода решения задачи на одном из языков программирования
/problem/files/executables /executable/source	путь до иных исполняемых файлов
/problem/judging/testset	путь до расположения наборов тестовых данных

В директории tests хранятся файлы тестовых данных. По умолчанию, эта папка содержит только входные данные добавленных вручную тестов, а генерируемые тесты и ожидаемые ответы создаются на основе данных из problem.xml, описывающих команду, используемую для вызова генератора и его входные данные.

В директории solutions находятся файлы с исходным кодом решений задачи, которых может быть несколько, но обязательно должно присутствовать решение, помеченное в problem.xml как main, являющееся эталонным решением данной задачи. Оно используется для генерации ответов на входные данные тестов.

В директории statements содержится условие задачи в разных форматах: HTML, PDF, TeX.

Задачу по программированию можно импортировать в Moodle в виде XML файла, состоящего из 25 полей, обязательными из которых являются название, условие, тип задачи, эталонное решение, а также необходимо наличие хотя бы одного тестового примера. Остальные поля являются либо необязательными, либо могут быть заполнены значениями по умолчанию.

По итогу проведённого анализа было выяснено, что формат Polygon содержит все необходимые для формата Moodle данные: тестовые примеры возможно сгенерировать на основе эталонного решения, условие сформировать на основе TeX и HTML представления из ZIP-архива.

Таким образом для достижения поставленной цели разрабатываемый инструмент на основе ZIP-архива, экспортируемого из Polygon, должен создавать XML файл в формате, используемом для представления задач Moodle.

Инструмент было принято разрабатывать в виде веб-приложения, потому что это обеспечит портативность и удобство сопровождения.

3. Реализация

Для создания инструмента были выбраны Java, Spring Boot, Flutter, nginx, Docker, а также сервис для компиляции и запуска программ Job Engine [5].

Инструмент построен на архитектуре клиент сервер, при этом на серверной стороне работают два сервиса: по обработке ZIP-архива и запуску программ для генерации тестов. Так как среди дополнительных требований к проекту были возможность редактирования задачи перед экспортом и создание помимо XML документов YAML файлов с задачей, сервисы передают клиентской стороне извлечённую задачу в виде JSON ответа, сокращённый пример которого представлен в листинге, а затем на его основе формируются необходимые файлы.

Листинг

Пример сокращённого JSON ответа

```
{"problem": {
    "name": "Linked list",
    "timeLimitMillis": 2500,
    "memoryLimitMB": 256,
    "statement": "<p>Максим изучает алгоритмы и структуры
данных, в частности, сейчас он изучает двусвязные списки...",
    "images": [{
        "name": "LinkedList.png",
        "path": "/",
        "encoding": "base64",
        "base64Data": "iVBORw0KGgoAAAANSUhEUgAAAjMAAABSCAY..."
      }], "mainSolution": {
      "content": "#include <bits/stdc++.h>...",
      "language": "CPP"
    }, "generators": {
      "GeneratorYes": {
        "content": "#include \"testlib.h\"...",
        "language": "CPP"
      },
    }, "testCases": [{
        "stdin": "11\r\n1 2 1 2 3 2 1 2 1 2 3",
        "expected": "YES\n3\n1 2 3",
        "display": true
      }, ...]},
  "stdinGenerationErrors": [],
  "expectedGenerationErrors": []}
```

Процесс конвертации включает в себя извлечение метаинформации о задаче, преобразование текста условия из формата TeX в формат HTML, а также генерацию тестовых данных. При этом особое внимание уделено обработке ошибок, связанных с отсутствием в переданном архиве нужных данных. Для повышения производительности процесс генерации тестов выполняется в 8 потоках, что обеспечивает быстрое и надёжное выполнение без перегрузки сервиса Job Engine.

Структура классов сервиса обработки ZIP-архива, представленная на рис. 2, имеет 3 уровня: контроллер, обрабатывающий HTTP запросы; сервисы извлечения из архива данных о задаче и доступа к Job Engine; объект доступа к данным из ZIP-архива. Так как в архиве содержаться файлы с разнородной информацией: XML документы, текстовые файлы и картинки – для получения содержимого каждого из них используется один из экстракторов, наследуемых от общего абстрактного класса.

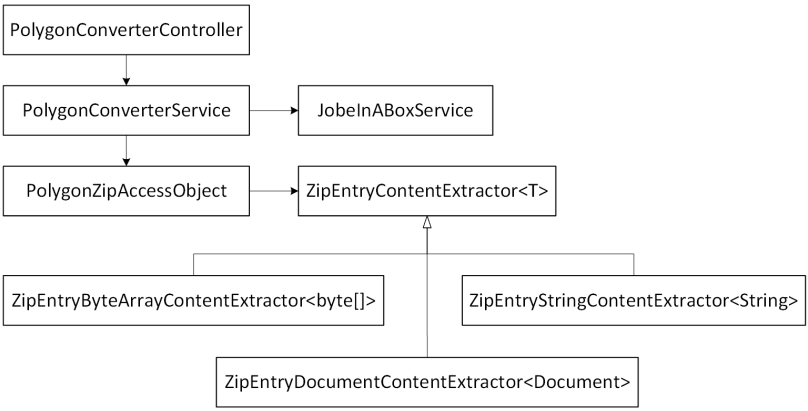


Рис. 2. Диаграмма ключевых классов системы

4. Интерфейс пользователя

Приложение реализовано в виде одностраничного сайта, разделённого на три блока: форму ввода, поле вывода и список ошибок (см. рис. 3). В поле вывода отображается конвертированная в XML задача, с возможностью скачать полученный файл. Форма ввода позволяет перед скачиванием файла внести при необходимости изменения в содержание задачи. Список ошибок отображает сообщения об ошибках, возникших при преобразовании задачи.

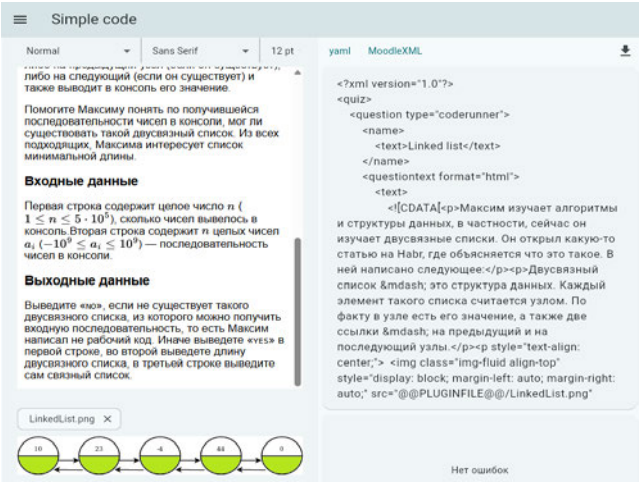


Рис. 3. Интерфейс инструмента конвертации

5. Тестирование

В ходе испытаний работоспособность разработанного инструмента подтвердилась – все выбранные задачи успешно перенесены в требуемый формат. На рис. 4 приведён пример преобразованной задачи, импортированной в систему Moodle:

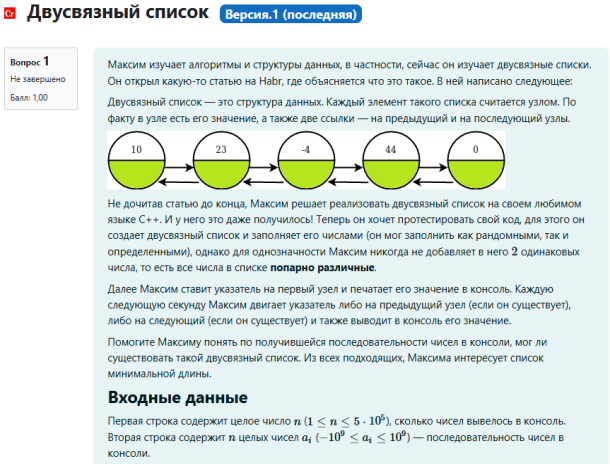


Рис. 4. Преобразованная задача в Moodle

Время конвертации задачи в среднем сократилось с 23 минут до 35 секунд, что отражено в табл. 3. Тестирование проводилось на компьютере с процессором 12th Gen Intel(R) Core(TM) i5-12400F 2.50 ГГц и 16 ГБ оперативной памяти на операционной системе Windows 11 Pro 24H2 26100.3775 с установленным Docker 28.0.4, build b8034c0.

Таблица 3

Оценка эффективности конвертации задач

Задача	**Время, с**
atomic-heart-puzzle	8,86
curtains2023	103,82
data-format	4,75
dummycheckersjan2023	49,37
integers-sum	36,87
linked-list	15,64
rockerarm2023	45,56
sort-table	22,08
Итог	**35,87 ± 15,86 с**

Заключение

Как итог, благодаря разработанному инструменту в 40 раз был ускорен процесс конвертации задач из Polygon в Moodle до среднего значения 35 секунд, что соответствует установленному критерию успеха.

Список литературы

1. Учебный план по программе бакалавриата 09.03.04 программная инженерия 2024 [Электронный ресурс]. – Режим доступа : https://edu.vsu.ru/pluginfile.php/7295641/mod_data/content/6048211/09.03.04_ПрогрИнженерия_ДО_ИСиСТ_2024%2B%2B.plx.xlsx.pdf
2. CodeRunner question type - MoodleDocs [Электронный ресурс]. – Режим доступа : https://docs.moodle.org/500/en/CodeRunner_question_type
3. Polygon End-User Documentation [Электронный ресурс]. – Режим доступа : https://polygon.codeforces.com/static/polygon.rtf?ccid=8f69e021af0883c224c5bc287058e878
4. Moodle XML format - MoodleDocs [Электронный ресурс]. – Режим доступа : https://docs.moodle.org/311/en/Moodle_XML_format
5. JOBE [Электронный ресурс]. – Режим доступа : https://github.com/trampgeek/jobe/tree/master

Применение SRE практик для построения отказоустойчивых систем

Е. А. Рогачев

Студент бакалавр

Д. И. Соломатин

Старший преподаватель

Введение

В условиях стремительного роста цифровых сервисов и увеличения объёмов пользовательского трафика обеспечение надёжности и доступности программных систем становится критически важной задачей. Современные приложения всё чаще представляют собой распределённые микросервисные архитектуры, в которых сбой одного компонента может привести к каскадным отказам всей системы. Для решения подобных задач инженеры по надёжности сайтов (Site Reliability Engineers, SRE) применяют инженерные подходы, ориентированные на систематическое улучшение стабильности, производительности и масштабируемости сервисов.

SRE — это инженерная дисциплина, возникшая в компании Google, целью которой является обеспечение высокой надёжности сервисов при сохранении темпов развития и выпуска новых функций. В основе подхода SRE лежит идея управления системой как программным продуктом, с применением автоматизации, мониторинга и измеримых целей надёжности — таких как SLA (соглашение об уровне обслуживания), SLO (цель уровня обслуживания) и SLI (индикаторы уровня обслуживания). Одной из ключевых задач SRE-инженера является построение систем, способных устойчиво функционировать даже в условиях сбоев — будь то ошибки в коде, перегрузка, отказ внешних зависимостей или непредвиденные пользовательские сценарии.

Для повышения отказоустойчивости SRE-практики предлагают использование специальных архитектурных паттернов устойчивости (resilience patterns), направленных на смягчение последствий сбоев и локализацию ошибок. Эти паттерны позволяют проектировать системы, способные восстанавливаться без внешнего вмешательства, эффективно

© Рогачев Е. А., Соломатин Д. И., 2025

использовать ресурсы и обеспечивать пользователю стабильный опыт взаимодействия даже в условиях частичной деградации.

В данной статье рассматриваются четыре наиболее широко применяемых паттерна из арсенала SRE-подхода:

- Retry pattern — повтор попытки операции в случае временного сбоя;
- Rate limiting — ограничение частоты запросов для защиты от перегрузки;
- Circuit breaker — автоматическое прерывание обращений к сбойному компоненту;
- Bulkhead — изоляция ресурсов для предотвращения каскадных отказов.

Целью статьи является анализ перечисленных паттернов и оценка их вклада в построение отказоустойчивых программных систем. [1]

1. Retry pattern

Retry pattern (паттерн повторной попытки) — это один из базовых методов повышения устойчивости к временным сбоям в распределённых системах. Его суть заключается в автоматическом повторении неудавшейся операции через заданный интервал времени, с надеждой на то, что сбой был временным и следующая попытка будет успешной.

Как показано на рис. 1, в случае временного сбоя первая попытка запроса может завершиться ошибкой, однако последующий ретрай через небольшую задержку оказывается успешным.

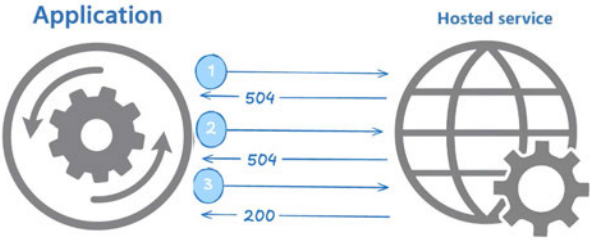


Рис. 1. Схема работы ретраев

Принцип работы паттерна заключается в следующем: при возникновении сбоя система не завершает операцию окончательной ошибкой, а инициирует повторную попытку. В промежутках между этими попытками может использоваться фиксированная задержка либо нарастающая — по экспоненциальной схеме. Количество попыток

ограничивается, чтобы не перегрузить систему — обычно устанавливается верхний предел в 3–5 повторов. Чтобы избежать одновременных ретраев от множества клиентов, может применяться так называемый jitter — небольшое случайное смещение времени между попытками. Повторяются только операции, при которых ошибка считается временной (например, ошибки сетевого соединения или HTTP-коды 502/503), а логические ошибки (например, 400 Bad Request) не ретраятся. Таким образом, ретрай-логика помогает повысить надёжность взаимодействия между сервисами без вмешательства пользователя [2].

Спекулятивные ретраи — это расширение классической схемы ретраев, которое предполагает запуск нескольких параллельных (или отложенных) копий одного и того же запроса, если первая попытка задерживается. Цель — сократить общее время отклика в ситуациях, когда возможно "подвисание" запроса, вызванное задержками на стороне сервиса или сети.

Как иллюстрируется на рис. 2, при превышении времени ожидания первой попытки, система запускает "спекулятивную" вторую попытку (обычно в другом экземпляре сервиса), не дожидаясь окончания первой. Успешный ответ от любой из копий принимается как результат, а остальные — отменяются или игнорируются.

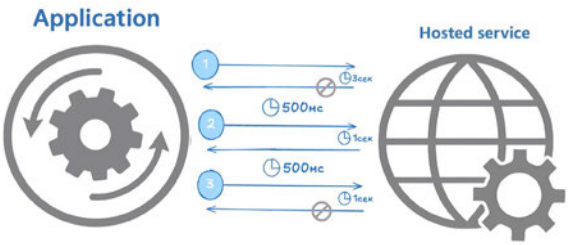


Рис. 2. Схема работы спекулятивных ретраев

2. Rate limiting

Rate limiting — это паттерн управления нагрузкой, направленный на ограничение количества операций, выполняемых системой или её компонентом за единицу времени. Этот механизм служит как средством защиты от перегрузки, так и инструментом контроля справедливого распределения ресурсов между клиентами. В системах, обрабатывающих большое количество внешних запросов (например, API), rate limiting становится необходимым элементом обеспечения

отказоустойчивости и стабильной работы. Схема работы паттерна продемонстрирована на рис. 3.

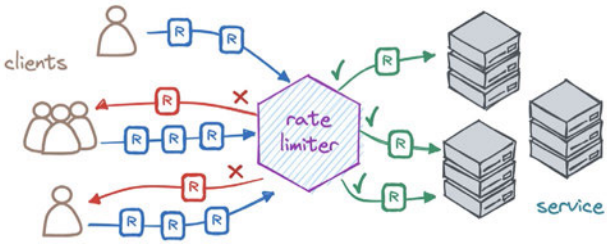


Рис. 3. Схема работы rate limiter

Принцип работы паттерна заключается в том, что для каждого клиента или группы клиентов система устанавливает лимит на число запросов в определённый временной интервал (например, 1000 запросов в минуту). При превышении лимита система либо отклоняет запрос (с кодом 429 Too Many Requests), либо помещает его в очередь, либо вводит искусственную задержку. Реализация может базироваться на различных алгоритмах, включая Token Bucket, Leaky Bucket, Fixed Window и Sliding Window. Эти алгоритмы различаются способом учёта запросов, допуском "всплесков" активности и адаптацией к равномерной или переменной нагрузке [3].

Например, в модели Token Bucket каждому клиенту выдаётся "ведро" с токенами, которые расходуются при каждом запросе. Токены пополняются с заданной скоростью. Если токены закончились — запрос блокируется или отклоняется. Это позволяет системе гибко контролировать и равномерно распределять нагрузку, допуская при этом кратковременные всплески активности.

На практике rate limiting реализуется на различных уровнях: в API-шлюзах, балансировщиках нагрузки, прокси-серверах, а также внутри отдельных микросервисов. Важно правильно выбрать уровень агрегации лимитов: на уровне IP-адресов, пользователей, клиентов или сессий — в зависимости от архитектуры и бизнес-логики.

3. Circuit breaker

Circuit Breaker — это паттерн, вдохновлённый аналогией с электрическим предохранителем. Его основная задача — прерывать поток запросов к зависимому компоненту, когда тот демонстрирует нестабильность или отказы, тем самым предотвращая каскадные сбои,

снижая нагрузку и способствуя более быстрому восстановлению системы. В рамках практик Site Reliability Engineering (SRE) этот механизм становится ключевым элементом стратегии устойчивости к отказам [4].

Как показано на рис. 4, паттерн circuit breaker представляет собой обёртку вокруг вызова к внешнему ресурсу, контролирующую прохождение запросов в зависимости от текущего состояния.

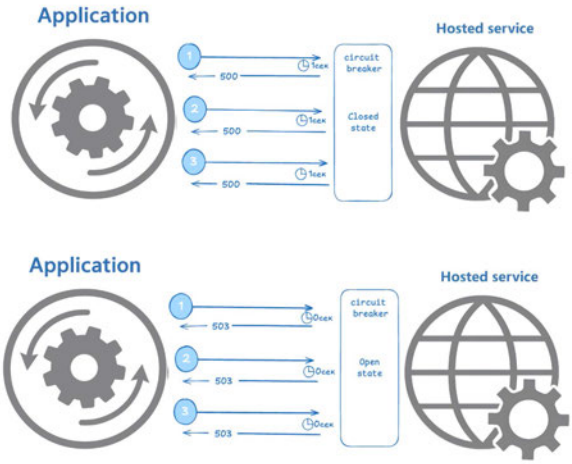


Рис. 4. Схема работы circuit breaker

Circuit breaker функционирует как автомат с тремя состояниями: закрытое (closed), открытое (open) и полуоткрытое (half-open). В нормальном режиме (закрытом) все запросы проходят к зависимому сервису. Система при этом отслеживает долю ошибок. Когда эта доля превышает заданный порог (например, 50% за последние N запросов), схема переходит в открытое состояние. В этом режиме все запросы немедленно отклоняются без попытки обращения к зависимому сервису, снижая нагрузку на систему.

Через определённый интервал времени (таймер восстановления) circuit breaker переходит в полуоткрытое состояние, в котором допускаются отдельные пробные запросы. Если они проходят успешно — компонент считается восстановленным, и схема возвращается в закрытое состояние. Если ошибки продолжаются — состояние снова становится открытым.

4. Bulkhead

Паттерн Bulkhead (в переводе — "переборка", термин из судостроения) получил своё название по аналогии с конструкцией корабля, где специальные перегородки предотвращают затопление всего судна при пробоине в одном отсеке. В системах, построенных по принципам Site Reliability Engineering, этот паттерн используется для изоляции компонентов — с целью минимизации последствий локальных сбоев и повышения устойчивости всей архитектуры.

На рис. 5 представлена схема, демонстрирующая принцип изоляции компонентов с помощью bulkhead, при которой каждая группа пользователей или задач работает в своём изолированном «отсеке» системы.

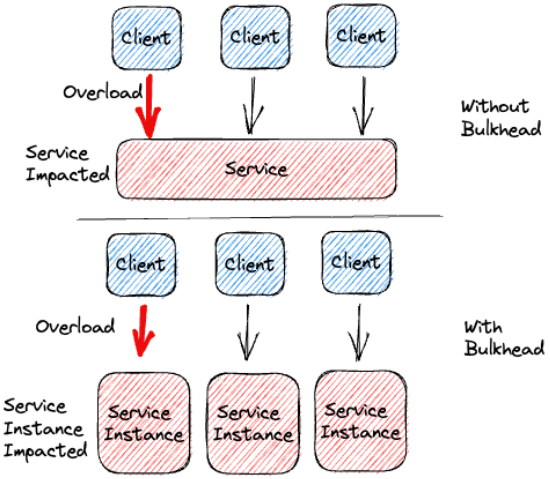


Рис. 5. Принцип изоляции компонентов с помощью bulkhead

Паттерн Bulkhead реализуется путём логического или физического разделения ресурсов между различными частями системы. Это может включать выделение отдельных пулов потоков для разных типов операций или клиентов, изоляцию сетевых соединений по направлениям или группам пользователей или разделение очередей сообщений или независимых задач на уровне обработки.

Таким образом, если одна часть системы начинает работать нестабильно (например, поток задач неожиданно возрастает и начинает «забивать» пул), это не влияет на другие части, у которых выделены собственные ресурсы. Например, сбой в обработке фоновых задач не

приведёт к отказу пользовательских HTTP-запросов, если они обслуживаются отдельным пулом потоков.

5. Результаты работы паттернов

Описанные паттерны отказоустойчивости были внедрены и протестированы в рамках реального проекта, работающего в продакшн-среде. Проект представляет собой REST приложение, взаимодействующее с несколькими базами данных. У сервиса также есть порядка 10 потребителей (внешних сервисов, посылающих запросы приложению) и помимо этого в самой логике сервиса настроена интеграция с другими системами. То есть в процессе выполнения внутренней бизнес логики приложение постоянно обращается для получения информации к сторонним системам, нескольким базам данных и отсюда следует, что при выходе из строя хотя бы одного компонента мы можем повлиять на всех потребителей сервиса и спровоцировать каскадный сбой. Именно поэтому было важно правильно спроектировать сервис для обеспечения его отказоустойчивости.

Для объективной оценки их эффективности применялся мониторинг с использованием Grafana, что позволило зафиксировать изменения в поведении системы до и после применения каждого из паттернов.

До внедрения паттерна Retry на графике (рис. 6) наблюдается резкий рост ошибочных ответов (5xx) от внешнего API при кратковременных сбоях в сети. Включение механизма повторных попыток с экспоненциальной задержкой позволило «пережидать» временные проблемы, тем самым повышая общее число успешных запросов, в результате чего всплески ошибок от кратковременных сбоев ушли и больше не повторялись.

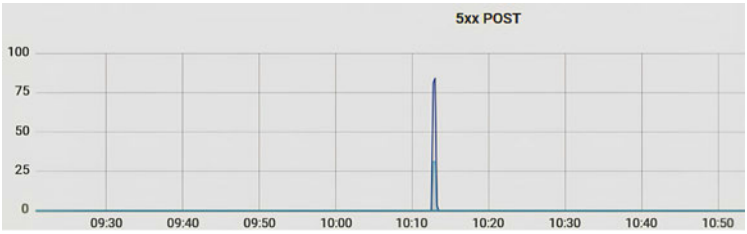


Рис. 6. 5xx ответы сервиса при кратковременных сбоях в сети до внедрения ретраев

В случае Rate Limiting на метриках отчётливо видно, как введение ограничения по количеству запросов помогло снизить нагрузку на критичные подсистемы. До применения лимитов происходили резкие всплески трафика, которые приводили к деградации производительности. После применения — частота входящих запросов стабилизировалась, а общая доступность сервиса выросла. Графики показывают, как система стала устойчивее к резким пикам, не теряя при этом основной функциональности. На рис. 7 виден всплеск 5xx ошибок в сервисе, после включения rate limiter лишняя нагрузка стала отбиваться 4xx ошибками (рис. 8), что помогло снизить нагрузку на сервис и быстрее ему восстановиться.

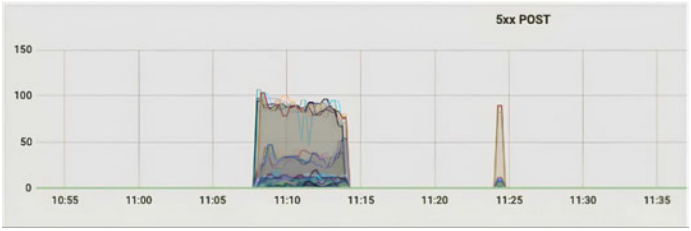


Рис. 7. 5xx ответы сервиса до и после включения rate limit

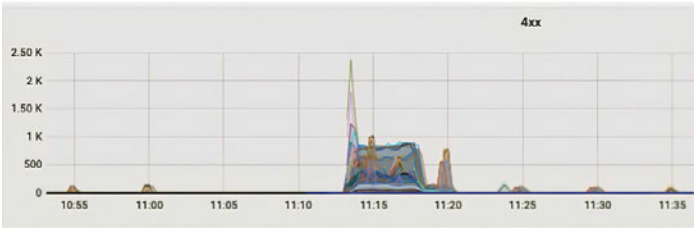


Рис. 8. 4xx ответы сервиса при включении rate limit

Результаты применения Circuit Breaker видны в виде резкого сокращения времени ответа и количества таймаутов в моменты нестабильной работы внешних сервисов. Ранее, при затяжных зависаниях зависимостей, весь сервис начинал деградировать. После активации Circuit Breaker неудачные вызовы оперативно блокируются, предотвращая накопление ошибок и освобождая ресурсы для других операций. На рис. 9 показан переход схемы в состояние open, которое спровоцировано резким скачком 5xx и 4xx ошибок (рис. 10 и рис. 11). Фиолетовые полоски – это интервалы времени, когда circuit breaker был в закрытом состоянии, а после всплеска ошибок можно видеть, как он

переходит в открытое (бирюзовые полоски), после чего, когда сервис начинает восстанавливаться, находится в промежуточном полуоткрытом состоянии (синяя полоска). При этом благодаря переключению circuit breaker сервис смог восстановиться и не потерял работоспособности. [5]

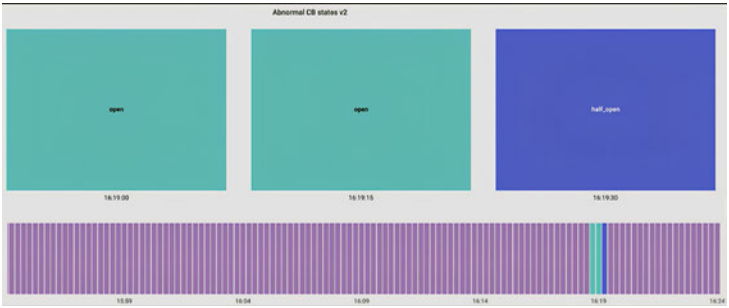


Рис. 9. Состояния circuit breaker во время сбоя

Рис. 10. Всплеск 5xx ошибок, спровоцировавший переход circuit breaker в состояние open

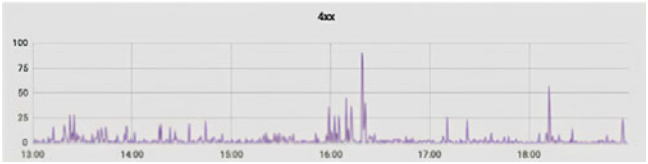


Рис. 11. Всплеск 5xx ошибок, спровоцировавший переход circuit breaker в состояние open

Применение паттерна Bulkhead в проекте позволило избежать полного отказа при перегрузке одной из частей. Ранее все типы задач использовали общий пул потоков, и при росте очереди фоновых задач задерживались и пользовательские операции. После внедрения изоляции (были выделены отдельные пулы потоков под сервисы работающие с разными базами данных) даже при росте нагрузки в одном сегменте, остальные продолжают работать стабильно и без деградации. То есть при конкуренции на одной базе данных или при ее

отказе, сервисы работающие с другими базами продолжают нормально функционировать. Это подтвердило эффективность паттерна в условиях реального отказа.

Эти данные подтверждают, что правильное применение описанных решений оказывает существенное положительное влияние на надёжность сервисов.

Заключение

Построение отказоустойчивых систем — одна из ключевых задач, стоящих перед инженерами, практикующими подход SRE. Паттерны Retry, Rate Limiting, Circuit Breaker и Bulkhead позволяют справляться с нестабильностью внешних зависимостей, ограничивать избыточную нагрузку, предотвращать каскадные сбои и локализовывать отказ в пределах одного компонента. Их совместное применение создаёт надёжную основу для построения устойчивой, предсказуемой и управляемой системы. В условиях постоянно возрастающей сложности распределённых архитектур, именно такие инженерные решения становятся залогом стабильной работы сервисов.

Список литературы

1. Google. Site Reliability Engineering [Электронный ресурс]. – Режим доступа: https://landing.google.com/sre/sre-book/toc/
2. Designing for Fault Tolerance in AWS [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/architecture/fault-tolerance/
3. Rate Limiting with NGINX [Электронный ресурс]. – Режим доступа: https://www.nginx.com/blog/rate-limiting-nginx/
4. Resilient design with the Circuit Breaker pattern [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/en-us/azure/architecture/patterns/circuit-breaker
5. Circuit Breaker Pattern [Электронный ресурс]. – Режим доступа: https://martinfowler.com/bliki/CircuitBreaker.html

Информационная система расчёта вероятностей совпадения нуклеотидных последовательностей в задачах биоинформатики

В. В. Рычков

Студент специалист

А. Н. Кондрашов

Студент бакалавр

Я. А. Туровский

Профессор

Введение

Одной из задач биоинформатики является поиск вероятностей встречаемости фрагментов одной нуклеотидной последовательности в другой нуклеотидной последовательности с целью предположения наличия механизмов удаления исследуемых фрагментов из нуклеотидной последовательности либо механизма накопления исследуемых фрагментов в нуклеотидной последовательности [1].

При этом с учетом особенности ДНК важной является процедура выравнивания. Алгоритмы выравнивания – это биоинформатические методы, которые позволяют размещать две или более последовательности мономеров ДНК, РНК или белков друг под другом таким образом, чтобы можно было легко увидеть сходные участки в этих последовательностях [2]. Выравнивание последовательностей – это процедура сравнения двух (попарное выравнивание) или нескольких (множественное выравнивание) последовательностей путём поиска рядов отдельных элементов или характерных комбинаций элементов последовательностей, которые расположены в выравниваемых последовательностях в одинаковом порядке.

Алгоритм Нидлмана–Вунша – это алгоритм для выполнения выравнивания двух последовательностей, который используется в биоинформатике при построении выравниваний аминокислотных или нуклеотидных последовательностей [3].

Суть алгоритма в следующем. По матрице расстояний между аминокислотами (или, соответственно, между нуклеотидами)

© Рычков В. В., Кондрашов А. Н., Туровский Я. А., 2025

итеративным образом рассчитывается матрица всех возможных маршрутов

```
*Sij*: Sij=Dij+max(Si-1,j-1,max k<j-1 (Si-1,k - G),max k<i-1
(Sk,j-1-S)),
```

где:

S_{ij} – элемент i-й строки j-го столбца,

D_{ij} – расстояние между i-й и j-й аминокислотами (или нуклеотидами),

G – штраф на делецию.

Затем осуществляется проход по матрице в обратном направлении, по максимальным элементам. Полученный маршрут соответствует оптимальному выравниванию. В качестве матрицы минимальных расстояний между аминокислотами обычно используется матрица минимальных мутационных расстояний по генетическому коду между аминокислотами, но могут использоваться и другие меры [4].

1. Программа StringsAligner

Загрузка короткого файла

Для это надо нажать на кнопку главной формы «Короткий файл».

После чего указать в файловом диалоге путь до файла и нажать «Открыть». Допускаются только файлы с расширением .txt.

В момент загрузки в файле допускается наличие любых символов.

Символы пробела, табуляции, переноса строки и прочие не видимые символы удаляются из загруженных данных.

Все остальные символы по возможности приводятся к верхнему регистру.

Все эти строки приводятся к единому представлению: «AGCTTA»

Загрузка длинного файла

Для это надо нажать на кнопку главной формы «Длинный файл».

После чего указать в файловом диалоге путь до файла и нажать «Открыть». Допускаются только файлы с расширением txt.

В момент загрузки в файле допускается наличие любых символов.

Символы пробела, табуляции, переноса строки и прочие не видимые символы удаляются из загруженных данных.

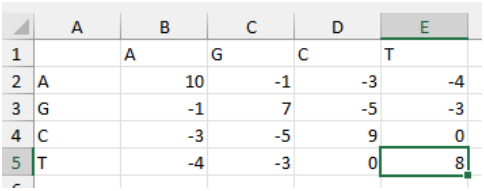
Все остальные символы по возможности приводятся к верхнему регистру.

Все эти строки приводятся к единому представлению: «AGCTTA»

Загрузка матрицы сравнений

Для это надо нажать на кнопку главной формы «Файл сравнений».

После чего указать в файловом диалоге путь до файла и нажать «Открыть». Допускаются только файлы с расширением xlsx (Лист Microsoft Excel).



	A	B	C	D	E
1		A	G	C	T
2	A	10	-1	-3	-4
3	G	-1	7	-5	-3
4	C	-3	-5	9	0
5	T	-4	-3	0	8

Рис. 1. Пример правильного оформления файла

Выбор папки сохранения результатов

Для это надо нажать на кнопку главной формы «Файл сравнений».

После чего указать в файловом диалоге путь до папки и нажать «Выбрать».

Проведение предобработки данных

Происходит после того, как были успешно загружены файл с короткой последовательностью, файл с длинной последовательностью и файл с матрицей сравнений. О процессе предобработки сигнализирует появление обновляющейся строки «Расчёт…» и неактивность пяти кнопок.

По завершении предобработки кнопки снова становятся активны.

Результаты обработки

Все результаты сохраняются в папку, которая была указана перед началом обработки.

Всего есть 5 видов файлов:

– Файл с входными параметрами алгоритма выравнивания
– Файлы выравниваний – в количестве показанных перед обработкой промежутков выравнивания
– Файл статистики
– Файл биномиального распределения
– Файл с указанием количества совпадений

Файл с входными параметрами алгоритма выравнивания

Обозначается как «Входные параметры алгоритма выравнивания.xlsx»

Указывает все параметры, указанные перед запуском алгоритма выравнивания, в том числе конфигурацию, загруженную из файла или использованную по умолчанию.

Файлы выравнивания

Представлены в количестве, соответствующем количеству показанного перед началом работы алгоритма количества промежутков выравнивания.

Обозначаются как «<ИНДЕКС_НАЧАЛА_ВЫРАВНИВАНИЯ_ПО_ДЛИННОЙ_ПОСЛЕДОВАТЕЛЬНОСТИ>.txt»

На первой строке записано выравнивание короткой последовательности, на второй строке – выравнивание промежутка из длинной последовательности

Файл статистики

Обозначается как «stats.xlsx»

– Данный файл содержит в себе краткую информацию по каждому из промежутков выравнивания:

– Индекс начала свертки – индекс в длинном массиве, начиная с которого считалось количество совпадений символов с коротким массивом.

– Количество совпадений – общее количество совпавших символов на конкретном промежутке свертки.

– Доля совпадений – отношения количества совпавших символов на промежутке выравнивания к длине промежутка вырывания (что равно длине короткой последовательности).

– Биномиальная вероятность – дискретное значение вероятности получения данного количества совпадений при $n =$ <длина промежутка выравнивания> и $p = 1$ <Количество названий символов в матрице выравниваний>.

– Количество пропусков по малому массиву – общее количество пропусков, возникшее при использовании алгоритма выравнивания в коротком массиве и обозначенных в файле выравниваний как «-».

– Количество пропусков по участку большого массива – общее количество пропусков, возникшее при использовании алгоритма выравнивания на промежутке из большого массива, по которому проводилась свёртка, обозначенных в файле выравниваний как «-».

– Длина короткого массива после обрезки – длина короткого файла после выравнивания без учета символов «-» в начале и в конце выровненной последовательности.

Файл биномиального распределения

Обозначается как «Биномиальное распределение.xlsx»

Содержит в себе коэффициенты биномиального распределения для

$$dist_x = C_n^x p^x (1-p)^{n-x},$$

n – длина промежутка выравнивания, p – количество символов в матрице сопоставлений и $X = \overline{0,n}$.

Стоит заметить, что при значениях длины короткой последовательности более нескольких сотен элементов в начале и конце файла с биномиальных распределением появляется элементы, равные 0. Связано это с ошибкой округления компьютера при расчете коэффициента [5].

Файл с указанием количества совпадений

В данном файле указаны количества совпадений при свертке короткой и длиной последовательности в каждой возможной точке.

Таким образом мы можем определить маргинальные частоты встречаемости последовательности, то есть фрагменты последовательности, которые встречались не только чаще чем это ожидается при случайном совпадении, но и статистически значимо реже чем это могло бы быть, если бы фрагменты последовательности совпадали случайно [6].

Специальное значение количества совпадений «-1» обозначает, что на данном промежутке выравнивания в длинной последовательности были встречены символы, которых не было среди символов в матрице сопоставлений, вследствие чего данный промежуток не участвовал в выравнивании.

2. Программа поиска маргинальных частот

Описание работы программы

Указание пороговых значений для нахождения значений маргинальных частот в полях «Нижний порог», «Верхний порог».

Указание исходного файла. Исходным файлом для этой программы является «Файл с указанием количества совпадений», получаемый из программы StringsAligner.

Кнопка «Выбор директории» позволяет выбрать папку, в которой находится несколько файлов.

Кнопка «Выбор файла» позволяет выбрать конкретный файл.

Результаты обработки

Все результаты сохраняются в папку, которая была указана при выборе исходного файла. Результат представлен в одном файле.

3. Программы-парсера данных из базы National Library of Medicine (NCBI)

Программа-парсер данных из базы National Library of Medicine (NCBI) предназначена для автоматического переноса данных, полученных по запросу пользователя, на локальный диск и их организации в виде базы данных.

Принцип работы программы начинается с формирования пользователем запроса на официальном сайте NCBI (https://www.ncbi.nlm.nih.gov/nuccore/). Пользователь копирует полученную ссылку и вставляет её в окно ввода программы. После этого парсер автоматически обрабатывает запрос, загружая все связанные данные и сохраняя их на локальном диске.

Автоматизированный процесс включает следующие этапы (рис. 2, 3):

– Запрос данных пользователем на сайте NCBI.
– Вставка ссылки в интерфейс программы.
– Автоматическая загрузка данных парсером.
– Сохранение данных на локальном диске.
– Организация данных в локальную базу данных.

Структура базы данных

Сохраненные данные организуются в виде файла формата .csv. В этом файле записаны:

– Уникальное имя файла, состоящее из основной части названия и хешированного полного имени.
– Иерархия таксонов, расположенных в порядке убывания.
– Местоположение последовательности нуклеотидов, сохранённой в формате .txt.

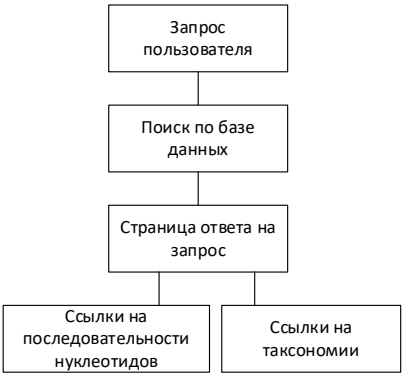


Рис. 2. Схема запроса к базе данных на сайте NCBI

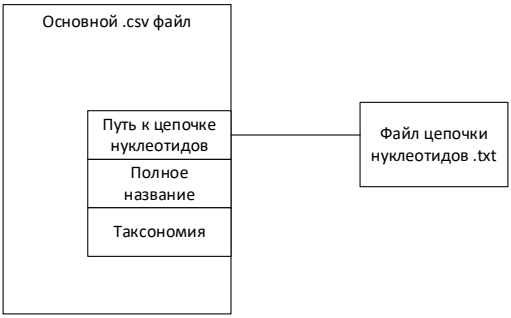


Рис. 3. Схема реализованной базы данных

Таким образом, разработанная программа-парсер автоматизирует процесс загрузки данных с базы NCBI и их организацию в виде локальной базы данных. Это снижает трудозатраты пользователя и обеспечивает удобный доступ к полученной информация [7].

Заключение

В рамках поставленной задачи было выявлено, что теоретическая встречаемость маргинальных частот ниже, чем действительная. Это может быть связанно со встраиванием фрагментов генома вируса клещевого энцефалита в геном животных-резервуаров.

Также были получены сведения о необходимых вычислительных мощностях для оптимального времени исследования. На время обработки данных влияет:

1. Эффективность процессора в обработке данных.
2. Объём оперативной памяти.

Исходя из данных, полученных во время работы, можно сделать вывод о том, что скорость обработки данных, в первую очередь, зависит от характеристик процессора, во вторую, от объёма оперативной памяти.

Был реализован проект, позволяющий выявлять встречаемости последовательностей ниже, чем при их случайном сравнении.

Список литературы

1. Огурцов, А. Н. Основы биоинформатики : учеб. пособие / А. Н. Огурцов. – Харьков : НТУ «ХПИ», 2013. – 400 с. – На рус. яз. ISBN 978-617-05-069-4
2. Компо, Ф. Алгоритмы биоинформатики : практическое руководство / Ф. Компо, П. Певзнер ; пер. с англ. И. Л. Люско. - Москва : ДМК Пресс, 2023. – 684 с. – ISBN 978-5-93700-175-7.
3. Мазакова, Б. М. Разработка информационных систем для алгоритмов по обработке биопоследовательностей / Б. М. Мазакова, А. Х. Нурбекова, А. А. Найзагараева // EESJ. 2016. – № 7
4. Бурнашев, Р. Ф. Роль информационных технологий в развитии современной биоинформатики / Р. Ф. Бурнашев // Universum: химия и биология. – 2023. – №8-1 (110).
5. Часовских, Н. Ю. Биоинформатика : учебно-методическое пособие / Н. Ю. Часовских. – Томск : Издательство СибГМУ, 2015. – 109 с.
6. Королюк, И. П. Медицинская информатика : учебник / И. П. Королюк. – 2 изд., перераб. и доп. – Самара : ООО «Офорт» : ГБОУ ВПО «СамГМУ». 2012.— 244 с; ил.
7. Антао, Т. Биоинформатика с Python: книга рецептов: Современные библиотеки и приложения Python для решения реальных задач вычислительной биологии : практическое руководство / Т. Антао ; пер. с англ. И. Л. Люско. – Москва : ДМК Пресс, 2023. - 345 с. – ISBN 978-5-93700-201-3.

Разработка платформы для организации соревнований по программированию

Е. А. Саков

Студент бакалавр

Д. Г. Шлыков

Студент бакалавр

Д. И. Соломатин

Старший преподаватель

Введение

Соревнования по программированию становятся всё популярнее среди студентов и IT-специалистов, помогая компаниям находить таланты, оценивать кандидатов и находить нестандартные решения [1].

Для их проведения нужна удобная платформа, обеспечивающая прозрачность и простоту участия. Она должна позволять организаторам управлять мероприятиями, а участникам – регистрироваться и отслеживать результаты. Важны интуитивный интерфейс, быстрая регистрация и гибкие механизмы аутентификации, включая поддержку командных соревнований.

Серверная часть платформы разрабатывается на основе микросервисной архитектуры для масштабируемости и устойчивости. Безопасность данных обеспечивается надёжной аутентификацией, шифрованием и проверенными протоколами.

Цель работы – создание серверной части платформы, отвечающей требованиям современных соревнований: надёжность, безопасность и удобство для организаторов и участников.

1. Обзор аналогов

Было выделено одно веб-приложение, которое можно считать аналогом рассматриваемого решения, – платформа Timepad. Она предназначена для организации мероприятий, включая конференции, семинары и культурные события, и предоставляет инструменты для создания событий, управления регистрацией участников и обработки платежей.

Среди её преимуществ – встроенная система продажи билетов,

© Саков Е. А., Шлыков Д. Г., Соломатин Д. И., 2025

поддержка платных регистраций, email-рассылок, CRM-функционал, интеграция с аналитическими сервисами (Google Analytics, Яндекс.Метрика), а также интуитивный интерфейс.

Однако для проведения командных соревнований по программированию платформа имеет ряд критических ограничений.

– Отсутствие поддержки командных регистраций. Участники регистрируются только индивидуально, что делает невозможным создание команд, распределение ролей (капитан, участник) или автоматическую сборку группы. Это усложняет организацию хакатонов и соревнований, где требуется командное участие [2].

– Необходимость повторного ввода данных. Платформа не предусматривает профилей пользователей. Участникам приходится каждый раз заново указывать личную информацию (например, навыки, опыт, контакты), что создаёт неудобства для постоянных пользователей и увеличивает время регистрации.

– Ограниченная кастомизация страниц мероприятий. Организаторы не могут гибко настраивать структуру страницы соревнования: добавлять разделы с техническими требованиями, правилами, файлами для загрузки (например, условиями задач, шаблонами решений) или интегрировать специализированные инструменты (онлайн-компиляторы, системы проверки кода).

– Отсутствие функциональности для управления соревновательным процессом. Нет возможности отслеживать этапы выполнения заданий, вручную или автоматически проверять результаты, публиковать таблицы лидеров или синхронизироваться с системами проведения контестов (Codeforces, LeetCode).

Таким образом, несмотря на удобство для стандартных мероприятий, Timepad не покрывает ключевые потребности командных соревнований: работу с группами участников, кастомизацию под технические требования и интеграцию со специализированными сервисами.

2. Структура базы данных

При написании серверной части нашего приложения, вначале была разработана база данных (см. рис. 1).

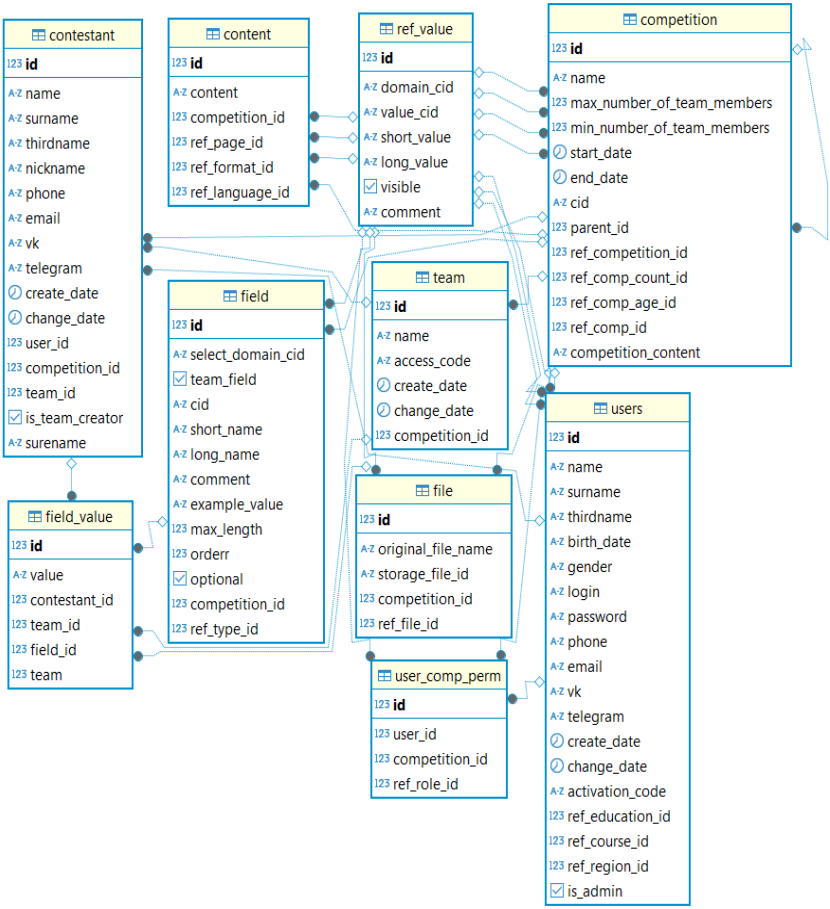


Рис. 1. База данных

В рамках данной базы данных были созданы три уникальных таблицы: ref_value, field и field_value.

Таблица ref_value представляет собой некий словарь, в котором хранятся все значения нужные администратору или организатору соревнований при создании или редактировании страниц, а участнику при регистрации. В ней с помощью sql-скрипта сразу сгенерированы значения, к примеру, города, области, тип соревнований, тип страницы или ступень обучения участника.

Field отвечает за поля, которые организатор добавляет в форму регистрации участника (см. рис. 2). У поля есть свой сид, тип, например, «имя участника», обязательное ли это поле или нет, а также примеры значений, которые надо вводить и комментарий, который уточняет назначение этого поля.

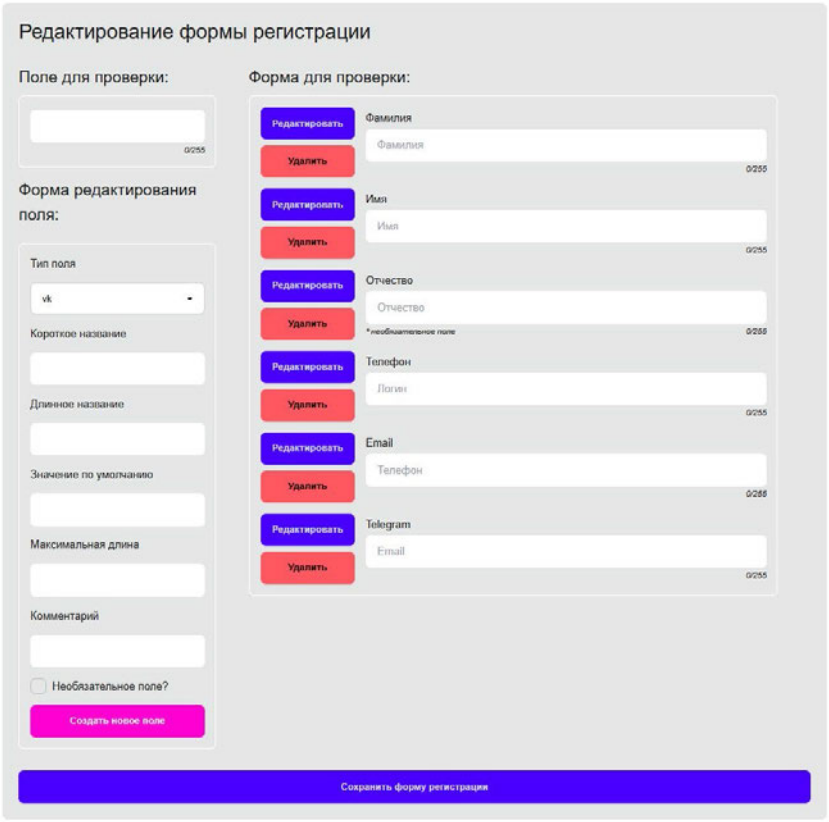


Рис. 2. Создание формы регистрации на соревнования

Таблица field_value отвечает за сами значения, которые находятся в полях. Они привязаны к определенному соревнованию, полю, а также участнику и команде, если соревнование командное (см. рис. 3).

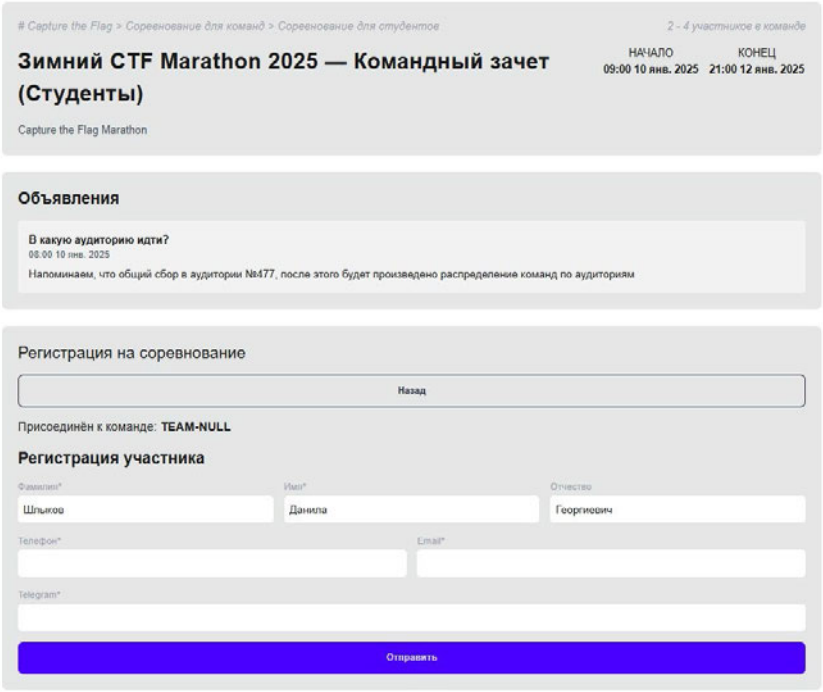


Рис. 3. Регистрация участника на соревнование

Эти три таблицы делают приложение масштабируемым и расширяемым, а также они позволяют, не меняя содержимое базы данных, изменять и создавать поля для регистрации участников на каждое соревнование так, как этого требует регламент и нужды самого мероприятия [3].

3. Серверная часть приложения

Серверная часть реализована на языке Java с использованием фреймворка Spring Boot, что обеспечивает модульность, устойчивость и удобство масштабирования системы. Архитектура построена по принципу микросервисов, каждый из которых отвечает за отдельную функциональность: управление пользователями, событиями, регистрацией и аутентификацией [4].

Основные технологии:

– язык: Java;
– фреймворк: Spring Boot;

– база данных: PostgreSQL;
– контейнеризация: Docker;
– архитектура: Микросервисная;
– API: REST, авторизация через JWT.

Система обеспечивает безопасную работу с персональными данными участников, включая шифрование и строгую аутентификацию. Использование PostgreSQL позволяет эффективно управлять гибкой и расширяемой структурой БД [5].

4. Клиентская часть приложения

Клиентская часть платформы разработана с учётом современных требований к юзабилити и адаптивности, что обеспечивает участникам интуитивно понятный доступ ко всем функциям системы. На главной странице каждому мероприятию соответствует карточка с ключевыми атрибутами – названием, датами начала и окончания, текущим статусом регистрации – а при её раскрытии пользователю становится доступен полный набор описательных материалов, включая регламент и правила проведения.

Регистрационная форма отличается интерактивностью: все обязательные поля отмечены визуально, а встроенная проверка корректности данных в реальном времени снижает вероятность ошибок и ускоряет оформление заявки. Интерфейс автоматически адаптируется под размер экрана мобильных устройств, что сохраняет удобство взаимодействия и делает возможным прохождение регистрации в любом месте и в любое время.

Административная панель реализует принципы эффективного управления содержимым и структурой мероприятий без необходимости привлекать разработчиков. Организаторы получают доступ к единому пространству для создания и редактирования соревнований (см. рис. 3). Используемая структура форм с проверкой вводимых данных обеспечивает консистентность и предотвращает ошибки на этапе подготовки соревнования.

Редактирование текстового контента (разделы «Описание», «Правила», «Результаты», «Расписание») происходит в реальном времени, что позволяет оперативно обновлять информацию. Форма регистрации поддерживает гибкую настройку: при необходимости можно добавить поля для ввода данных о команде, менторе или других параметров, зависящих от специфики конкретного мероприятия.

5. Модуль генерации дипломов и сертификатов

Платформа предоставляет также модуль генерации дипломов, интегрированный в административный интерфейс (см. рис. 4). Он

позволяет автоматически формировать именные дипломы на основе загруженного шаблона и введённых данных. Это решение оптимизирует процесс итогового документооборота и обеспечивает персонализацию наградных материалов.

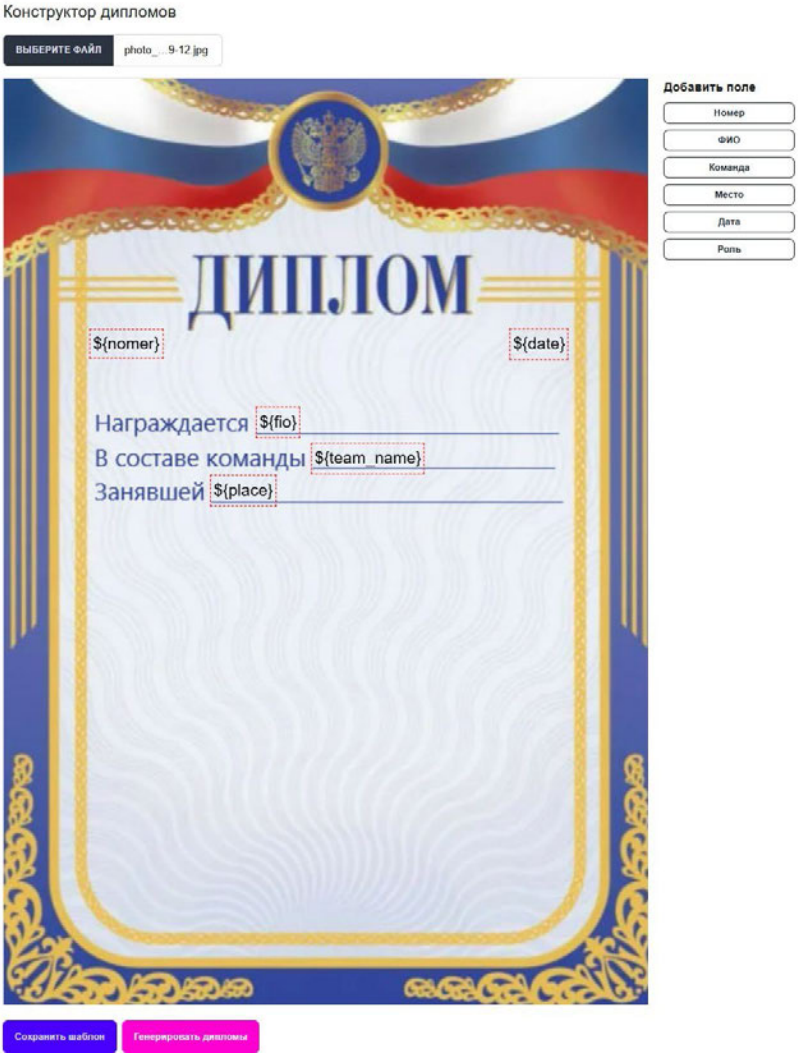


Рис. 4. Создание диплома

Сравнительный анализ с существующими платформами (например, Timepad) позволяет выделить ряд технических и функциональных особенностей. Так, в отличие от систем с фиксированными шаблонами и ограниченной структурой ввода, предложенная платформа поддерживает динамическую генерацию форм и позволяет организаторам самостоятельно настраивать интерфейс под конкретные цели и аудиторию. Это обеспечивает более высокую гибкость и адаптивность, важную для проведения как массовых, так и специализированных мероприятий.

Заключение

Данная статья посвящена разработке платформы для организации соревнований по программированию, обеспечивающей удобное управление мероприятиями, регистрацию участников и контроль результатов. Была реализована система с микросервисной архитектурой, поддерживающая безопасную аутентификацию, командные соревнования и гибкие механизмы доступа.

В результате разработана рабочая платформа, которая упрощает проведение IT-соревнований, повышает их прозрачность и удобство для организаторов и участников. Решение позволяет автоматизировать процессы регистрации, проверки решений и формирования рейтингов, что делает его применимым как в образовательных, так и в профессиональных конкурсах.

В перспективе платформу можно расширить за счёт интеграции с системами автоматической проверки кода (например, Яндекс.Контест или Codeforces), добавления аналитики и инструментов для борьбы с читерством. Данная разработка может быть полезна университетам, IT-компаниям и организаторам хакатонов, заинтересованным в проведении качественных и технологичных соревнований.

Список литературы

1. Кормен, Т.Х. Алгоритмы: построение и анализ / Т. Х. Кормен, Ч. Э. Лейзерсон, Р. Л. Ривест. – 3-е изд. – М.: Вильямс, 2022. – 1328 с.

2. Tanenbaum, A.S. Distributed Systems: Principles and Paradigms / A.S. Tanenbaum, M. Van Steen. – 3rd ed. – N. Y.: Pearson, 2017. – 800 p.

3. Martin, R.C. Clean Architecture: A Craftsman's Guide to Software Structure and Design / R.C. Martin. – N. Y.: Prentice Hall, 2017. – 432 p.

4. Фаулер, М. Микросервисы: паттерны разработки и рефакторинга / М. Фаулер. – СПб.: Питер, 2020. – 288 с.

5. Одинцов, И.О. Профессиональное программирование. Системный подход / И.О. Одинцов. – 4-е изд. – М.: БХВ-Петербург, 2021. – 848 с.

Разработка мобильной игры «Цифровой кузнечик»

П. А. Сапегин

Студент бакалавр

Д. И. Соломатин

Старший преподаватель

Введение

В последнее время информационные технологии (ИТ) – одна из сфер, где скорость развития кажется головокружительной. При этом уже давно ИТ служат не только бизнесу и всевозможным оптимизациям, но в равной мере и отдыху, развлечениям.

Например, игровая индустрия становится всё более популярной. Согласно исследованиям newzoo.com, в 2024 году количество людей, играющих в компьютерные игры, достигло отметки в более чем 3.4 миллиарда [1], хотя отношение общества к ним неоднозначно. Кто-то твердит об их вредности или бесполезности, а кто-то говорит об обратном [2, 3].

Однако иногда гаджет может служить просто физической оболочкой для игры, которая существовала до появления компьютеров. Например, компьютерные шахматы в первую очередь будут рассмотрены как шахматы, и только потом как «компьютерная игра». Некоторые становятся «геймерами» по той причине, что игра, которая им нравится или позволяет развивать какие-то личностные навыки, реализована как информационный продукт. Возможно, в чём-то делая её более интересной или понятной.

Данная работа посвящена описанию работы игры, цель которой – развитие логического и тактического мышления.

1. Описание игры

Существует множество классических игр-головоломок, для достижения победы в которых нужно думать: тетрис, судоку, го и другие. Рассматриваемая игра похожа на известные пятнашки. Её правила крайне просты, однако прохождение может потребовать немало времени и попыток.

Правила таковы: у игрока есть поле, состоящее из клеток. В некоторых из клеток находятся фишки с числами (рис. 1).

© Сапегин П. А., Соломатин Д. И., 2025

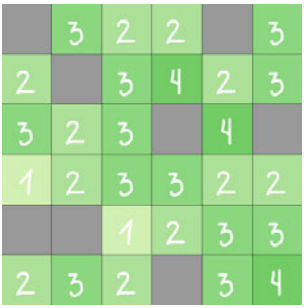


Рис. 1. Пример заполненного игрового поля

Игрок должен передвинуть каждую из фишек на число клеток, указанное на ней. Двигать можно только по прямой (влево, вверх, вправо, вниз). Фишки можно перепрыгивать, но каждую фишку можно двигать только один раз. Если в какой-то момент игрок не может сделать ход (остались не передвинутые фишки, но следующие ходы невозможны), считается, что он проиграл. На рис. 2 приведен пример конфигурации поля при поражении.

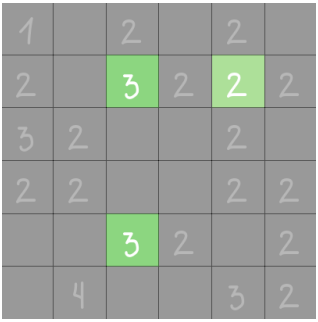


Рис. 2. Пример конфигурации поля при поражении

2. Ход игры

При запуске игры начинается отсчет времени. Игрок нажимает на активную клетку (ту, которая содержит ещё не передвинутую фишку). Если для этой клетки есть доступные ходы, то они подсвечиваются. Если игрок нажимает на подсвеченную клетку, то фишка с выбранной клетки перемещается на неё и становится неактивной. Та же клетка, откуда фишка была перемещена, становится пустой. Такую манипуляцию игрок должен провести с каждой активной клеткой.

После каждого хода выполняется проверка на победу и поражение. Победа достигается, если на поле не остается активных клеток. Поражение – ситуация, когда отсутствуют доступные ходы, либо какая-то клетка оказывается заблокированной со всех сторон неактивными клетками (то есть уже передвинутыми) – пример на рис. 3.

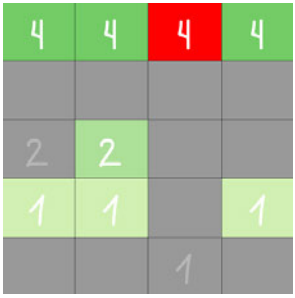


Рис. 3. Пример заблокированной клетки

В случае победы или поражения пользователю будет показано всплывающее сообщение о соответствующем событии. В любой момент игрок может начать игру заново или сделать ход назад.

3. Генерация уровней

Очевидно, что игроку важно заранее знать, что очередное расположение фишек на поле (назовем его «уровнем») можно разрешить в победу. При этом создание уровней вручную обладает рядом недостатков: это крайне утомительная задача, при выполнении которой несложно ошибиться; она отнимает время и ограничивает игроков в количестве доступных уровней. Итак, возникает задача, которую можно разбить на следующие пункты:

1. Нужно реализовать возможность автоматической генерации уровня.
2. Уровень должен быть проходимым.
3. Необходима настройка уровня сложности – дополнительное требование, которое сделает игру доступной для более широкого круга лиц.

Для решения задачи был разработан алгоритм, который удовлетворяет каждому из выдвинутых требований. Рассмотрим его детально.

Пользователь должен указать размер поля, задав высоту и ширину. В оригинальной игре поле должно быть квадратным, но данная реализация позволяет создавать и прямоугольные поля. Размер может

быть любым, но для играбельности установлены ограничения от 2x2 до 10x10.

Затем задается максимальное значение фишки. В общем случае оно может принимать значения от 1 до min(field.height, field.width) - 1, но зависит также от уровня сложности, которых всего четыре: «Легкий», «Средний», «Сложный» или «Ультра». Сложность влияет на минимальный и максимальный размер поля, его заполненность и процент фишек с большим значением.

Теперь подробно рассмотрим работу функции генерации уровня.

В нее передаются следующие входные параметры: высота и ширина поля, максимальное значение фишки и необходимая сложность.

Функция подготавливает данные: создает поле и считает общее количество клеток на нем. Далее запускается алгоритм. Его общая идея заключается в следующем: в цикле случайные клетки заполняются числами. Цикл выполняется до тех пор, пока на поле есть пустые клетки, в которые ещё можно записать какое-то значение с гарантией сохранения проходимости уровня.

Рассмотрим принцип работы алгоритма. Он выбирает случайную пустую клетку (назовем её cell_from). Далее находит все пустые клетки, куда из неё потенциально можно передвинуть фишку (учитывая переданное в функцию максимальное значение фишки). Из этого списка потенциально доступных клеток случайным образом выбирается одна (назовем ее cell_to). Происходит это с учетом заданной сложности уровня: чем он выше, тем больше шанс, что будет выбрана самая далекая клетка (таким образом, на поле будет много фишек с большим значением).

Далее в cell_from записывается число, которое равно расстоянию до выбранной cell_to. А в качестве следующей cell_from берется текущая cell_to (таким образом как бы строится цепочка клеток).

Если же для какой-то из cell_from список потенциально доступных клеток окажется пустым, то алгоритм помечает её как занятую и переходит к поиску следующей пустой клетки, беря её в качестве cell_from.

Таким образом, данный алгоритм, как уже было сказано, как бы строит цепочки клеток на поле: в случайную клетку пишет число, которое равно расстоянию до следующей клетки. И так последовательно до тех пор, пока продолжение цепочки не станет невозможным. Последняя клетка в этой цепочке остаётся пустой. Таким образом, «решить» цепочку можно, просто переместив фишки в обратном порядке.

Однако фактически таких цепочек может оказаться несколько. А в конце работы алгоритма случайным образом часть клеток становятся пустыми (чем ниже сложность, тем больше окажется пустых клеток). С одной стороны, это упрощает прохождение. Однако становится невозможным проследить те цепочки, которые были созданы генератором.

4. Технологии

В качестве целевой платформы при создании игры были выбраны мобильные устройства, работающие на операционной системе Android. Согласно статистике за 2024 год [4], мобильные игры – самые популярные среди всех поколений. Android же является более популярной платформой, чем iOS, причем как в России, так и во всем мире [5, 6].

Для разработки мобильного приложения была использована технология Kivy — кроссплатформенный фреймворк на языке Python. Kivy позволяет разрабатывать приложения под Android, а также под другие платформы (Windows, Linux, macOS, iOS).

Фреймворк позволяет использовать широкий перечень Python-библиотек для разработки мобильных приложений, что упрощает разработку и сокращает расходы на неё. Такой подход делает технологию особенно подходящей для прототипирования и MVP-разработки.

Интерфейс для приложений может быть реализован как в .py коде, так и с помощью специального языка разметки, оформляемого в .kv файлах.

Возможность написания мобильного приложения на Python позволяет сделать его разработку удобной и быстрой. Кроссплатформенные технологии известны тем, что работают медленнее нативных приложений, а также не позволяют использовать аппаратные особенности тех или иных операционных систем. Однако в задаче разработки простой игры данные недостатки не существенны.

Kivy применяет OpenGL ES 2 для графического рендеринга и предоставляет набор виджетов с возможностью настройки. Визуальные элементы в Kivy стандартизированы и одинаковы на всех мобильных платформах, что с точки зрения пользовательского интерфейса может быть как плюсом, так и минусом [7].

Главное достоинство Kivy – это возможность писать мобильные приложения, используя чистый Python. Тем не менее, сам фреймворк не преобразует Python-код в нативный Android-код. Для этого требуется применять дополнительные инструменты. Например, Buildozer, который и был использован в реализованном проекте. При этом код выполняется

внутри встроенного интерпретатора Python, а не компилируется в нативный машинный код, что влияет на производительность приложений не лучшим образом [8].

Однако стоит отметить, что части Kivy, которые интенсивно используют процессор, написаны на Cython-коде [7]. Cython-компилятор преобразует его в C/C++ код, при этом предоставляя возможность вызывать функции на C и задавать C-типы для переменных и атрибутов классов. Это позволяет компилятору генерировать высокоэффективный C-код [8], тем самым ускоряя работу приложения.

Для хранения данных (уровни, статистика) приложения используется база данных SQLite.

Заключение

Таким образом, поставленная цель была достигнута. Была реализована логическая игра с простым интерфейсом и возможностью автоматической генерации уровней. Мобильный интерфейс позволяет удобно инкапсулировать игровые механики, чтобы игрок фокусировался только на самом процессе игры. А современные технологии позволяют создавать масштабируемые и простые приложения, при минимальных затратах времени.

Список литературы

1. Тенденции развития игровой индустрии в 2025 году [Электронный ресурс]. – Режим доступа: https://games.logrusit.com/ru/news/game-industry-trends-year/

2. Лаптева, А. В. Обзор современных исследований влияния видеоигр на когнитивные процессы / А. В. Лаптева // Современная зарубежная психология. – 2023. – № 4. – С. 111-122. – Режим доступа: https://psyjournals.ru/journals/jmfp/archive/2023_n4/Lapteva

3. Cutting, J. Higher working memory capacity and distraction-resistance associated with strategy (not action) game playing in younger adults, but puzzle game playing in older adults / J. Cutting, B. Copeland, F. McNab // Heliyon. – 2023. – Vol. 9. – Iss. 12. – e06306. – Режим доступа: https://www.cell.com/heliyon/fulltext/S2405-8440(23)06306-5

4. Т-Банк. Исследование T-Pay: чем в России отличаются друг от друга владельцы iPhone и смартфонов на Android [Электронный ресурс]. – Режим доступа: https://www.tbank.ru/about/news/09122024-t-pay-research-how-do-iphone-and-android-smartphone-owners-differ-from-each-other-in-russia/

5. StatCounter. Доля мобильных операционных систем на мировом рынке [Электронный ресурс]. – Режим доступа: https://gs.statcounter.com/os-market-share/mobile/worldwide

6. Evrone. Создание приложения для Android на Python [Электронный ресурс]. – Режим доступа: https://evrone.ru/blog/articles/python-mobile
7. Официальная документация Kivy [Электронный ресурс]. – Режим доступа: https://app.readthedocs.org/projects/kivy/downloads/pdf/latest/
8. GitHub-репозиторий компилятора cython [Электронный ресурс]. – Режим доступа: https://github.com/cython/cython

Автоматическая суммаризация больших текстов: современные нейронные подходы и их ограничения

В. В. Ситникова

Студент бакалавр

Е. Д. Проскуряков

Ассистент

Введение

Обработка естественного языка (Natural Language Processing, NLP) представляет собой раздел искусственного интеллекта и математической лингвистики, изучающий методы компьютерного анализа и генерации текстов на естественных языках. Задачи NLP направлены на создание таких систем, которые способны выполнять задачи, связанные с языком, на таком же уровне, как и человек. Одна из таких задач – автоматическая суммаризация текста (Automatic Text Summarization, ATS) [1].

Особую актуальность эта задача приобретает в образовательной сфере, где лекционные материалы, как правило, имеют значительный объем и структурную сложность. Зачастую студентам становится сложно прочитать, а главное понять, суть преподаваемых материалов. Суммаризация лекций поможет преподнести информацию в сжатом изложении, что экономит время, делает обучение доступным.

Данная работа посвящена анализу нейронных сетей для задачи суммаризации текста и выбора наилучшей из них для проекта курсовой работы.

1. Предметная область

Задача суммаризации сложная, потому что совмещает в себе два навыка из сферы обработки естественных языков. Это – понимание читаемых текстов и возможность писать тексты самостоятельно.

Современные нейросетевые архитектуры, достигли многого в области суммаризации, обеспечивая не только выделение ключевых идей, но и генерацию связных сжатых текстов за счет механизмов внимания и многоуровневого анализа контекста. Одним из таких архитектурных подходов является Text-to-Text Transfer Transformer(T5), в рамках которого все задачи NLP рассматриваются как генерация текста

© Ситникова В. В., Проскуряков Д. Е., 2025

по входному тексту. Многоязычная версия этой архитектуры (mT5) позволяет работать с текстами на более чем 100 языках.

Так же, в последнее время широко популярны Большие языковые модели, которые обучены на огромных объемах текстовых данных. Принцип, лежащий в основе таких моделей: предсказать следующий токен, учитывая последовательность предыдущих токенов.

В общем контексте модели демонстрируют хорошие результаты, но при применении их на большие тексты, появляется ряд проблем:

– Во-первых, ограничения на длину входного контекста делают обработку больших текстов фрагментарной, приводя к потере общего контекста.
– Во-вторых, недостаточная длина выходного резюме. Для текста объемом в несколько десятков страниц модель может ограничить вывод в пару абзацев, что приводит к потере значимой информации.

Существует несколько методов суммаризации текста. По формату вывода они делятся на экстрактные (extractive) и абстрактные. Экстрактные находят подмножество самых информативных частей и составляют краткое изложение. Абстрактные методы генерируют новый текст, содержательно обобщающий первичный документ [2].

2. Подготовка к исследованию

Для исследования выбраны несколько нейросетей, предназначенных для решения задачи автоматической суммаризации текста. Выбор модели был обоснован стремлением протестировать различные методы и архитектуры моделей. Модели Gigachat Lite и DeepSeek-V3 (Free) были выбраны в качестве представителей больших языковых моделей. MT5 multilingual XLSum выбрана из-за своей популярности на платформе Hugging Face, а google/mt5-small – это облегченная модель. Рассмотрим их детальнее:

– Gigachat Lite – это компактная языковая модель от Сбера на архитектуре Transformer, содержащая около 7 миллиардов параметров. Модель поддерживает генерацию, суммаризацию и понимание текстов на русском языке. У модели есть несколько тарифных планов API: Freemium, Business и Enterprise. Каждая из них предлагает различное количество потоков запросов, пакетов токенов и вариант использования API. Для тестирования был выбран тариф Freemium.
– DeepSeek-V3 построена на архитектуре Mixture-of-Experts (MoE) и оптимизирована для многозадачности – от программирования до математических вычислений и работы с разными языками. При обработке каждого токена активируется

всего 37B параметров из общих 671B, что делает модель более эффективной с точки зрения использования ресурсов.
- MT5 multilingual XLSum – модель, предназначенная для автоматического суммирования на нескольких языках. Она обучена на датасете XL-Sum, содержащем более 1 миллиона пар "статья-сводка", собранных с сайта BBC News. Эта модель позволяет суммировать длинные фрагменты текста в более короткие версии, сохраняя при этом основные моменты и идеи.
- google/mt5-small – облегченная версия многоязычной модели Text-to-Text Transfer Transformer от компании Google. Предназначенная для обработки текста на 105 языках, в том числе поддерживает русский язык. Требуется дообучение на специализированных датасетах. Обратиться можно напрямую через библиотеку в python.

3. Характеристики входных текстов

В качестве данных для анализа был выбран датасет IlyaGusev/gazeta с платформы Hugging Face, содержащий новостные статьи и соответствующие краткие саммари. Благодаря наличию разметки и разбиения на обучающую, валидационную и тестовую выборки, датасет подходит для оценки современных моделей суммаризации, включая трансформеры [3].

Исходных текст в среднем составляет 37 предложений, и 767 токенов. Саммари – 50 токенов и 3–4 предложения.

4. Метрика, используемая для оценки результатов

На данный момент существует множество метрик, позволяющих оценить эффективность языковых моделей. Одной из таких является набор метрик ROUGE. Основная идея ROUGE заключается в измерении сходства между словами, фразами и последовательностями в сгенерированном тексте и в эталонном. Это сходство помогает оценить, насколько сгенерированный машиной текст соответствует ожиданиям человека в отношении содержания и структуры [4].

Анализ был проведен по трем метрикам: ROUGE-1, ROUGE-2, ROUGE-L. Для каждой метрики были вычислены Precision, Recall и F1-score [5].

Precision – показывает, какая доля объектов, выделенных как положительные, действительно являются релевантными. Чем выше показатель, тем меньше в сгенерированном тексте лишней информации:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (1)$$

где TP – верно предсказанные положительные случаи, FP - отрицательные случаи, ошибочно предсказанные, как положительные.

Recall – показывает, какую долю всех значимых фрагментов оригинального текста модель извлекла и включила в итоговый саммари:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

где FN – положительные случаи, ошибочно предсказанные, как отрицательные.

F1-score – среднее гармоническое между Precision и Recall:

$$F1 = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

Использование этой метрики позволяет оценить модели суммаризации, которые не только включают точные элементы, но и не упускают значимую информацию.

ROUGE-1 измеряет пересечение отдельных слов (унитермов) между саммари, созданным моделью, и эталонным текстом. Метрика показывает, насколько много слов из оригинала присутствует в сгенерированном тексте.

ROUGE-2 оценивает совпадения пар слов (биграмм). Это делает метрику более чувствительной к структуре и связности текста. ROUGE-2 полезен для анализа того, насколько хорошо модель передаёт смысловые связи и порядок слов.

ROUGE-L основан на концепции самой длинной общей подпоследовательности слов (Longest Common Subsequence, LCS). Метрика измеряет длину самой длинной подпоследовательности слов, которые встречаются как в тексте-эталоне, так и в сгенерированном саммари, в том же порядке, но не обязательно подряд.

5. Результаты тестирования моделей

Модели были протестированы на выборке из 1000 примеров и сравнивались с референсами из датасета IlyaGusev/gazeta.

На рис. 1 представлены значения Precision по метрикам ROUGE-1, ROUGE-2 и ROUGE-L. По результатам тестирования видно, что модель GigaChat Lite показала наивысшие результаты, что свидетельствует о том, что она обладает высокой способностью к точному включению информативных элементов оригинального текста в исходное саммари. Модели DeepSeek V3 (free) и google/mt5-small, часто включали в итоговый текст слова и фразы, не соответствующие ключевой информации из исходного текста.

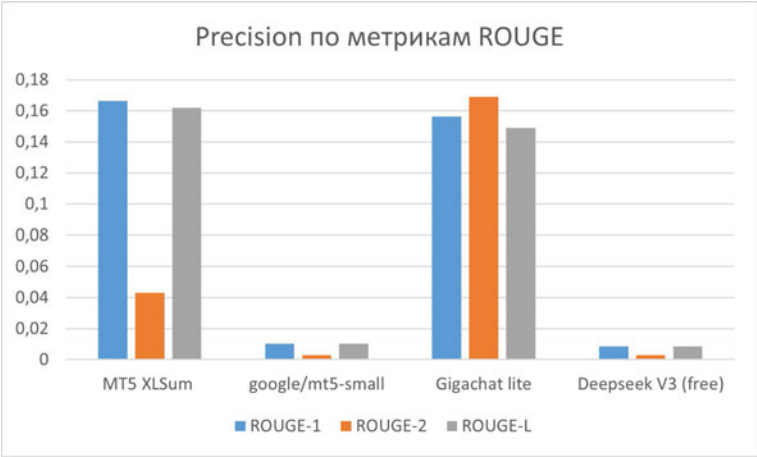


Рис. 1. Результаты precision тестируемых моделей

На рис. 2 приведены результаты Recall для всех протестированных моделей. Модель MT5 XLSum демонстрирует наилучшую полноту, это значит, что модель охватывает значительную часть релевантной информации, однако показатель ROUGE-2 низкий, что может означать на менее устойчивую генерацию связных фраз. Модель GigaChat Lite лидирует по ROUGE-2, это может свидетельствовать о том, что модель лучше восстанавливает связные выражения. Остальные модели показали низкие результаты в текущем тестировании, что говорит о необходимости настройки или повторном тестировании.

На рис. 3 представлены результаты F1-score для всех моделей. GigaChat Lite показала высокие результаты, это свидетельствует о том, что модель обладает лучшей согласованностью между включением релевантной информации и ее точными формулировками. Модель MT5 XLSum так же продемонстрировала высокие значения, но уступает в ROUGE-2. Остальные модели показали низкие результаты, что свидетельствует о недостаточной эффективности с точки зрения полноты и точности.

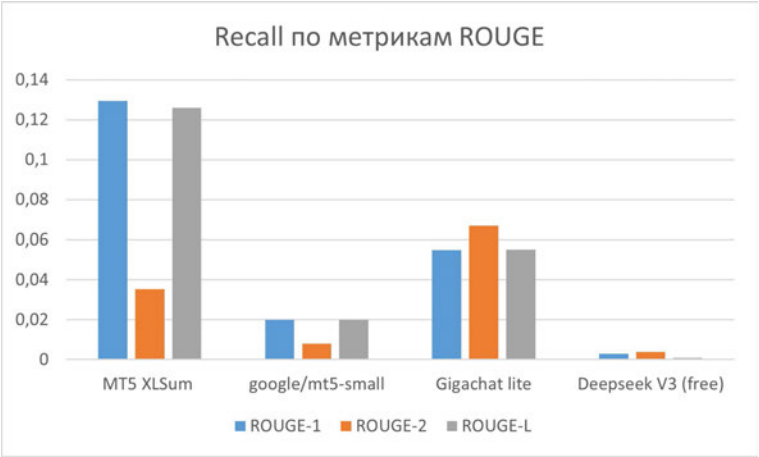


Рис. 2. Результаты Recall тестируемых моделей

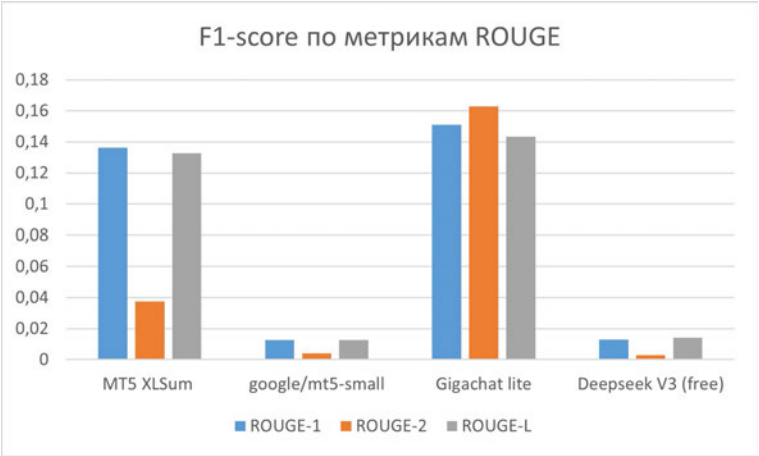


Рис. 3. Результаты F1-score тестируемых моделей

Заключение

Данная статья посвящена автоматической суммаризации больших текстов с использованием нейросетевых моделей. В ней описаны модели, а также рассмотрена метрика ROUGE для оценки саммари. В ходе исследования модели были протестированы и сравнены, в результате чего была определена наиболее эффективная. По результатам тестирования наилучшие показатели F1-score и ROUGE-2 показала

модель GigaChat Lite, что говорит о ее способности точно извлекать релевантную информацию и формулировать ее в связной форме. Модель MT5 XLSum показала высокие значения Recall, что свидетельствует о ее способности охватывать значительную часть ключевой информации. Низкие результаты Deepseek V3 могут быть связаны с ограничением количества бесплатных токенов для API или возможным несоответствием формата входного текста предпочтениям модели. Модель google/mt5-small, как облегченная модель, требует дообучения для задачи суммаризации. Таким образом, модель GigaChat Lite была выбрана в качестве лучшей модели суммаризации текста. В дальнейшем планируется протестировать модель на лекционных материалах и интегрировать её в чат-бот, предназначенный для помощи студентам в суммаризации учебных текстов.

Список литературы

1. Белякова А. Ю. Обзор задачи автоматической суммаризации текста / А. Ю. Белякова, Ю. Д. Беляков // Инженерный вестник Дона. – 2020. – №10(70) – С. 142-159.

2. Суммаризация текста: подходы, алгоритмы, рекомендации и перспективы [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/514540/

3. Gazeta [Электронный ресурс]: датасет. – Режим доступа: https://huggingface.co/datasets/IlyaGusev/gazeta

4. LLM Studio. BLEU, ROUGE и BERTScore: сравнение метрик оценки качества текстов [Электронный ресурс]. – Режим доступа: https://llmstudio.ru/blog/bleu-rouge-bert-score

5. Olson, D. L. Advanced data mining techniques / D. L. Olson, D. Delen. – Berlin : Springer, 2008. – 180 с.

Практики и подходы к организации баз данных в высоконагруженных системах

Д. Ю. Скарга

Студент бакалавр

Д. И. Соломатин

Старший преподаватель

Введение

Современные информационные системы, обрабатывающие большие объемы данных и интенсивный трафик, нуждаются в архитектурных решениях, которые обеспечивают устойчивость к отказам, масштабируются горизонтально и быстро обрабатывают запросы. Растущие требования к доступности и времени отклика означают, что решение с одним сервером нежизнеспособно, что требует перехода к распределенным базам данных.

В этой статье рассматриваются два основных механизма масштабирования – репликация и шардирование, а также связанная с ними проблема согласованности данных.

Целью этой статьи является построение модели отказоустойчивости распределенной системы и анализ архитектурных решений, которые могли бы обеспечить надежность и масштабируемость баз данных.

1. Репликация

Репликация – это процесс копирования данных между несколькими узлами (серверами), позволяющий повысить доступность, устойчивость к сбоям и масштабируемость по чтению. Наиболее распространённый подход – репликация с ведущим узлом. Один из узлов, лидер, получает все операции записи, а остальные, ведомые, реплицируют его изменения и могут обслуживать запросы на чтение.

При синхронной репликации лидер дожидается подтверждения от одной или нескольких реплик прежде, чем ответить клиенту. Это гарантирует согласованность, но увеличивает задержку и делает систему чувствительной к сбоям реплик.

© Скарга Д. Ю., Соломатин Д. И., 2025

В асинхронной репликации лидер сразу подтверждает запись, не дожидаясь ответа от ведомых. Это ускоряет отклик, но при сбое лидера часть записей может быть потеряна.

Схема работы синхронной и асинхронной репликации представлена на рис. 1 [1].

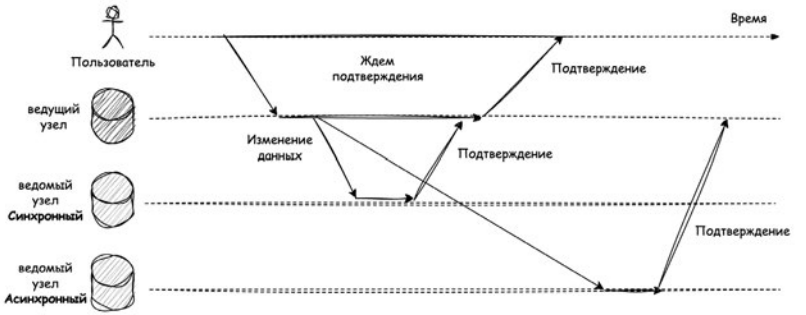


Рис. 1. Репликация с ведущим узлом и с одним синхронным, одним асинхронным ведомыми узлами

На практике часто применяется полусинхронная конфигурация, при которой одна реплика работает синхронно, а остальные – асинхронно. Это обеспечивает разумный баланс между доступностью и надёжностью.

2. Шардирование

Шардирование представляет собой разбиение набора данных на логические части (шарды), каждый из которых размещается на отдельном узле. Это позволяет распределить нагрузку и данные по кластеру и обеспечивает горизонтальное масштабирование по объёму хранимой информации. Существует несколько подходов к тому, как именно делить данные между узлами.

Одним из способов является шардинг по диапазонам значений ключей. Каждый шард содержит непрерывный диапазон значений ключа. Этот метод прост в реализации и эффективен для запросов по диапазону ключей. Тем не менее, он чувствителен к неравномерному распределению данных. При высокой концентрации конкретных значений возможна перегрузка отдельных серверов – образование "горячих точек".

Другой способ – разделение по хэш-функции. Здесь значение ключа преобразуется с помощью хэш-функции, а результат используется

для определения секции. Такой подход обеспечивает равномерное распределение данных и позволяет избежать перегрузки отдельных шардов. Основной недостаток – неэффективность запросов по диапазонам, поскольку хэширование нарушает естественный порядок ключей.

Эффективная перебалансировка нагрузки достигается за счёт фиксированного количества шардов и динамического шардинга. При добавлении сервера, секции распределяются между ними, минимизируя при этом объем перемещаемых данных.

Для корректной маршрутизации запросов необходимо хранить информацию о местоположении данных. Часто для этого используется ZooKeeper – распределённый координатор, содержащий актуальные сведения о размещении шардов и обеспечивающий согласованность маршрутизации внутри кластера. [2]

3. Согласованность, доступность и устойчивость к сбоям

В распределённых системах критически важно учитывать компромиссы между тремя основными свойствами: согласованностью данных, доступностью системы и устойчивостью к сетевым сбоям. Эти характеристики не всегда могут быть обеспечены одновременно, особенно в условиях нестабильной сетевой среды.

CAP-теорема постулирует базовое ограничение: при наличии сетевых разделений система не может одновременно гарантировать согласованность (Consistency), доступность (Availability) и устойчивость к разделению сети (Partition tolerance). [3]

1. Согласованность означает, что все участники системы видят идентичные данные.
2. Доступность подразумевает, что каждый запрос получает ответ, даже если он не всегда соответствует последним изменениям.
3. Устойчивость к разделению сети – что система продолжает функционировать, несмотря на потерю связи между частями кластера.

Когда возникает сетевой сбой, приходится делать выбор: сохранить либо согласованность, либо доступность, но не обе характеристики одновременно. Это ограничение лежит в основе архитектурных решений при проектировании распределённых баз данных и систем хранения.

Одним из практических механизмов управления согласованностью при репликации является использование кворума. Предположим, данные реплицируются на n узлов. Для записи данных требуются

успешное завершение операции на w узлах, а для чтения – получение данных хотя бы с r узлов. Если выполняется условие $w+r>n$, то хотя бы один узел участвовал и в чтении, и в записи, что гарантирует согласованность данных. [4]
Принцип работы кворумов представлен на рис. 2.

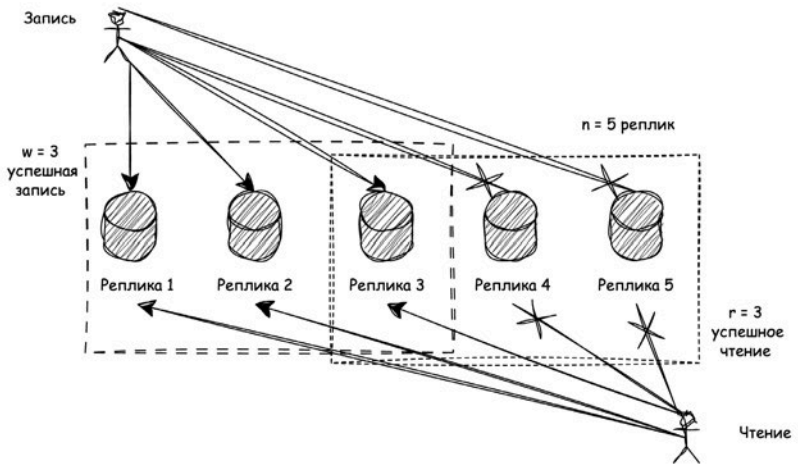


Рис. 2. При $w+r>n$ как минимум одна из читаемых r реплик точно получила информацию о самой последней операции записи

Этот подход позволяет гибко настраивать баланс между доступностью и согласованностью в зависимости от конкретных требований.

4. Модель оценки отказоустойчивости

Для количественной оценки устойчивости системы к сбоям была построена модель на основе биномиального распределения. Пусть:

- p – вероятность отказа одного узла
- N – общее число узлов
- K – максимальное допустимое число одновременных отказов

Тогда вероятность того, что система будет работоспособной:

$$P = 1 - \sum_{i=K+1}^{N} C(N,i) * p^i * (1-p)^{N-i} \quad (1)$$

На основе модели построен график зависимости доступности от числа реплик и допустимого количества отказов (рис. 3).

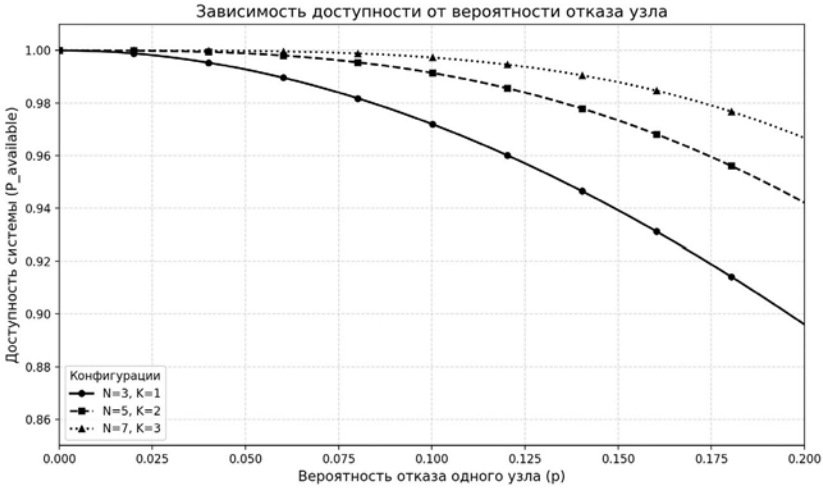


Рис. 3. График зависимости доступности от числа реплик и количества отказов

Результаты показывают, что при $p = 0.1$ и $K = 2$, для обеспечения доступности $\geq 99\%$ необходимо использовать не менее 5 узлов.

Для более глубокого понимания того, как работает система при заданных параметрах, использован метод Монте-Карло, который с помощью случайных чисел моделирует отказы узлов. Было проведено 500 000 испытаний, чтобы оценить вероятность того, что количество отказавших узлов не превысит допустимый предел.

На графике, продемонстрированном на рис. 4 показано распределение числа отказавших узлов при $p = 0.1$, $K = 2$ и 5 узлах. Результаты подтверждают теоретические расчеты, демонстрируя, что система будет доступна в 99% случаев, что согласуется с предыдущей моделью.

Предложенная модель позволяет проектировать системы с прогнозируемым SLA : в зависимости от допустимого количества отказов и вероятности сбоя отдельного узла, можно вычислить минимальное необходимое число реплик [5].

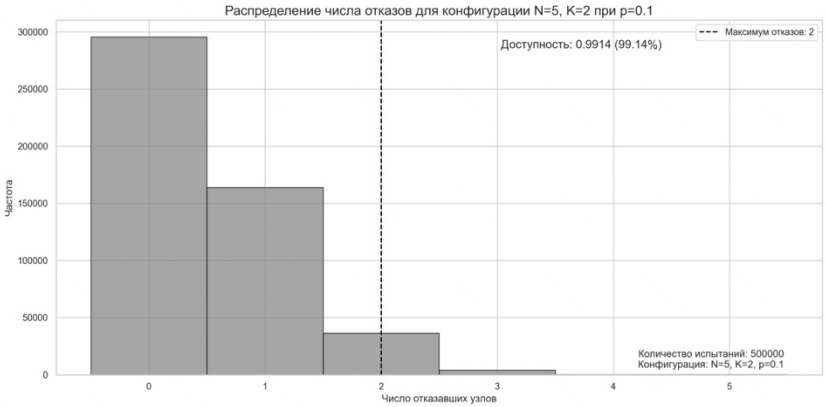


Рис. 4. Распределение числа отказов

Заключение

В представленном исследовании был произведен анализ ключевых архитектурных решений, применяемых при создании высоконагруженных распределенных баз данных. Были изучены процессы репликации и шардирования, их воздействие на способность системы к росту нагрузки и устойчивость к сбоям, а также ограничения, возникающие в связи с теоремой CAP и алгоритмами консенсуса на основе кворумов.

Разработанная математическая модель отказоустойчивости предоставляет возможность принятия обоснованных решений при выборе числа реплик и допустимого уровня рисков в контексте разработки систем, критичных к сбоям. В итоге можно сделать вывод, что достижение необходимой надежности и масштабируемости, соответствующей требованиям современных приложений, требует комплексного подхода, учитывающего специфику нагрузки и характер данных, с использованием всех вышеперечисленных методов.

Список литературы

1. Клеппман, М. Высоконагруженные приложения. Программирование, масштабирование, поддержка / М. Клеппман. – СПб.: Питер, 2018. – 640 с.

2. Junqueira, F. P. ZooKeeper: Distributed Process Coordination / F. P. Junqueira, B. Reed. – Sebastopol: O'Reilly Media, 2013. – 238 p.

3. Все о CAP теореме [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/328792/

4. Abadi, D. J. Consistency Tradeoffs in Modern Distributed Database System Design / D. J. Abadi // IEEE Computer Magazine. – 2012. – Vol. 45. – Iss. 2. – P. 37–42. [Электронный ресурс]. – Режим доступа: http://cs-www.cs.yale.edu/homes/dna/papers/abadi-pacelc.pdf

5. Соглашение об уровне обслуживания [Электронный ресурс]. – Режим доступа: https://yandex.cloud/ru/docs/glossary/sla

Разработка видеоплатформы с поддержкой транскрибации на базе мобильного приложения

М. Б. Смагина

Студент бакалавр

В. А. Ушаков

Ассистент

Введение

В последние годы наблюдается значительный рост интереса к разработке видеоплатформ, обеспечивающих обмен видеосообщениями и их автоматическую транскрибацию. Это связано с увеличением потребности пользователей в многофункциональных платформах, способных адаптироваться к требованиям быстрого обмена информацией. Согласно данным Mediascope, 49% населения России старше 12 лет используют Telegram ежедневно [1], что свидетельствует о высокой популярности и значимости данной платформы для коммуникации.

Средний пользователь проводит в мессенджере около 44 минут в день, что на пять минут больше, чем в аналогичный период 2023 года. Аудитория Telegram продолжает расти, с охватом в 73% по итогам первого квартала 2024 года. Особенно активно мессенджер используется молодежной аудиторией: 80% россиян в возрасте от 12 до 24 лет заходят в Telegram хотя бы раз в день [1]. В то время как старшие возрастные группы также проявляют интерес к платформе, свидетельствуя о её универсальности и адаптивности к различным пользователям.

В условиях быстрого развития технологий и увеличения скорости обмена информацией возникает необходимость в создании решений, которые могут обеспечить удобный и эффективный способ коммуникации. В связи с этим создание мессенджера, в котором пользователь имеет легкодоступный способ общения с людьми, со специализацией на видеосообщениях и возможностью транскрибации видеосообщений в текст, представляется актуальным шагом для улучшения качества общения и удовлетворения растущих потребностей пользователей.

© Смагина М. Б., Ушаков В. А., 2025

1. Используемые технологии для реализации мессенджера ViChat

Современные мессенджер-приложения представляют собой сложные системы, требующие тщательного планирования архитектуры, обеспечения безопасности и оптимизации производительности. В данной работе мы рассмотрим реализацию мессенджера ViChat, выполненную с использованием фреймворков Flutter, SpringBoot и СУБД PostgreSql и проанализируем принятый набор инструментов разработки.

Данные технологии выбраны исходя возможностей, которые они дают для решения задач. В качестве преимуществ выбранных технологий можно отметить:

Для фреймворка SpringBoot:

– большой набор функциональных возможностей;

– удобные инструменты для работы с PostgreSQL;

– готовые встроенные серверы (Tomcat), обеспечивающие ускоренное и более продуктивное развертывание приложений.

Для PostgreSQL:

– наличие широкого функционала для горизонтального и вертикального масштабирования;

– поддержка множества типов данных.

Для Flutter:

– кроссплатформенность;

– понятная и полная документация.

2. Обоснование выбора архитектурных паттернов в проекте ViChat

1. Выбор MVP для Flutter-клиента.

Для реализации frontend-части был выбран паттерн Model-View-Presenter (рис. 1), поскольку такой подход предлагает четкое и простое разделение ответственности между компонентами [2]. View отвечает исключительно за отображение пользовательского интерфейса, Presenter управляет бизнес-логикой, а Model представляет данные. Такое разделение на компоненты позволяет разработчикам легко ориентироваться в коде.

Еще один немало важный «плюс»: такая архитектура обеспечивает простоту внедрения новых разработчиков в проект, так как структура кода проста и интуитивно понятна даже для неопытных разработчиков.

При выборе архитектуры рассматривались альтернативные подходы, такие как MVVM и BLoC. Однако они не были выбраны из-за избыточной сложности для мессенджер-приложения. MVVM, например, требует больше шаблонного кода и сложнее в тестировании, а BLoC

излишне усложняет простые экраны и требует больше усилий на поддержку.

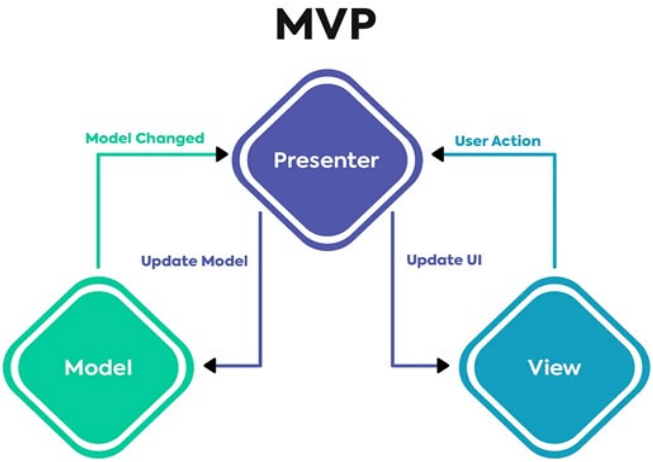


Рис. 1. Архитектура Flutter MVP

2. Выбор MVC для Spring Boot-сервера.

Архитектура Model-View-Controller (рис. 2) для серверной части является довольно-таки распространённой благодаря хорошей интеграции со Spring Boot и большим сообществом. Большинство ошибок, которые могут возникнуть при разработке, уже разобраны на различных площадках Интернета другими пользователями, которые предлагают свои решения. MVC является стандартным подходом в Spring Boot, что позволяет получить доступ к большому количеству готовых инструментов [3]. Это значительно ускоряет разработку и упрощает поддержку приложения.

Производительность и безопасность также сыграли важную роль в выборе MVC. Spring Boot предоставляет оптимизированную обработку запросов и эффективное управление ресурсами. Встроенная поддержка Spring Security и простая интеграция с JWT обеспечивают необходимый уровень безопасности для мессенджер-приложения [4].

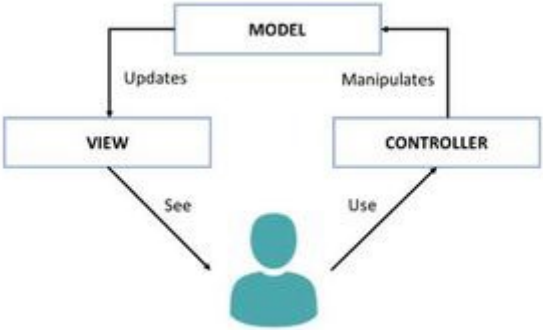


Рис. 2. Архитектура MVC для Spring Boot-сервера

3. Анализ использования ключевых Flutter библиотек в проекте

Использование библиотек при разработке с фреймворком Flutter значительно облегчает процесс создания приложений. Одной из ключевых библиотек для организации взаимодействия с серверной частью является Retrofit Generator. Рассмотрим её основные преимущества:

1. Автоматическая генерация кода снижает количество ручного написания кода, минимизирует ошибки при реализации API-клиентов и обеспечивает единообразие в стиле кода.
2. Типобезопасность позволяет компилятору проверять корректность типов, что ведет к раннему обнаружению ошибок.
3. Интеграция с JSON сериализацией обеспечивает автоматическую конвертацию JSON в Dart объекты, поддерживает сложные модели данных и упрощает работу с API.
4. Поддержка различных HTTP методов включает GET, POST, PUT и DELETE, а также поддержку query параметров.

Для управления навигацией в приложении применяется библиотека AutoRouter. У данного подхода есть свои положительные стороны:

1. Компиляция проверяет существование маршрутов, а код маршрутизации генерируется автоматически. Это помогает разработчику предотвратить ошибки при навигации.
5. Поддерживаются глубокие ссылки, вложенные маршруты. Сохраняется состояние навигации.
6. Интеграция с dependency injection: автоматическое внедрение зависимостей, управление жизненным циклом экранов; передача параметров между экранами.

7. Поддержка анимаций включает кастомные анимации переходов, параллельную навигацию и управление стеком навигации.

4. JWT аутентификация в ViChat

Бэкенд-часть ViChat построена на Spring Boot, что обеспечивает надежность, безопасность и масштабируемость приложения. Особое внимание уделено реализации JWT аутентификации.

На сервере JWT реализован с использованием Spring Security. При успешной аутентификации генерируется токен, содержащий ID пользователя и его роли. Токен подписывается алгоритмом HS512 и имеет ограниченный срок действия. Каждый входящий запрос проверяется на валидность токена, его подпись и срок действия.

На клиенте токены хранятся в защищенном хранилище (EncryptedSharedPreferences для Android, Keychain для iOS). При каждом запросе токен автоматически добавляется в заголовок Authorization. Система отслеживает срок действия токена и автоматически запрашивает новый при необходимости.

Безопасность обеспечивается несколькими механизмами:

– Надежный алгоритм подписи (HS512);
– Ограниченный срок действия токенов;
– Механизм обновления токенов.

Для оптимизации производительности используется кэширование токенов как на клиенте, так и на сервере. Система обновления токенов минимизирует количество запросов к серверу, запрашивая новый токен только при окончании срока действия текущего.

5. Анализ решений для транскрибации видеосообщений в ViChat

Для реализации транскрибации видеосообщений в ViChat была выбрана библиотека SpeechFlow[5]. Это решение обеспечивает высокую точность распознавания речи и поддержку множества языков. SpeechFlow предлагает простой API для интеграции и оптимизированную работу с аудиопотоками.

Выбор SpeechFlow обусловлен несколькими факторами. Библиотека предоставляет готовые решения для обработки аудио и распознавания речи, что ускоряет разработку. Она имеет хорошую документацию. Кроме того, SpeechFlow оптимизирована для работы на мобильных устройствах, что соответствует требованиям ViChat.

Система транскрибации в ViChat построена на основе SpeechFlow. Она включает компоненты для обработки аудио и распознавания речи. Архитектура обеспечивает гибкость и масштабируемость системы, позволяя легко добавлять новые функции и оптимизировать существующие.

Процесс транскрибации представлен на диаграмме последовательностей (рис. 3) и начинается с получения видеосообщения от пользователя. Система извлекает аудиодорожку и передает ее в SpeechFlow для распознавания. Результат распознавания сохраняется и становится доступным пользователю. SpeechFlow автоматически определяет язык сообщения и применяет соответствующие модели распознавания.

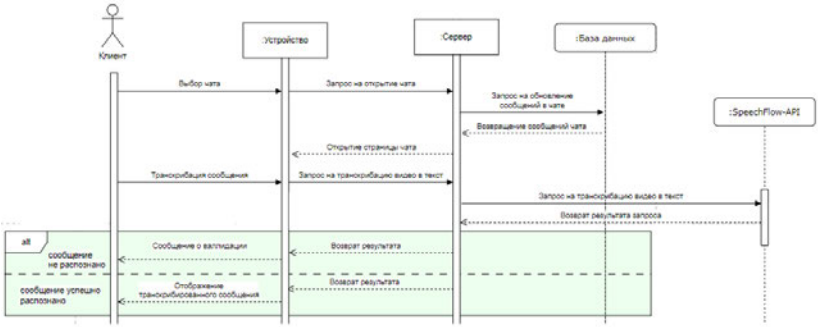


Рис. 3. Диаграмма последовательностей, описывающая процесс транскрибации

6. Демонстрация продукта

В ViChat реализованы следующие принципы дизайна:

– минимализм и простота;
– интуитивная навигация;
– быстрый доступ к основным функциям;
– консистентность интерфейса.

На рис. 4 отображены экраны смартфона в процессе отправки сообщения пользователем.

Заключение

Выбранный стек технологий позволил разработать продукт, который соответствует всем требованиям и решает задачи, которые были поставлены перед разработкой. Помимо этого, выбранные фреймворки позволили создать продукт в кротчайшие сроки, что является важным фактором разработки, а использование готового API SpeechFlow позволило внедрить главное преимущество мессенджера ViChat – транскрибацию видеосообщений в текст.

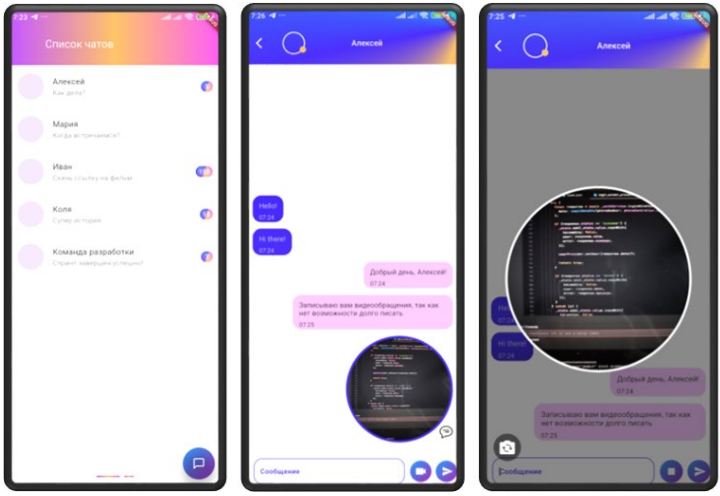


Рис. 4. Демонстрация основной функциональности (отправки сообщения)

Список литературы

1. Mediascope: Telegram [Электронный ресурс]: Mediascope. – Режим доступа: https://www.sostav.ru/publication/auditoriya-telegram-67586.html

2. Flutter MVP Architecture [Электронный ресурс]: Medium. – Режим доступа: https://medium.com/codex/mvp-architecture-pattern-in-flutter-b7bfe712a0ba

3. Spring MVC — основные принципы [Электронный ресурс]: Хабр. – Режим доступа: https://habr.com/ru/articles/336816/

4. Уоллес, К. Spring в действии / К. Уоллес. – 1-е изд. – М.: Питер, 2021. – 432 с.

5. Точный перевод речи в языков [Электронный ресурс]: SpeechFlow. – Режим доступа: https://speechflow.io/ru

Информационная система оценки особенностей управления оператором беспилотными летательными аппаратами

Н. Д. Солопанов

Студент бакалавр

Я. А. Туровский

Профессор

Введение

Беспилотные летательные аппараты (БПЛА) активно внедряются в различные сферы деятельности, включая военную, гражданскую, агропромышленную и транспортную. Значительная часть БПЛА управляется не автономно, а операторами, что обуславливает актуальность более глубокого изучения деятельности оператора в процессе управления БПЛА [1]. Перед оператором БПЛА стоят различные задачи: выполнение разведывательных и поисково-спасательных операций, доставка грузов, аэрофотосъемка, мониторинг сельскохозяйственных угодий и обеспечение безопасности. Кроме того, от оператора может требоваться точное пилотирование в условиях ограниченной видимости, выполнение автономных маршрутов и оперативное реагирование на изменения окружающей среды. Очевидно, что выполнение данных задач требует от оператора разных свойств характеристик управления, включая быстроту реакции, пространственное восприятие, способность к принятию решений в условиях неопределенности, а также владение специализированными интерфейсами управления. Ошибки в управлении, вызванные психологическими, физическими или эмоциональными факторами, могут привести к снижению эффективности выполнения задачи или даже к потере летательного аппарата. В связи с этим актуальной является задача разработки информационной системы, позволяющей комплексно оценивать особенности управления БПЛА, учитывая влияние различных факторов на эффективность управления. Такая система может быть использована как для анализа и повышения качества подготовки операторов, так и для совершенствования самих БПЛА с точки зрения функциональности их интерфейсов управления.

© Солопанов Н. Д., Туровский Я. А., 2025

Целью настоящей работы является создание информационной системы для оценки особенностей управления БПЛА с учетом факторов, влияющих на взаимодействие человека оператора и дронов.

1. Задачи, возникающие при использовании БПЛА

При выполнении задач в условиях ограниченного пространства, связанного с промышленными предприятиями, лесонасаждениями, плотной застройкой зданий, от оператора требуется активное управление дроном и быстрая оценка ситуации. Таким образом, основные элементы поведения оператора в задаче можно смоделировать путем постановки следующего модельного эксперимента — пролет через ворота. Суть модельного эксперимента заключается в следующем: оператору необходимо провести дрон через три ряда специальных рамок (ворот) в минимальное возможное время, ворота в каждом ряде имеют разные размеры, что требует от оператора высокой точности управления, пространственного мышления и умения адаптироваться к изменяющимся условиям.

Следующая задача, встающая перед оператором — выполнение сложных разведывательных работ и полетов внутри помещений, когда пилотирование вдоль продольной оси невозможно или затруднено из-за особенностей конфигурации пространства. В этом случае в информационной системе моделируется задача близкая к рассмотренной ранее, но с изменениями в формате прохождения ворот. В отличие от стандартного полета, перед воротами оператору необходимо поставить БПЛА «боком», сохраняя ориентацию корпуса аппарата перпендикулярно основному направлению полета.

Важнейшая задача при управлении БПЛА во всех сферах является выполнение точного приземления на указанную точку. Например, в гражданской сфере (доставка грузов, операции в зоне чрезвычайных ситуаций) или для специализированных сценариев (приземление на движущиеся объекты, платформы с ограниченным пространством). При отсутствии навыка точного приземления возникает как угроза здоровья оператора, так и угроза поломки БПЛА из-за некорректно выполненной посадки. В информационной системе моделируется эксперимент точного приземления на указанную точку. Суть эксперимента заключается в следующем: оператору необходимо выполнить управляемое снижение и обеспечить приземление БПЛА в центр заранее заданной мишени, расположенной на полу. При этом оценивается удаленность посаженного дрона от изначальной цели.

Управление дроном с ограниченной визуальной обратной связью — задача, рассмотренная в статье [2]. Такие задачи возникают в военной сфере, когда канал видеосвязи работает с перебоями и не позволяет в

условиях реального времени получать кадры с камеры БПЛА. В гражданской же сфере возникает необходимость полета БПЛА в условия задымленности или плохой видимости. Суть модельного эксперимента заключается в следующем: оператору необходимо провести БПЛА по заданному маршруту, выполняя взлет и точную посадку в контрольных точках. Камера дрона включается (на 5 секунд) только в момент начала испытания и при взлете с контрольной точки, в остальное время визуальный контроль отсутствует, оператор управляет дроном «вслепую», без возможности непосредственного наблюдения за аппаратом.

Выполнение полетов по замкнутой круговой траектории вокруг опор — следующая задача, для которой реализована модель в информационной системе. Такая задача возникает при полете через вертикальные сооружения (мосты, опоры линий электропередач, разрушенные здания). В информационной системе задача моделируется следующим образом: оператору необходимо провести БПЛА по круговой траектории вокруг двух вертикальных опор, установленных на определенном расстоянии друг от друга. Задача выполняется на время, что создает дополнительные требования к скорости и точности выполнения маневров.

2. Программная реализация информационной системы

Для объективной оценки характеристик управления БПЛА требуется специальный инструмент, который будет фиксировать ключевые параметры деятельности оператора, учитывать влияние внешних и внутренних факторов и предоставлять результаты в удобном для анализа виде. Решением данной задачи является информационная система, позволяющая систематизировать изложенные выше эксперименты и производить сбор данных на каждом из них.

При разработке информационной системы в качестве платформы был использован беспилотник DJI Ryze Tello квадрокоптерного типа. Программная реализация выполнена на языке Python. Для организации взаимодействия с БПЛА использована открытая библиотека djitellopy, с ее помощью реализовано подключение к дрону через Wi-Fi, отправка команд на выполнение маневров и получение телеметрических данных с камеры и датчиков дрона. Это позволило интегрировать управление дроном непосредственно в информационную систему и обеспечить стабильное выполнение модельных экспериментов. В качестве среды для создания графического интерфейса использована библиотека PyQt. Обработка видеопотока с камеры дрона производится с помощью библиотека OpenCV. С ее помощью реализована передача изображения в реальном времени в окно интерфейса PyQt.

Информационные потоки в системе разделены на три уровня: оператор — программная часть — БПЛА. Взаимодействие между ними осуществляется посредством Wi-Fi, графического интерфейса и аппаратных средств. На рис. 1 изображена схема потоков данных в приложении.

Рис. 1. Схема потоков данных

Информационная система реализована в виде набора взаимосвязанных модулей и классов, каждый из которых отвечает за выполнение конкретных функций. На рис. 2 изображена диаграмма классов.

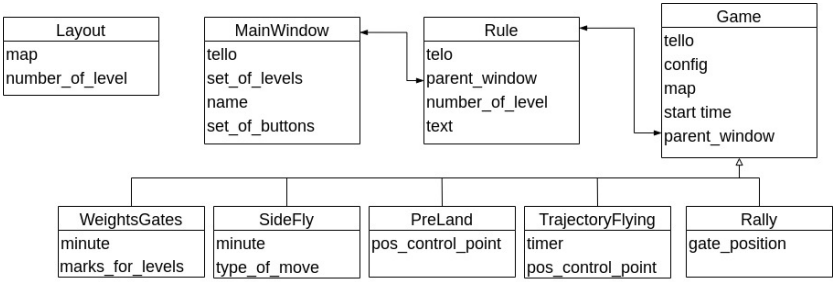


Рис. 2. Диаграмма классов

Для каждого эксперимента формируются логи управления в виде .txt и .mp4 файлов, содержащих информацию о текущем состоянии дрона в момент инициализации эксперимента, передаваемых командах и полную запись с момента начала эксперимента до его завершения. Для каждого испытуемого формируется отдельная папка, в которую для каждого пройденного уровня записываются текстовые и видео логи.

Класс MainWindow выступает в качестве стартового окна приложения и является отправной точкой взаимодействия пользователя с системой. На рис. 3 показано стартовое окно приложения.

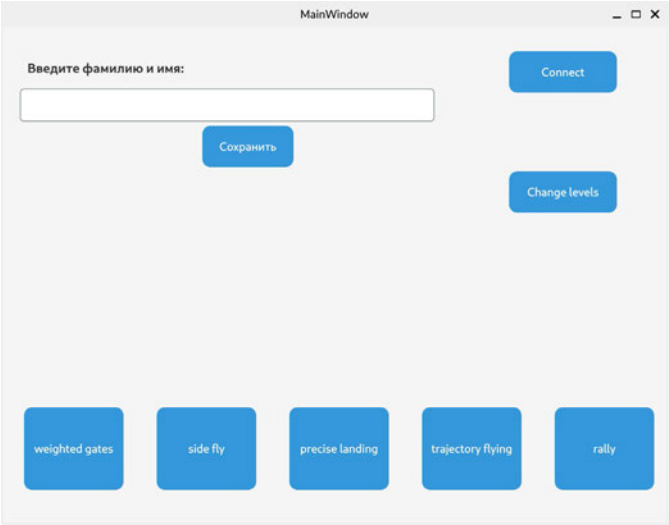


Рис. 3. Стартовое окно приложения

В верхней части стартового окна расположено текстовое поле для ввода фамилии и имени оператора. Введенное значение используется для идентификации пользователя и последующего сохранения данных с экспериментов. При нажатии кнопки «Connect» открывается диалоговое окно, предназначенное для ввода IP-адреса и порта подключения к БПЛА. Данное окно позволяет оператору при необходимости вручную задать параметры подключения к сети беспилотника.

Кнопка «Change levels» открывает выпадающее окно, в котором при необходимости можно изменить конфигурацию экспериментов (расположение «ворот», контрольных точек). В нижней части стартового экрана находится панель выбора экспериментов. При нажатии на одну из пяти кнопок открывается окно с правилами выбранного эксперимента. За генерацию окна с правилами отвечает класс Rule, через это окно осуществляется переход к запуску эксперимента. Далее, при нажатии кнопки «OK» формируется основное окно эксперимента. Для каждого эксперимента базовым классом, формирующим окно, является класс Game. Он отвечает за создание интерактивной карты местности, вывод изображения с камеры дрона, запись логов управления (включая команды оператора, состояние дрона и видеопоток).

От базового класса наследуются специализированные классы, реализующие различные сценарии модельных экспериментов. Класс

WeightsGates реализует эксперимент с пролетом ворот на время. SideFly по структуре аналогичен классу WeightsGates, но имеет отличие в сценарии движения. Класс PreLand формирует окно эксперимента моделирующего посадку дрона на заранее заданную точку. Класс TrajectoryFlying создает окно для эксперимента с полетом по заданной траектории и посадкой в заданных контрольных точках с ограничением доступа к видеопотоку с камеры дрона. Отличительной особенностью классов PreLand и TrajectoryFlying является всплывающее окно при посадке, в которое вносится число — расстояние от центра дрона до указанной контрольной точки. На рис. 4 изображено окно приложения для эксперимента, реализуемого классом TrajectoryFlying.

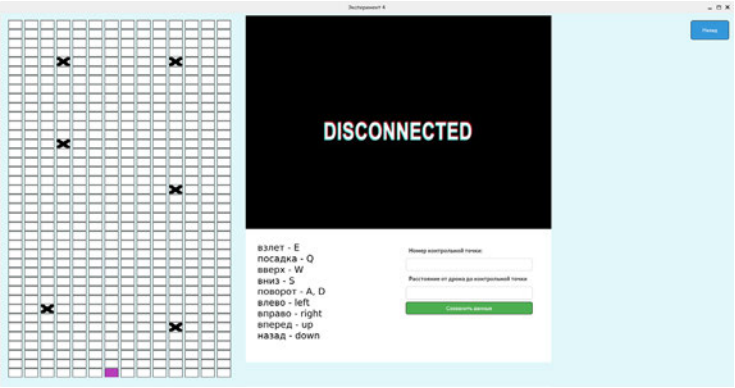


Рис. 4. Окно эксперимента полета по заданной траектории с виджетом записи расстояния

В левой половине экрана находится интерактивная карта, в которой фиолетовым прямоугольником показана стартовая точка, на которую необходимо поставить дрон перед экспериментом. Черные крестики — контрольные точки для приземления. В центре окна находится видеопоток с камеры дрона обрабатывающийся с помощью opencv, в правом верхнем углу находится кнопка выхода из эксперимента. Класс Rally формирует окно для эксперимента — полета по маршруту с ориентацией по воротам.

Информационная система интегрирована в экосистему человеко-машинных интерфейсов лаборатории медицинской кибернетики. Ключевую роль в организации взаимодействия между компонентами экосистемы играет NMServer, выступающий в качестве точки встречи для подключаемых интерфейсов. Через него происходит передача команд управления от различных источников в

информационную систему. В состав подключаемых интерфейсов [3] входят: миографический интерфейс; нейрокомпьютерный интерфейс (например, с помощью программы); окулографический интерфейс; речевой интерфейс. С помощью информационной системы осуществляется прием управляющих команд, поступающих от разных человеко-машинных интерфейсов, выполнение соответствующих действий дроном и фиксация результатов полета. Реализованная информационная система может работать и в локальном формате, тогда управление дроном происходит посредством нажатия кнопок на клавиатуре. Такой подход позволяет оценивать эффективность и особенности управления в зависимости от типа используемого интерфейса и индивидуальных характеристик оператора. На рис. 5 изображена схема описанной выше экосистемы.

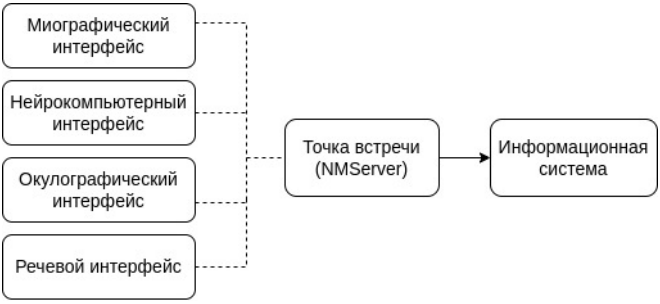


Рис. 5. Схема экосистемы

Разработанная информационная система представляет собой универсальную платформу для оценки операторов БПЛА. Потенциал данной системы открывает широкие возможности для дальнейшего развития и масштабирования. В перспективе внедрение модулей автоматической обработки видеопотока с использованием компьютерного зрения и алгоритмов машинного обучения, что позволит автоматически определять пройденные ворота, точность посадки и другие параметры устраняя необходимость ручного ввода и анализа данных. Структура системы позволяет легко дополнять систему новыми сценариями экспериментов и допускает расширение функционала для поддержки других моделей беспилотников.

Заключение

В данной работе была разработана и реализована информационная система оценки особенностей управления беспилотными летательными аппаратами. Система представляет собой программно-аппаратный

комплекс, включающий в себя графический интерфейс, инструменты управления дроном и модуль для сбора данных, характеризующих действия оператора в модельных экспериментах. Информационная система интегрирована в экосистему лаборатории медицинской кибернетики и поддерживает прием команд с разных человеко-машинных интерфейсов: миографического, окулографического, нейрокомпьютерного и речевого. Разработанная система может быть применена в образовательных учреждениях, исследовательских лабораториях и центрах подготовки операторов БПЛА в качестве тренажера и платформы для проведения экспериментов. Техническая направленность решения охватывает область робототехники, человеко-машинных интерфейсов, нейротехнологии и кибернетики. В дальнейшем возможна модернизация системы за счет автоматизированного сбора результатов, добавления новых экспериментов и поддержки новых типов БПЛА. Это позволит ускорить процесс анализа и повысить точность оценки операторов и расширить область применения в прикладных задачах управления автономными системами.

Список литературы

1. Тюкалов, Д. Е. Моделирование и подготовка операторов транспортных эргатических систем / Д. Е. Тюкалов, А. М. Данилов // Молодой учёный. – 2015. – №3 – С. 247-249.

2. Turovsky, Ya. A. Sensory Characteristics of Operators in Tasks Involving Control of Ergatic Systems without Visual Feedback / Ya. A. Turovsky, V. Yu. Alekseev, R. A. Tokarev // Neuroscience and Behavioral Physiology. – 2024. – Vol. 54. – No. 7. – PP. 1026–1034.

3. Туровский, Я. А. Концепция использования биологических интерфейсов в управлении техническими устройствами / Я. А. Туровский, Д. А. Бирин, Р. В. Мещеряков // Перспективные системы и задачи управления. Материалы XVII Всероссийской научно-практической конференции и XIII молодёжной школы-семинара. Таганрог, 2022. – 2022. – С. 137–139.

Использование Prometheus для обеспечения безопасности

А. А. Танаев

Студент бакалавр

К. А. Маковий

Доцент

Введение

В современном мире информационные технологии играют ключевую роль в развитии бизнеса, науки и государственных структур. Распределенная информационная система (РИС) – информационная система, объекты данных и/или процессы которой физически распределяются на две или более компьютерные системы [1].

РИС широко внедряются благодаря их способности обрабатывать и хранить большие объемы данных, а также обеспечивать высокий уровень отказоустойчивости и масштабируемости. Однако, с увеличением сложности и распределённости таких систем, возрастает и риск безопасности. В этом контексте особое внимание привлекает инструмент для обеспечения мониторинга Prometheus, который предоставляет мощные возможности для наблюдения за состоянием и производительностью компонентов РИС.

Основными требованиями к системе мониторинга, обеспечивающей безопасность РИС, являются непрерывный сбор и анализ событий, выявление аномалий и автоматическое реагирование на угрозы. Основной проблемой при построении такой системы является уникальность каждой рассматриваемой РИС, а также различные требования по обеспечению безопасности, предъявляемые конкретной системе. В результате универсального решения на данный момент не существует. Исходя из вышеперечисленных проблем возникает необходимость конфигурирования системы мониторинга, обеспечивающей достижение целей информационной безопасности для РИС.

1. Обзор существующих решений.

Для обеспечения мониторинга и анализа событий безопасности в информационных системах обычно применяются SIEM системы.

© Танаев А. А., Маковий К. А., 2025

SIEM (Security Information and Event Management) системы предназначены для сбора, приведения к единому виду и корреляции событий в автоматизированных системах и информационной инфраструктуре предприятия, что позволяет обнаруживать подозрительные события и возможные инциденты, исходя из предположений о поведении злоумышленников [2].

Прежде чем перейти к выбору конкретной системы мониторинга, необходимой для осуществления поставленных задач, нужно проанализировать преимущества и недостатки существующих SIEM-систем:

Kaspersky Unified Monitoring and Analysis Platform (KUMA) [3]. Данная система основывается на микросервисной архитектуре, обладает высокой производительностью, а также позволяет интегрировать различные инструменты. Основными недостатками являются ограничения при использовании продуктов, не являющихся частью экосистемы Kaspersky, а также ограниченность документации по разработке собственных модулей и плагинов.

IBM QRadar SIEM. Данный продукт использует масштабируемую модульную архитектуру, задействует искусственный интеллект для автоматизации корреляции инцидентов, а также автоматизации анализа. Поддерживает интеграцию с внешними источниками. Данное программное обеспечение имеет следующие недостатки: стоимость лицензии, которая может быть высокой для организаций с ограниченным бюджетом.

Splunk SIEM. Данная платформа является более универсальной, позволяет анализировать не только данные об угрозах и инцидентах, но также позволяет индексировать любые типы данных и анализировать их с использованием технологий машинного обучения. В системе доступен встроенный графический интерфейс, который позволяет вручную просматривать и настраивать различные дешборды. Основным недостатком являются высокая стоимость при больших объёмах данных.

Существенной проблемой всех вышеперечисленные системы является их стоимость. В разрабатываемой SIEM системе используются исключительно бесплатные приложения, что позволяет снизить расходы на закупку лицензии. Также для реализации SIEM системы предложен микро-сервисный подход, что позволит масштабировать систему.

Для реализации мониторинга распределенных информационных систем существует множество инструментов таких, как Zabbix, Nagios, Icinga и другие. Основными преимуществами Prometheus в сравнении с конкурентами являются:

– наличие мощного языка запросов PromQL;

– обширная поддержка различных систем и платформ разработки;
– интеграция с многими инструментами визуализации;
– поддержка временных рядов.

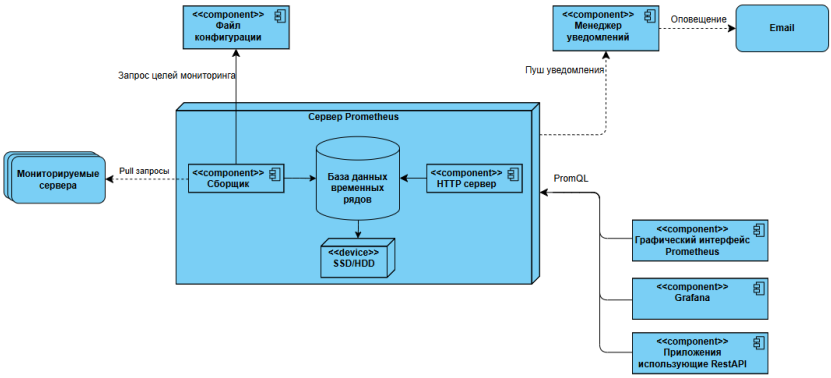


Рис. 1. Схема работы Prometheus

На рис. 1. представлена схема работы Prometheus, а также различных подключаемых к ней компонентов. Prometheus используя Pull запросы собирает с заданных устройств метрики, которые в дальнейшем можно просматривать через внутренний интерфейс Prometheus или с использованием внешнего интерфейса, например Grafana, взаимодействие с которыми обеспечивает HTTP сервер, а также язык запросов PromQL. Также Prometheus поддерживает отправку уведомления через AlertManager. Поскольку Prometheus не предоставляет инструменты для анализа данных, необходимо обращаться к базе данных для извлечения собранных ранее метрик для последующего анализа, для этого применяются библиотеки, которые используют REST API.

Для обеспечения безопасности системы необходимо обосновать выбор метрик, позволяющих отслеживать подозрительную активность.

2. Виды атак на РИС

Одной из задач нашей работы является сбор метрик для последующего обнаружения сетевых атак; для своевременного противодействия. Рассмотрим основные виды сетевых атак.

Атаки на транспортные протоколы, TCP-атаки [4]:

1. затопление SYN-пакетами – суть атаки в создании множества неостановленных соединений, путем отправки SYN-пакетов. Сервер при получении данного пакета выделяет ресурсы для нового соединения

в соответствии с протоколом и отвечает клиенту пакетами ACK и SYN, и ожидает от клиента завершающий пакет ACK;

2. подделка TCP-сегмента – атака состоит в генерации TCP-сегментов, все атрибуты которых имеют значения, легитимные для некоторого существующего сеанса атакуемой системы;

3. сброс TCP-соединения используется для разрыва существующего соединения легальных пользователей. Для проведения атаки нарушитель должен подделать заголовок TCP-сегмента с флагом RST.

Атаки с использованием протокола ICMP [4]:

4. атака перенаправлением трафика. В соответствии с протоколом ICMP маршрутизатор посылает хосту непосредственно присоединенной локальной сети ICMP-сообщение о перенаправлении маршрута при отказе этого маршрута или в тех случаях, когда обнаруживает, что для некоторого адреса назначения хост использует нерациональный маршрут;

5. ICMP-атака Smurf. Атака строится на возможности отправки эхо-запроса не только по индивидуальному, но и по широковещательному адресу некоторой сети. В ICMP-атаке Smurf используется характерный прием – усиление атаки за счет отражения посланного пакета большим количеством компьютеров;

6. «пинг смерти» – Атака состоит в отправке на информационную систему эхо-запроса в IP-пакете, длина которого превышает его допустимый размер, составляющий, согласно стандарту, 65 535 байт;

7. ping-затопление – является достаточно простой – злоумышленник использует утилиту ping своей ОС для отправки эхо-запросов на атакуемый компьютер с максимально возможной частотой. Если быстродействие сетевого интерфейса его компьютера выше, чем у атакуемого компьютера, то атака удается.

Для обнаружения данных сетевых атак предложены следующие метрики:

8. invalidPackets – подсчитывает количество повреждённых пакетов;

9. tcpRequestTotal – собирает общее количество TCP пакетов приходящих на устройство;

10. tcpSYNFlood – агрегирует количество пакетов с SYN флагом с фильтрацией по IP адресу отправителя. Счётчик увеличивается если количество пакетов с SYN флагом превышает заранее определенный порог;

11. tcpInvalidChecksum – регистрирует количество входящих TCP пакетов с неверной контрольной суммой;

12. tcpRSTAttack – отслеживает количество пакетов с RST флагом;

13. tcpSuspiciousFlags – собирает количество TCP пакетов с нетипичной комбинацией флагов;

14. icmpRequestTotal – подсчитывает общее число ICMP запросов на сервер;

15. icmpRedirect – регистрирует количество пакетов с флагами перенаправления;

16. icmpSmurfAttack – метрика регистрирует количество ICMP запросов по широковещательному адресу;

17. icmpPingOfDeath – метрика отслеживающая ICMP пакеты с очень большой длиной;

18. icmpPingFlood – отслеживает количество ICMP пакетов с учётом IP адреса источника и увеличивается при превышении определенного порога.

В приведенной ниже таблице представлены типы атак, которые позволяет обнаруживать конкретная метрика.

Таблица

Метрики обнаружения атак

Метрика	**Тип атаки**
invalidPackets	Затопление SYN пакетами, Подделка TCP сегмента, Сброс TCP
tcpRequestTotal	Затопление SYN пакетами, Подделка TCP сегмента, Сброс TCP
tcpSYNFlood	Затопление SYN пакетами
tcpInvalidChecksum	Затопление SYN пакетами, Подделка TCP сегмента, Сброс TCP соединения
tcpRSTAttack	Сброс TCP соединения
tcpSuspiciousFlags	Подделка TCP сегмента
icmpRequestTotal	Атака перенаправлением трафика, ICMP Smurf атака, Ping затопление
icmpRedirect	Атака перенаправлением трафика
icmpSmurfAttack	ICMP-атака Smurf
icmpPingOfDeath	«Пинг смерти»
icmpPingFlood	Ping затопление

Сетевые атаки представляют серьёзную угрозу для РИС, поскольку позволяют злоумышленникам нарушать стабильность соединений, перехватывать сессии или выводить систему из строя. Своевременное обнаружение и противодействие данным атакам позволят повысить

защищённость системы, для этого необходимо собирать и анализировать соответствующие метрики.

3. Методика исследования

Для изучения возможностей реализации SIEM системы с использованием Prometheus в среде EVE-NG развёрнута упрощённая модель реально существующей РИС, обеспечивающей работу автомоек самообслуживания.

Текущая топология сети представлена на рис. 2.

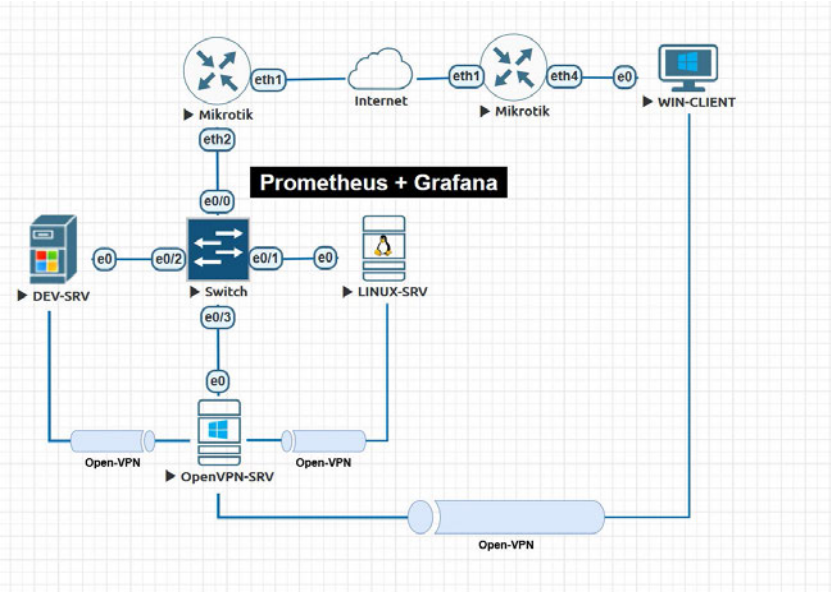


Рис. 2. Топология виртуальной сети

На машине DEV-SRV под управлением Windows Server 2012 R2 развернуто ASP.Net Core приложение, использующее Internet Information Services, которое моделирует работу реального сервера, который собирает данные от клиентских компонент РИС, смоделированных с помощью машины Win-Client.

В реальной системе серверная часть опрашивает клиентские модули для получения от них информации о проведенных операциях и состоянии исполнительных механизмов. Но клиентская часть не подразумевает наличие белого IP-адреса, поэтому взаимодействие с серверной реализовано через использование OpenVPN. При

использовании OpenVPN клиентская часть на первом этапе устанавливает OpenVPN туннель с OpenVPN сервером, в результате чего создается виртуальная частная сеть, в которой работают все компоненты РИС. Сервер SRV-DEV также является клиентом OpenVPN.

Для мониторинга этой системы нами предложено использовать Prometheus в связке с Alertmanager и Grafana. Для этого на сервере Linux-SRV развернуты соответствующие программные средства.

Использование Prometheus для мониторинга РИС подразумевает установку агента на каждый компонент РИС. На данный момент на DEV-SRV внедрён агент Web Application.

В ходе разработки были реализованы метрики, описанные ранее. На рис. 3. представлены результаты сбора данных метрик на экране окна Grafana, который позволяет объединить и вывести сразу несколько графиков отслеживаемых метрик. Архитектура сервиса, собирающего метрики на конечных устройствах, позволяет быстро добавлять и изменять процесс вычисления той или иной метрики, что упрощает процесс разработки.

Рис. 3. Экран окна Grafana с отображением собранных метрик

Собранные метрики в дальнейшем будут подвергаться анализу с применением технологий машинного обучения. В случае выявления аномалий с использованием AlertManager будут отправляться уведомления на заранее заданные почтовые или иные адреса. А также в случае заранее заданной последовательности действий будет запускаться процесс противодействия возникшей угрозе.

Заключение

Проведённое исследование демонстрирует возможность использования Prometheus в качестве основы для SIEM системы в распределённых информационных системах. Развёрнутая модель показала, что даже на базе бесплатных инструментов можно создать гибкую и масштабируемую систему мониторинга, собирающую данные, которые позволяют обнаружить атаки на сетевую инфраструктуру.

Список литературы

1. ГОСТ 34.321–96. Информационная технология. Взаимосвязь открытых систем. Основы взаимосвязи. Протокол маршрутизации для передачи данных по виртуальному каналу [Текст]. – Введ. 1997–07–01. – М. : Госстандарт России, 1997. – 45 с.
2. Герчин, И. А. Использование средств мониторинга информационной безопасности на базе открытых решений / И. А. Герчин, А. Г. Анацкая // Цифровизация и кибербезопасность: современная теория и практика: сб. материалов IV Междунар. науч.-практ. конф. (Омск, 30–31 октября 2024 г.). – Омск: СибАДИ, 2025. – С. 404-412.
3. SIEM-система Kaspersky Unified Monitoring and Analysis Platform [Электронный ресурс]. – Режим доступа: https://go.kaspersky.com/ru-kuma
4. Олифер, В. Г. Компьютерные сети: принципы, технологии, протоколы: учебник / В. Г. Олифер, Н. А. Олифер. – 5-е изд. – СПб. : Питер, 2016. – 944 с.
5. Пивотто, Ж. Запускаем Prometheus: мониторинг инфраструктуры и приложений / Ж. Пивотто, Б. Бразил ; пер. с англ. А. Н. Киселева. – М. : Books.kz, 2023. – 392 с.
6. Абдрашитов, М. С. Разработка модели SIEM-системы для аудита информационной безопасности в веб-приложении / М. С. Абдрашитов, И. С. Черных // Сила и разум: сб. ст. науч.-практ. конф. (Санкт-Петербург, 13–14 июня 2024 г.) / Военно-медицинская академия им. С. М. Кирова. – СПб. : Военно-медицинская академия, 2024. – С. 92-99.

Разработка мобильного приложения для туристов

А. С. Телегина

Студент бакалавр

В. А. Ушаков

Ассистент

Введение

С развитием транспортных технологий, коммуникации и доступности информации почти каждый человек имеет возможность отправиться в путешествие в любую точку мира. Люди, путешествуя как внутри своей страны, так и за ее пределами, стремятся получить более глубокое понимание истории и культуры региона, посещаемого ими.

Современный туризм и путешествия все больше ориентируются на использование мобильных технологий, ведь они предоставляют возможность просматривать данные в любом месте и в любой момент времени. Мобильные приложения для поиска экскурсий помогают быть открытыми для новых идей, открывают доступ к интересным местам и событиям, повышая уровень комфортности при планировании путешествий. Разработка мобильного приложения для поиска и создания экскурсий представляет собой актуальную и значимую задачу в контексте современных потребностей людей, желающих быть более осведомленными о мире вокруг себя.

1. Средства реализации

Мобильное приложение имеет архитектуру, соответствующую смешанной модели Клиент - Серверного взаимодействия на основе REST API.

Для реализации серверной части приложения использованы следующие средства:

– язык программирования Java;
– фреймворк Spring.

Фреймворк Spring позволяет упростить и ускорить разработку приложений для разработчика. Spring предоставляет каркас будущего приложения. При этом фреймворк диктует правила построения приложения – есть определенная архитектура приложения, в которую нужно встроить свою функциональность.

© Телегина А. С., Ушаков В. А., 2025

– СУБД PostgreSQL.

Данная СУБД является свободно распространяемой и предоставляет функционал аналогичный платным конкурентам. Также PostgreSQL имеет в своей функциональности расширение PostGIS, предоставляющее возможность индексации географических объектов, что является важным в рамках разрабатываемого приложения.

– Docker.

Контейниризатор позволит быстрее и надежнее масштабировать приложения в рамках системы, упаковывая их в отдельные блоки.

Для реализации клиентской части приложения будут использоваться следующие средства:

– язык программирования Dart;

– фреймворк Flutter.

Данный стек технологий был выбран, так как Flutter имеет одинаковый пользовательский интерфейс и бизнес-логику для всех платформ, позволяет сократить время разработки кода, а также есть возможность использовать плагин от Google для обработки разрешений и др.

– OpenTripMap API.

Это API предоставляет доступ к данным OpenTripMap о достопримечательностях и туристических объектах. OpenTripMap объединяет и сопоставляет информацию из глобальных источников открытых данных (OpenStreetMap, Wikidata, Wikipedia, открытые данные Минкультуры и Роснедр РФ) и содержит более 10 миллионов достопримечательностей и туристических объектов по всему миру. Категории объектов иерархически структурированы. Возвращаемые данные зависят от запрашиваемого языка [1].

2. Серверная часть приложения

Серверная часть приложения построена на основе классического многоуровневого архитектурного паттерна [2], состоящего из следующих слоев:

– Controller.

Слой контроллеров отвечает за прием Rest запросов клиента, передачу информации на обработку слою сервисов приложения и возврат соответствующего ответа пользователю.

– Service.

Сервисы представляют собой всю бизнес-логику приложения. Здесь происходит обработка данных, использование уровня репозиториев для доступа к базе данных, а также порождение исключений в процессе работы приложения.

– Repository.

Слой jpa репозиториев позволяет на уровне абстракции работать с базой данных, схема которой представлена на рис. 1. С помощью этого слоя происходит работа с сущностями PostgreSQL.

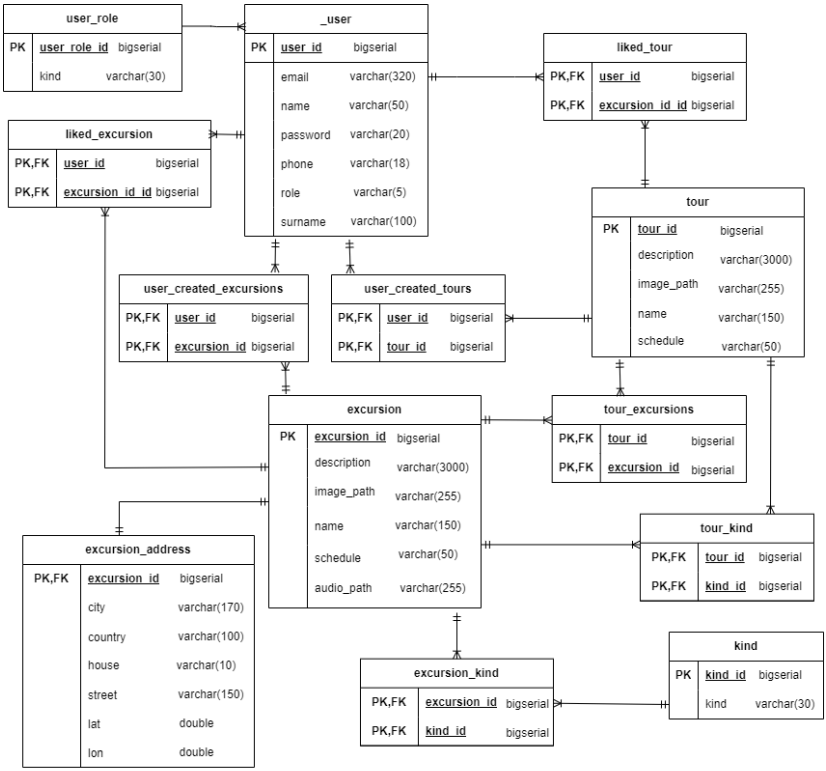


Рис. 1. ER-диаграмма

3. Клиентская часть приложения

При разработке клиентской части приложения применялась BLoC-архитектура. Блок-архитектура (BLoC) во Flutter приложениях является паттерном проектирования, который отделяет бизнес-логику приложения от пользовательского интерфейса. Это помогает улучшить тестируемость, управляемость и разделение ответственности в приложении.

Основные компоненты BLoC-архитектуры:

– BLoCs - неизменяемые объекты, которые содержат состояние (данные) и логику для обновления этого состояния [3].

Действия, такие как нажатия кнопок или сетевые запросы, отправляются в блоки, которые затем обновляют свое состояние и уведомляют слушателей об изменениях.
– Events - неизменяемые сообщения, которые отправляются в блоки для инициирования изменений состояния.
– States - неизменяемые объекты, которые представляют текущее состояние приложения.
– Listeners - компоненты пользовательского интерфейса, которые подписываются на изменения состояния и обновляют свой внешний вид в соответствии с изменениями.

Преимущества BLoC-архитектуры:
– Управляемость. Разделение бизнес-логики и пользовательского интерфейса улучшает управляемость кода и делает его проще для понимания и сопровождения.
– Разделение ответственности. BLoC-архитектура четко разделяет ответственность между различными компонентами приложения, что повышает его согласованность и надежность.
– Предсказуемость. Поскольку BLoC-архитектура управляет состоянием централизованным образом, изменения состояния приложения становятся более предсказуемыми и контролируемыми.

4. Уровень доступа пользователей

В данном приложении существуют два вида пользователей: турист и администратор. Для них определены разные пользовательские сценарии, представленные на рис. 2, в связи с чем, реализован разный уровень доступа. На уровне сервера доступ к контенту проверяется с помощью JWT. Веб-токен JSON (JWT) - это открытый стандарт, который определяет компактный и автономный способ безопасной передачи информации между сторонами в виде объекта JSON [4]. Эта информация может быть проверена и ей можно доверять, поскольку она имеет цифровую подпись. Любой пользователь приложения для начала работы в приложении должен получить случайно сгенерированный при регистрации токен с уровнем доступа туриста, либо получить токен при авторизации с соответствующим уровнем доступа. Данный JWT сохраняется в базе данных. Во всех следующих запросах клиент обязан прикладывать этот токен, чтобы использовать функции приложения. Срок жизни токена ограничен, чтобы уменьшить вероятность кражи токена. В связи с этим применяется два вида токена: access и refresh [5]. Если при получении запроса сервером указан JWT, срок жизни которого уже истек, возвращается код ошибки 403.Данный код говорит о том, что доступ к запрашиваемой странице запрещен или у пользователя нет

прав на просмотр контента. После чего, при наличии refresh JWT, клиент отправляет запрос на обновление токенов, указывая в header запроса данный refresh JWT. Получив новую пару токенов, клиент повторно отправляет свой запрос с новым access токеном.

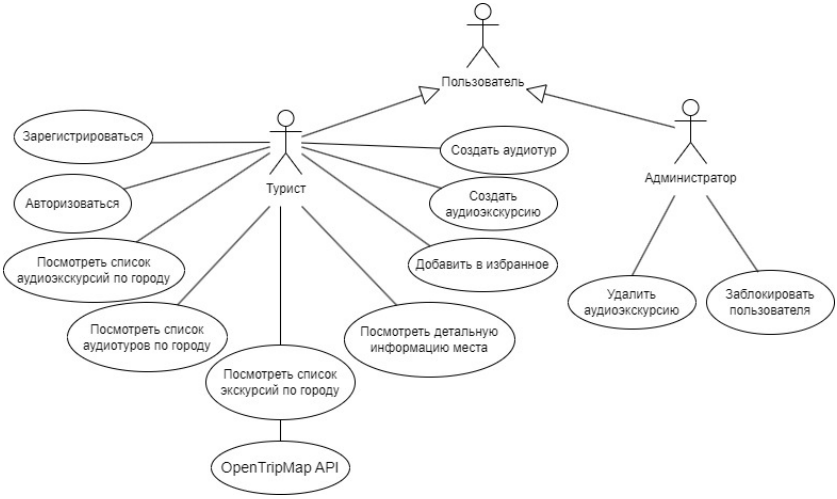


Рис. 2. Диаграмма прецедентов

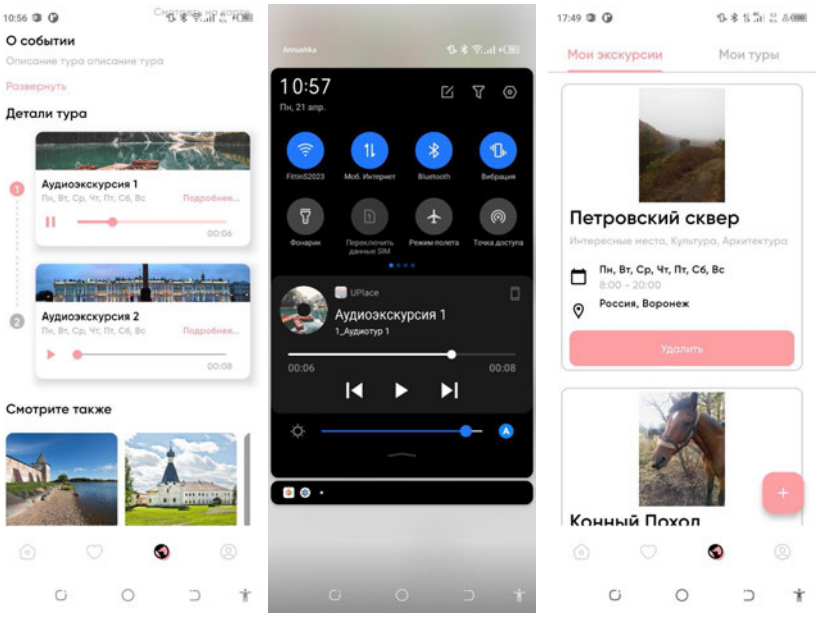
Заключение

В результате было создано приложение для поиска интересных мест и прослушивания экскурсий, а также для добавления своих собственных экскурсий и туров.

Для визуализации приложения были разработаны макеты интерфейса, которые отражают основные элементы дизайна и взаимодействия с пользователем. Был выбран подходящий стек технологий и платформа для разработки приложения, обеспечивающие наилучшую производительность и масштабируемость.

В ходе разработки были реализованы основные функции приложения. Часть функций приложения продемонстрирована на рис. 3.

Разработанное мобильное приложение для поиска экскурсий имеет большой потенциал, например, для повышения интереса людей к культурному наследию города и страны, для привлечения людей в туристические места.



а б в

Рис. 3. Демонстрация работы приложения: а – воспроизведение аудиоэкскурсии, б – воспроизведение аудио в фоновом режиме, в – просмотр списка созданных аудиоэкскурсий и аудиотуров

Список литературы

1. Атлас достопримечательностей для туристов [Электронный ресурс]. – Режим доступа: https://opentripmap.com/ru

2. Многоуровневая архитектура в проекте на Java (Часть 1) [Электронный ресурс]. – Режим доступа: https://alexkosarev.name/2018/07/27/n-tier-java-part1

3. Разделение бизнес-логики и UI во Flutter с помощью BLoC-архитектуры [Электронный ресурс]. – Режим доступа: https://inostudio.com/blog/articles-develop/razdelenie-biznes-logiki-i-ui-vo-flutter-s-pomoshchyu-bloc-arkhitektury

4. Introduction to JSON WebToken [Электронный ресурс]. – Режим доступа: https://jwt.io/introduction

5. Access Token and Refresh Token: A Comprehensive Guide [Электронный ресурс]. – Режим доступа: https://dev.to/tijan_io/access-token-and-refresh-token-a-comprehensive-guide-40g7

Разработка мобильного приложения для домашних тренировок

Д. С. Третьяков

Студент бакалавр

В. А. Ушаков

Ассистент

Введение

В условиях современного образа жизни многие сталкиваются с проблемой недостатка физической активности. Работа за компьютером, передвижение на транспорте и пассивный отдых формируют привычный, но вредный для здоровья распорядок дня. При этом традиционные решения вроде посещения спортзала часто оказываются неэффективными - люди быстро теряют мотивацию, когда тренировки становятся рутиной [1].

Именно здесь может помочь геймификация - подход, который добавляет игровые элементы в обычные процессы [2]. В отличие от стандартных фитнес-приложений с их сухими списками упражнений, геймификация превращает тренировки в увлекательный процесс. Даже простые игровые механики - система уровней, виртуальные награды, визуализация прогресса - способны значительно повысить вовлеченность.

Приложение Dungeon Fit использует эти принципы, предлагая пользователям выбрать персонажа (например, воина, волшебника или рейнджера) и выполнять упражнения в игровом формате [3]. Каждый класс персонажа определяет тип рекомендуемых тренировок, а система автоматически адаптирует сложность заданий в зависимости от физических возможностей пользователя.

Преимущество такого подхода в том, что он делает тренировки более осмысленными. Когда упражнения представлены не как скучная обязанность, а как часть игрового процесса, у людей появляется дополнительная мотивация заниматься регулярно. При этом приложение не требует специального оборудования и подходит для занятий в домашних условиях.

© Третьяков Д. С., Ушаков В. А., 2025

1. Архитектура приложения

Архитектура приложения DungeonFit построена на принципах микросервисной организации с четким разделением функциональных обязанностей между компонентами. Клиентская часть, разработанная на Flutter [4], обеспечивает кроссплатформенную работу и взаимодействует с серверной частью, разработанной на Ruby on Rails [5], через единый Orchestrator Service, выполняющий роль маршрутизатора запросов. Этот центральный узел принимает все входящие запросы от клиента и распределяет их между специализированными сервисами.

Серверная часть состоит из специализированных микросервисов: User Service (профили и авторизация), Workout Service (аналитика тренировок), а также геймификационные сервисы - Quest (задания), Reward (награды) и Achievement (достижения). Отдельный Notification Service отвечает за персонализированные уведомления для повышения вовлеченности.

Сервисы взаимодействуют через REST API и очереди сообщений, каждый использует собственную БД PostgreSQL для независимости и масштабируемости. Такая организация обеспечивает отказоустойчивость и гибкость при добавлении новой функциональности.

Схема взаимодействия компонентов в микросервисной архитектуре представлена на рисунке ниже.

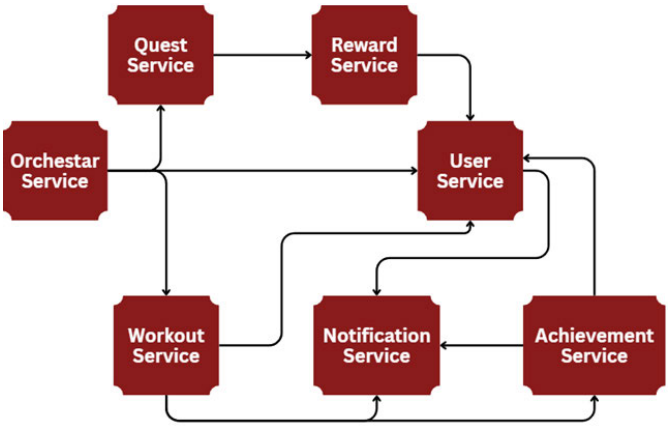


Рисунок. Схема взаимодействия компонентов

Клиентское приложение построено на Flutter с использованием паттерна BLoC. Архитектура разделена на четыре четких слоя для обеспечения модульности и простоты поддержки.

Presentation Layer отвечает исключительно за пользовательский интерфейс, преобразуя данные в визуальное представление без собственной логики. Все экраны и виджеты работают через этот слой, получая готовые к отображению данные.

Business Logic Layer содержит BLoC-компоненты, которые обрабатывают всю бизнес-логику приложения. Здесь управляются состояния тренировок, обрабатываются достижения и награды, реализуется геймификация. Каждый BLoC изолирован и отвечает за конкретную функциональную область.

Data Layer объединяет работу с данными через репозитории, которые абстрагируют источники информации. Этот слой обеспечивает синхронизацию между серверным API и локальным хранилищем, поддерживая офлайн-режим работы. Все преобразования данных происходят именно здесь.

Core Layer включает общие компоненты: конфигурацию, сетевые клиенты, утилиты и базовые расширения. Этот фундаментальный слой используется всеми остальными слоями приложения, обеспечивая единые стандарты работы.

Такая структура позволяет легко модифицировать отдельные компоненты без влияния на всю систему. Например, изменения в API затрагивают только Data Layer, оставляя бизнес-логику и интерфейс неизменными.

2. Формирование персонализированных тренировок

В Dungeon Fit каждая серия тренировок представлена в виде подземелья, где один уровень соответствует одной тренировке. Алгоритм построения этого подземелья активируется сразу после регистрации пользователя или при завершении текущего цикла. На этапе регистрации пользователь указывает, сколько раз в неделю он готов заниматься. Эта цифра напрямую определяет количество уровней в подземелье - например, при желании тренироваться трижды в неделю будет создано подземелье из трёх уровней с днями отдыха между ними.

Каждый уровень подземелья представляет собой сессию упражнений, сгенерированных индивидуально. При этом ключевую роль играет выбранный пользователем класс персонажа. Для воина подбираются преимущественно силовые. Маг получает упражнения на гибкость и контроль тела. Рейнджеру предлагаются кардионагрузки. Это позволяет учитывать предпочтения и цели пользователя уже на стадии выбора архетипа.

Список упражнений для каждого класса загружается из централизованного репозитория, содержащего описания, изображения, уровни сложности и технические параметры. Метод генерации build_level_plan(user, level_index) формирует тренировку на день с учётом физической информации пользователя (веса, роста, опыта и истории прохождения предыдущих уровней). Каждому упражнению назначается количество повторений, количество подходов, а также порядок, в котором они будут выполняться. Чтобы тренировка не была монотонной, система стремится чередовать типы нагрузок: силовые, кардио и статические, обеспечивая тем самым разнообразие и равномерную нагрузку на тело.

Сложность подземелья постепенно нарастает. Первый уровень предполагает лёгкий вход, тогда как последние содержат более интенсивные блоки и могут включать усложнённые этапы с увеличенной нагрузкой. После завершения подземелья или отмены прохождения пользователю предлагается откорректировать параметры: оставить всё как есть, сделать тренировки легче или сложнее, а также изменить периодичность. Это решение сохраняется и влияет на генерацию следующего подземелья, поддерживая гибкую и персонализированную логику тренировочного процесса.

Генерация подземелья реализована следующим образом (листинг 1).

Листинг 1

Генерация подземелья

```
def build_dungeon(user)
  level_count = user.sessions_per_week
  (1..level_count).map do |level_index|
    build_level_plan(user, level_index)
  end
end

def build_level_plan(user, level_index)
  exercises       =
WorkoutRepository.for_class(user.character_class)
  mixed           = mix_exercises(exercises)
  count           = calculate_exercise_count(user)
  selected        = mixed.first(count)

  difficulty = 1 + (level_index - 1) * 0.1

  plan = selected.map.with_index(1) do |exercise, i|
    reps = calculate_reps(exercise, user, count) * difficulty
    sets = calculate_sets(exercise, user, count) * difficulty

    {
```

Окончание листинга 1

```
      step:        i,
      name:        exercise.name,
      description: exercise.description,
      image_url:   exercise.image_url,
      reps:        reps.round,
      sets:        sets.round
    }
  end

  { level: level_index, exercises: plan }
End
```

Количество упражнений внутри уровня определяется на основе рекомендаций ведущих организаций в области спортивной медицины, таких как ACSM и ВОЗ [6]. Базовый набор варьируется в зависимости от архетипа. На этот базовый объём накладываются корректировки в зависимости от физического состояния пользователя. Метод calculate_exercise_count(user) (листинг 2) использует вес, рост, выбранный класс и частоту тренировок в неделю для вычисления индивидуального объёма нагрузки.

Листинг 2

Адаптивный расчёт количества упражнений

```
def calculate_exercise_count(user)
  base = {
    'warrior' => 10,
    'mage'    => 6,
    'ranger'  => 8
  }[user.character_class]

  freq_factor = case user.sessions_per_week
                when 5..7 then 0.9
                when 3..4 then 1.0
                else         1.1
                end

  bmi = user.weight_kg.to_f / (user.height_m ** 2)
  bmi_adjust = if bmi < 18.5
                 -1
               elsif bmi < 25
                 0
               else
                 -2
               end

  [(base * freq_factor).round + bmi_adjust, 1].max
end

def calculate_reps(exercise, user, total_exercises)
```

Окончание листинга 2

```
  base_reps = exercise.base_reps || 10
  adjustment = (12.0 / total_exercises).clamp(0.6, 1.4)
  (base_reps * adjustment).round
end

def calculate_sets(exercise, user, total_exercises)
  case total_exercises
  when 1..4 then 4
  when 5..7 then 3
  else           2
  end
end

def mix_exercises(exercises)
  groups = exercises.group_by(&:type)  # :strength, :cardio,
:static
  mixed  = groups.values.transpose.flatten.compact
  mixed.shuffle
end
```

Заключение

В данной статье рассмотрены ключевые аспекты архитектуры и бизнес-логики мобильного приложения, использующего микросервисный подход и кроссплатформенный клиент на Flutter с использованием паттерна BLoC и принципов чистой архитектуры.

Использование микросервисов и адаптивной генерации тренировок позволяет создать гибкую и масштабируемую систему, ориентированную на удобство и вовлечённость пользователей.

Список литературы

1. Johnson, M. Fitness App Development: Current Trends and Future Directions [Текст] / M. Johnson, L. Smith // International Journal of Mobile Computing. – 2020. – Vol. 12. – Iss. 3. – P. 34-49
2. Williams, R. The Impact of Mobile Fitness Apps on User Engagement and Physical Activity [Текст] / R. Williams // Journal of Health Informatics. – 2018. – Vol. 11. – Iss. 2. – P. 123-135.
3. Геймификация [Электронный ресурс] // Habr. – 2024. – Режим доступа: https://habr.com/ru/articles/874788
4. Flutter и Dart. Официальная документация [Электронный ресурс]. – Режим доступа: https://flutter.su/docs
5. Ruby on Rails. Официальное руководство [Электронный ресурс]. – Режим доступа: https://rusrails.ru
6. Всемирная организация здравоохранения. Глобальные рекомендации по физической активности для здоровья [Текст] / пер. с англ. – Женева : ВОЗ, 2010. – 60 с. – Режим доступа: https://minzdrav.gov.ru/system/attachments/attaches/000/016/375/original/9789244599976_rus.pdf

Определение степени созревания плодов томатов в теплицах с помощью YOLOv5

Н. С. Ульянов

Студент бакалавр

А. В. Швырева

Старший преподаватель

Введение

Определение зрелости плодов томатов в тепличных условиях является важной задачей современного сельского хозяйства. Традиционные методы визуального контроля часто субъективны и требуют значительных трудозатрат. Развитие технологий машинного обучения, в частности методов компьютерного зрения и глубоких нейронных сетей, позволяет автоматизировать процесс оценки зрелости, повысить его точность и снизить производственные издержки.

Данная работа посвящена исследованию и разработке программного модуля для автоматического определения степени зрелости томатов на основе изображений с использованием архитектуры YOLO (You Only Look Once) для детектирования плодов и свёрточной нейронной сети (CNN) для их классификации по трём стадиям зрелости: неспелый, полуспелый, спелый.

1. Обзор используемых методов

YOLO является одной из самых популярных архитектур для задач детектирования объектов на изображениях [1]. Её основное преимущество — высокая скорость обработки, что делает возможным применение в реальном времени. YOLO предсказывает координаты объектов и их классы за один проход по изображению, что позволяет эффективно выделять томаты на изображениях, полученных в условиях теплицы [2-3].

После сегментации и вырезания объектов на изображениях применяется классификация. Для этого используется CNN — свёрточная нейронная сеть, обученная на предварительно размеченном наборе изображений томатов. CNN эффективно извлекает признаки из изображений и позволяет проводить точную классификацию зрелости плодов [4].

© Ульянов Н. С., Швырева А. В., 2025

Также были проанализированы теоретические особенности альтернативных методов классификации: метод опорных векторов (SVM), случайный лес, метод ближайших соседей (k-NN). Однако выбор в пользу CNN был сделан на основании свойств самой архитектуры. Сверточные нейронные сети специально разработаны для обработки изображений и способны автоматически извлекать пространственные признаки, что делает их более подходящими для задач, связанных с визуальным распознаванием. Кроме того, CNN устойчивы к вариациям на изображениях, таким как освещение и ракурс, что особенно важно в условиях теплицы [5].

2. Описание набора данных

Набор данных включает 277 оригинальных изображений, размеченных вручную [6]. Аннотации представлены в формате JSON и содержат координаты ограничивающих рамок (bounding boxes) для каждого томата. Пример исходного изображения представлен на рис. 1.

Рис. 1. Пример изображения из исходного набора данных

После сегментации было получено 2421 объекта, из которых:

– 1593 — неспелые томаты,
– 397 — полуспелые,
– 431 — спелые.

Каждый объект отнесён к одному из трёх классов в соответствии со стадией зрелости. Изображения нормализованы и приведены к размеру 224x224 пикселя для подачи в классификатор. Пример полученных сегментированных изображений представлен на рис. 2.

Рис. 2. Пример изображений сегментированного набора данных

3. Аугментация

Для увеличения разнообразия данных и улучшения обобщающей способности модели была применена аугментация изображений из двух ключевых классов, спелых и полуспелых томатов. Для каждого изображения из целевых классов генерировалось по 3 аугментированных варианта.

Применялись следующие преобразования:

– Поворот на случайный угол до 15 градусов
– Случайный сдвиг по ширине и высоте на 10%
– Случайный сдвиг на 10%
– Случайное масштабирование в диапазоне ±20%
– Горизонтальное отражение (с вероятностью 50%)
– Заполнение новых пикселей по стратегии 'nearest'

В результате был получен сбалансированный набор данных, который улучшит устойчивость модели к вариациям в положении и ориентации объектов. Количество изображения по классам после аугментации представлено на рис. 3.

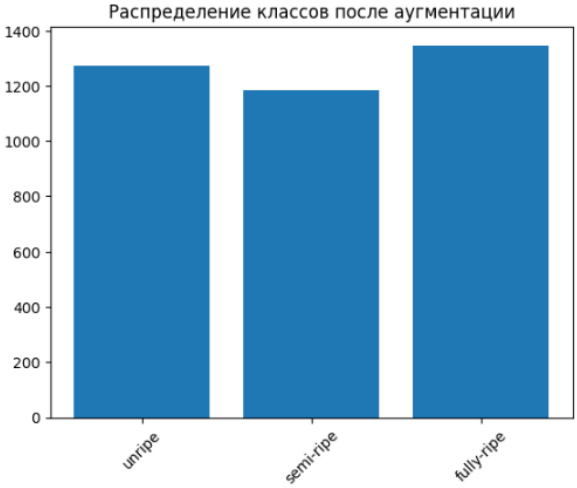


Рис. 3. Диаграмма распределения аугментированных данных

4. Описание алгоритма

Поставленную задачу необходимо разбить на 2 части: сегментация объектов на исходном изображении и классификация детектированных объектов.

Для детектирования объектов будем использовать нейросетевую модель YOLOv5, обученную на размеченных обучающих изображениях.

Для классификации детектированных объектов будем использовать сверточную нейронную сеть, архитектура которой представлена на рис. 4, обученную на 2400 объектах исходного набора данных.

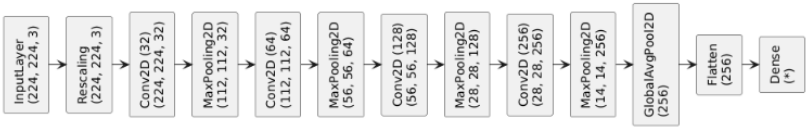


Рис. 4. Архитектура классификатора

5. Результат обучения

Пример работы модели YOLO представлен на рис. 5.

Рис. 5. Пример работы модели YOLO

В процессе обучения модели YOLO были достигнуты следующие метрики:

Таблица

Метрики обученной модели YOLO

Класс	Precision	Recall	mAP@0.5
Все	0.568	0.906	0.755
Неспелые	0.740	0.969	0.953
Полуспелые	0.366	0.779	0.473
Спелые	0.599	0.970	0.839

Метрики показывают малую точность классификации полуспелых и спелых плодов, что можно объяснить сложностью переходной стадии. Также это видно на матрице путаницы на рис. 6.

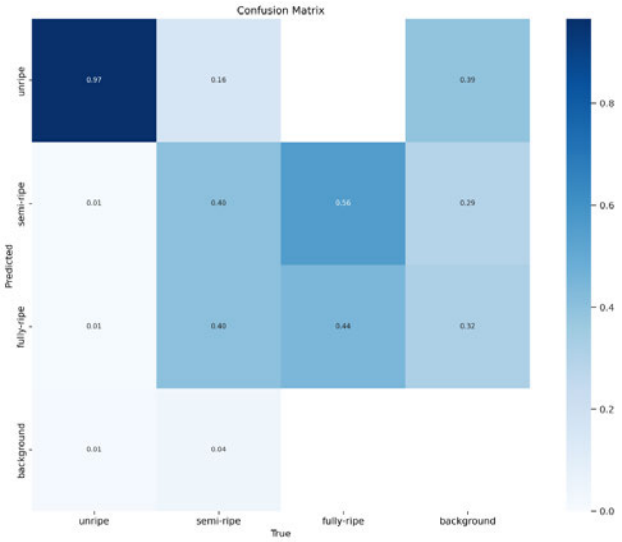


Рис. 6. Матрица путаницы модели YOLO

Для повышения точности классификации отдельно используется классификатор CNN, график обучения которого показан на рис. 7.

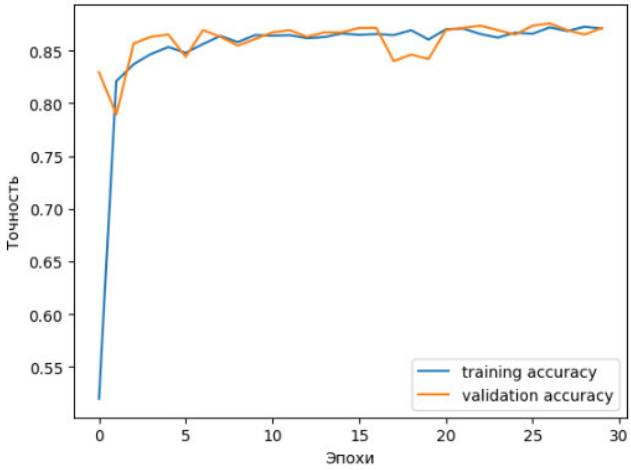


Рис. 7. График точности классификатора на обучающей и валидационной выборке по мере обучения

Классификатор показал следующие результаты точности на тестовой выборке:

- Неспелые: 97.4%
- Полуспелые: 83.9%
- Спелые: 82.6

Заключение

В ходе работы был реализован программный модуль для определения степени зрелости томатов с применением архитектуры YOLO и свёрточных нейронных сетей. Алгоритм успешно выявляет и классифицирует плоды на изображениях, полученных в условиях теплицы, демонстрируя высокие показатели точности.

Разработанное решение может быть использовано для автоматизации процесса оценки зрелости томатов, что способствует повышению эффективности сельскохозяйственного производства.

Список литературы

1. Никитин, Д. В. Детектирование дорожных знаков на основе нейросетевой модели YOLO / Д. В. Никитин, И. С. Тараненко, А. В Катаев // Известия вузов. Проблемы энергетики. — 2023. — №7(103). — С. 73–79.

2. Филичкин, С. А. Применение нейронной сети YOLO для распознавания дефектов / С. А. Филичкин, С. В. Вологдин // Интеллектуальные системы в производстве. — 2022. — Т. 20, №2. — С. 61–67.

3. Полещенко, Д. А. Опыт применения нейронной сети YOLOv5 для детектирования растений подсолнечника / Д. А. Полещенко, В. А. Петров, И. С. Михайлов // Инженерный вестник Дона. — 2022. — №10. — С. 217–228.

4. Свирина А. Р. Классификация изображений с помощью свёрточных нейронных сетей / А. Р. Свирина // Молодой учёный. — 2024. — № 24 (523). — С. 96–99.

5. Маршалко, Д. А. Архитектура свёрточных нейронных сетей / Д. А. Маршалко // Учёные записки Брянского государственного университета. — 2019. — №4(16). — С. 10–13.

6. TomatOD [Электронный ресурс] : база данных / NexusWho. – Режим доступа: https://www.kaggle.com/datasets/nexuswho/tomatod

Разработка и применение семейства алгоритмов K-Means для сегментации изображений рукописного текста с фоном

В. В. Шепляков

Студент бакалавр

В. С. Тарасов

Старший преподаватель

Введение

Алгоритмы кластеризации играют важную роль в анализе данных. Они позволяют группировать объекты на основе их сходства, что упрощает понимание больших объемов информации. Одним из популярных и перспективных алгоритмов кластеризации является модель K-Means.

Сегментация изображений представляет собой область науки, занимающаяся поиском возможности к разделению изображения на сегменты, представляющие определенный интерес: содержащие объект или другую информацию.

Целью данной работы является обзор алгоритма кластеризации K-Means и изучение свойств его распространения/модификации на задачи сегментации в виде добавления пространственной связности, которое призвано сохранить целостность сегментов. Также будет предложен метод начальной инициализации, привязанный к исследуемой предметной области.

Предполагается, что модификация алгоритма и критерий выбора начальной инициализации улучшит предполагаемую визуальную сегментацию и продемонстрирует возможности алгоритма при работе с данными записей рукописного текста в его возможностях отделения непосредственно записей от фона или другой посторонней информации.

Ожидается, что данное исследование предоставит исследованный инструмент, готовый к адаптации к предметной области, а также реализует пример, предоставляющий оптимальные результаты.

1. Особенности классического алгоритма

В обыкновенной реализации при работе с изображениями алгоритм K-Means использует 3 признака, не беря в расчет пространственное положение пикселей:

© Шепляков В. В., Тарасов В. С., 2025

$$R_{x,y}, G_{x,y}, B_{x,y}, x \in 1..n, y \in 1..m \quad (1)$$

где m – высота изображения, n – ширина изображения, компоненты $R_{x,y}$, $G_{x,y}$, $B_{x,y}$ представляют собой значения красной, зеленой и синей составляющих пикселя.

Работу K-Means можно разделить на 2 основных этапа [1]:

– Начальная инициализация центроидов.
– Запуск цикла обучения с повторным прикреплением пикселей и их смещением, а также условием для выхода из него.

Среди этих классических этапов можно выделить 2 основных параметра, которые будут в значительной степени влиять на качество получаемой кластеризации в частом случае работы с изображениями. Это:

1. Начальная инициализация центроидов, по поводу которой существует множество соображений, каким образом корректнее ее производить в той или иной ситуации. Классические примеры:

– Случайная инициализация центроидов, которая не дает стабильного результата, но в результате множества повторов может дать ошибку, сопоставимую с адаптированными под предметную область алгоритмами.
– Инициализация центроидов с помощь K-Means++, в которой каждой новый добавляемый центр выбирается самым далеким от числа уже инициализированных. Для этого алгоритма характерно выделение цветов изображения, которые могут занимать лишь малую часть изображения, что в некоторых случаях является проблемой.

Однако выбор конкретного критерия чаще всего зависит от доменной области. В данной статье будет рассмотрен адаптивный вариант инициализации центроидов, который будет учитывать пространственное позиционирование текстовых записей на изображении. Демонстрация представлена на рис. 1.

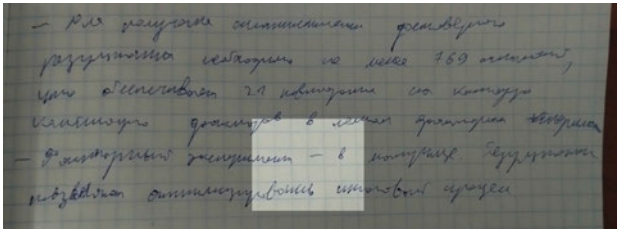


Рис. 1. Окно первого кластера

Выбор центроидов кластеров осуществляется следующим образом:

В качестве первого центроида выбирается пиксель, входящий в окно, который минимизирует ошибку по цвету относительно всего окна.

В качестве всех оставшихся центроидов выбираются пиксели, лежащие на ребрах изображения и делящие их на одинаковые части. При этом на каждое ребро отводится число центроидов, отличающееся не более чем на 1.

Примечательно, что такой подход будет действовать единообразно для разных цветов изображения, так как нет привязки по конкретному цвету, но используется пространственная составляющая.

2. Условие выхода из цикла обучения. Для классической реализации K-Means характерно задание малого значения ε, которое является пороговым для движения центров между итерациями, что обязательно приводит к сходимости алгоритма. Это позволяет путем увеличения числа итераций получать все более корректные результаты кластеризации по внутри- и меж-кластерной метрикам дисперсий, однако в частном случае при работе с изображениями и модернизацией алгоритма это свойство теряется, что приводит лишь к некоторым эмпирическим соображениям по поводу конкретного критерия выхода.

С другой стороны, для достижения поставленной задачи по отделению рукописных записей от фона, будет недостаточно использовать метрики кластерных дисперсий, так как речь идет о пространственной связности.

2. Оценка ошибок

В качестве датасета для всех алгоритмов было представлено 6 среднестатистических бумажных записей студентов ВГУ, переданных на добровольной основе. Пример фотографии представлен на рис 1.

Датасет был подобран таким образом, чтобы любые два изображения не были похожи между собой по структуре цвета или местоположению бумаги, но похожи по смыслу: каждое содержит бумажные записи, которые имеют задний фон.

Разметка производилась в бинарном формате, то есть предполагаемые обрезаемые части изображения помечались белым цветом, а сегменты бумаги черным.

Так как число кластеров при K-Means может отличаться и не всегда делит изображение на черное и белое, то необходимо было сопоставить сегментированное изображение с размеченным.

Таким образом, среди получившейся кластеризации был произведен поиск сегмента, который имеет наибольшее представительство все в том же окне, представленном на рис 1. И

именно этот сегмент и будет принят алгоритмом за “черный” в бинарной маске разметки, а все остальные оставшиеся сегменты за “белый”.

А далее происходит вычисление суммарной ошибки на языке java использующем нативную библиотеку OpenCV [2]. Реализующий код представлен на листинге.

Листинг

Вычисление суммарной ошибки

```
        Mat correct_binary_mask = new Mat();
        Core.compare(calculated_markup, expected_markup,
correct_binary_mask, Core.CMP_EQ);
        double correct =
Core.sumElems(correct_binary_mask).val[0] / 255 /
centers_labels.rows() / centers_labels.cols();
        return 1 - correct;
```

3. K-Means with connectivity constraints

В текущем разделе речь пойдет о той модификации алгоритма, о которой говорилось во введении. Модификация получила свое название из ее основного смысла – сохранение пространственной связности при сегментации [3]. Предполагается, что учет пространственного положения пикселей улучшит визуальную сегментацию и уменьшит ошибку.

Для реализации такого подхода было предложено использовать приоритетную очередь, приоритет элемента в которой определяется цветовым расстоянием пикселя от своего центра, а соседство учитывается путем обхода внутри очереди только по соседним пикселям.

На рис. 2 представлена обобщенная диаграмма деятельности данного алгоритма в текущем варианте реализации.

В данном случае вместо определения центроидов, как в K-Means, которые отражают среднее значение, не привязываясь к выборке, будут использоваться медоиды, каждый из которых представляет собой один из образов выборки, сводящий к минимуму ошибку по цвету от центроида.

Основная идея заключается в том, чтобы при каждом извлечении элемента очереди присваивать ближайший по цвету к медоиду и вплотную прилегающий к уже образованному сегменту пиксель.

Сложность алгоритма составляет:

$$S = O(i * n * \max(\log n, K)) \quad (2)$$

где i – количество итераций по обучению, n – общее количество пикселей, K – количество кластеров, S – сложность алгоритма в худшем случае.

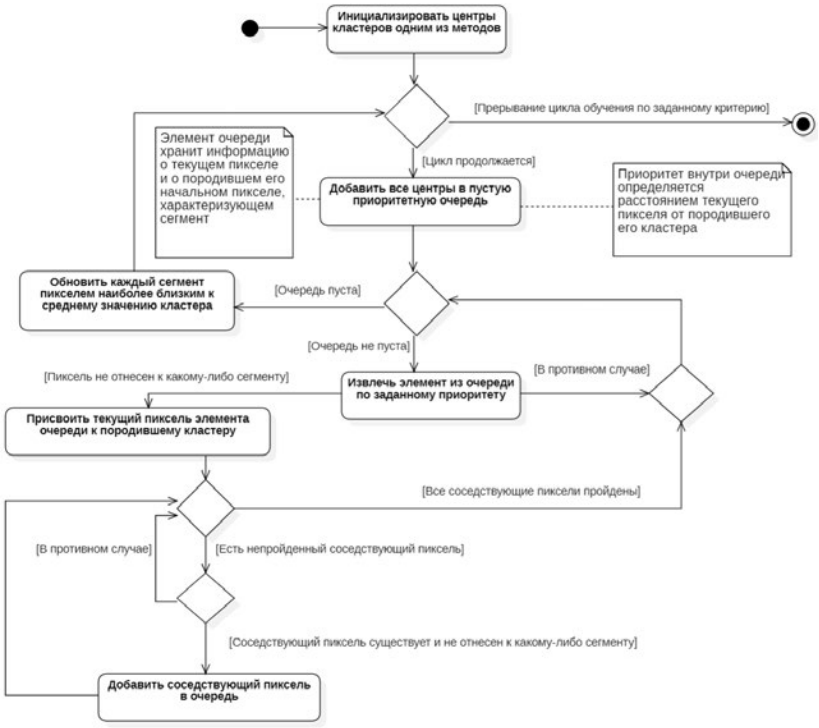
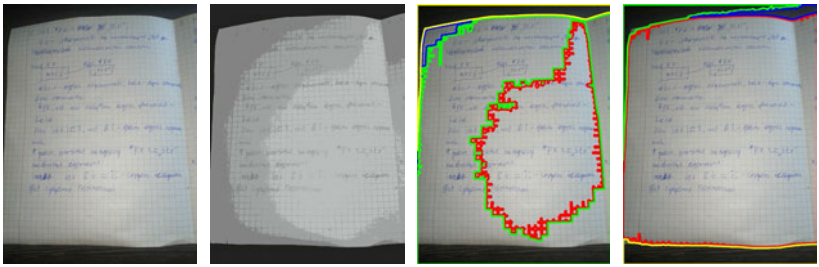


Рис. 2. Диаграмма активности алгоритма

В отличие от K-Means, где с тремя признаками сложность в худшем случае составляет $O(i*n*K)$, в данном алгоритме множитель K заменяется на $\max(\log n, K)$, так как при бóльших значениях K в алгоритме все еще присутствует цикл для вычисления новых положений центров, а в противном случае $\log n$, который отражает сложность вставки или удаления элемента при работе со структурой данных приоритетной очереди в виде бинарной кучи [4].

На рис. 3 представлен пример обработки изображения обычным алгоритмом K-Means и K-Means WCC с двумя возможными вариантами начальной инициализации.



а б в г

а – исходное изображение (размеры: 960x1280 пикселей), б – 4-Means со случайной инициализацией центроидов (цвет отображает кластер), в – 4-Means with connectivity constraints со случайной инициализацией медоидов (на рисунке отображены границы), г – 4-Means with connectivity constraints с адаптивной инициализацией медоидов (на рисунке отображены границы)

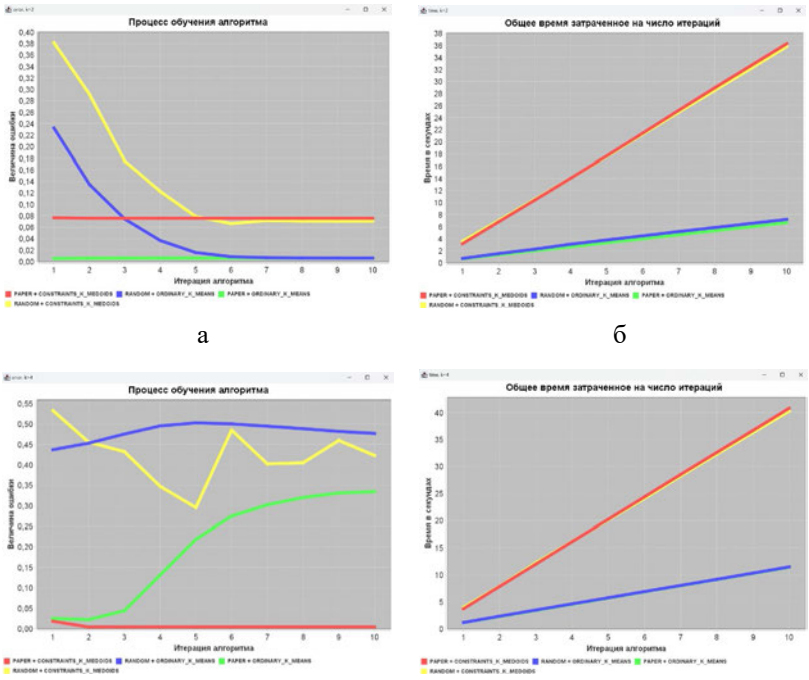
Рис. 3. Результаты сегментации различными алгоритмами K-Means

4. Тестирование алгоритмов

Тестирование алгоритма было произведено на два основных параметра: скорость и точность. Точностью выступает величина ошибки, подсчет которой был представлен ранее, а скорость была измерена в секундах в среде исполнения java: 24-jdk-bullseye [5] внутри docker, который был запущен в wsl2 Ubuntu-24.04. Конфигурация системы: Core I7-13620H + DDR5 2593 MHz 16GB RAM с таймингами CL 42, tRCD 42, tRP 42, tRAS 82.

Wsl машина была ограничена 4 виртуальными ядрами, которые распределялись между 4 физическими энергоэффективными ядрами данного процессора, а ограничение по памяти составляло 4 ГБ, которое осуществлялось на уровне docker и jvm.

На рис. 4 представлена сравнительная характеристика алгоритмов K-Means (ORDINARY_K_MEANS) и K-Means WCC (CONSTRAINTS_K_MEDOIDS) со случайной инициализацией центроидов (RANDOM) и адаптивным алгоритмом (PAPER) на примере изображения, представленного на рис. 3а.



а б

в г

а – график точности алгоритмов при k=2,
б – график времени работы при k=2,
в – график точности алгоритмов при k=4,
г – график времени работы при k=4

Рис. 4. Графики оценок производительности и точности алгоритмов K-Means в процессе обучения при работе с изображением, представленном на рис. 3а

Результаты оказались неоднозначными в рамках данной фотографии. Исходя из рис. 4в, можно заметить, что PAPER + K-Means WCC с количеством кластеров = 4 показали низкую ошибку, которая, к тому же, в отличие от K-Means, соответствует критериям визуальной сегментации. Полученная в этом случае сегментация представлена на рис. 3г.

Заключение

Таким образом, в ходе работы был представлен алгоритм K-Means WCC и его адаптация в рамках предметной области, которые в

некоторых случаях могут показывать результаты, приближенные к визуальной сегментации.

Величина ошибки пока слишком велика в общем случае, чтобы говорить о сколько-нибудь корректной сегментации, однако в перспективе возможно использование метрик кластеризации (например, силуэта), которые позволили бы определять оптимальное число кластеров для каждого конкретного изображения при работе со случайными бумажными данными.

Итоговая версия алгоритма оказалась достаточно гибкой и, хотя её адаптация требует немалых усилий, она позволяет достигать требуемой точности при решении практических задач сегментации.

Список литературы

1. Простейшая кластеризация изображени методом к-средних (k-means) [Электронный ресурс] : Статья. – Режим доступа: https://habr.com/ru/articles/165087/

2. OpenPnP OpenCV 4.9.0-0 API [Электронный ресурс] : Документация. – Режим доступа: https://javadoc.io/doc/org.openpnp/opencv/4.9.0-0/index.html

3. Extending K-Means to Preserve Spatial Connectivity / S. Soor [et al.] // IEEE International Geoscience and Remote Sensing Symposium (IGARSS) (Valencia, 22-27 July 2018). – Valencia, Spain, 2018. – P. 6959-6962.

4. Queue [Электронный ресурс] : Курс лекций Java. – Режим доступа: https://javarush.com/quests/lectures/jru.module1.lecture23

5. What is OpenJDK? [Электронный ресурс] : Репозиторий dockerhub. – Режим доступа: https://hub.docker.com/_/openjdk

Разработка приложения для размещения рекламы на радиостанции. Жизненный цикл

Д. А. Шестопалов

Студент бакалавр

А. И. Чекмарев

Старший преподаватель

Введение

В наше время радио остаётся эффективным средством массовой информации, способным оперативно вещать информацию на широкую аудиторию, несмотря на появление новых медиаплатформ. Радиостанции стремятся максимально автоматизировать все бизнес-процессы, включая размещение рекламных интеграций, чтобы повысить точность планирования эфирного времени и оперативно реагировать на запросы рекламодателей.

Одним из ключевых компонентов такой автоматизации является REST-API, обеспечивающий загрузку, проверку и управление рекламными аудиофайлами на стороне сервера. В рамках данной работы рассматривается разработка серверной части системы на базе Spring Boot [1], отвечающей за полный жизненный цикл рекламного аудиофайла: от приёма и валидации файла, расчёта стоимости размещения и сохранения его метаданных в базе до модерации статуса и интеграции в расписание вещания.

1. Реализация загрузки файла в формате MP3

Загрузка рекламного аудиофайла в формате MP3 в системе организована через контроллер [1] AudioRecordingController. Клиент делает запрос, как на рис. 1. На стороне сервера метод createRecording получает объект MultipartFile, хранящий всю информацию о загружаемом файле. Метод выполняет проверку на соответствие типа файла, как аудиофайла в формате MP3, а затем приступает к предварительному анализу.

Прежде чем сохранять файл на диск и записывать информацию в базу, система вычисляет длительность аудиозаписи. Для MP3-файла это можно сделать по соотношению между размером файла и его битрейтом [2]:

© Шестопалов Д. А., Чекмарев А. И., 2025

$$D = \frac{8 \cdot (endOffset - startOffset)}{bitrate} + 0,5 \quad (1)$$

где *bitrate* – битрейт в битах в секунду, *endOffset* и *startOffset* – смещения в байтах.

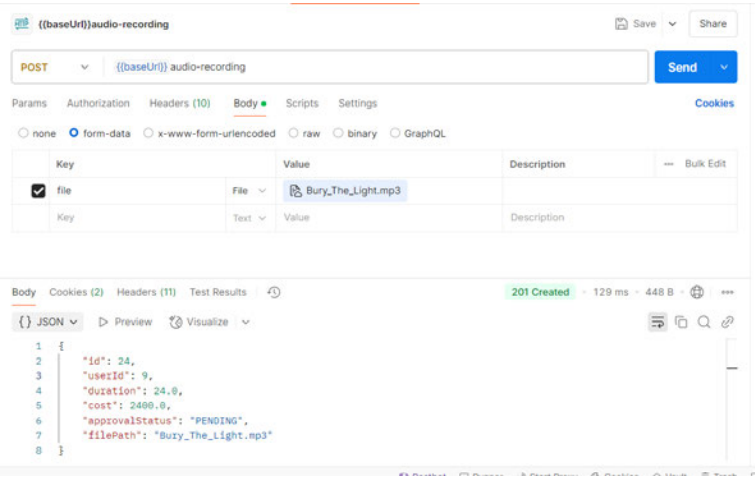


Рис. 1. Пример запроса для загрузки файла через Postman

Затем полученная длительность файла используется для вычисления цены аудиофайла в методе из сервиса AudioRecordingService getCostByDuration(Double duration). В нем идет расчёт по формуле:

$$C = D \cdot Fp \quad (2)$$

где D – длительность файла в секундах, Fp – стоимость одной секунды в рублях.

После всех вычислений данные об ID владельца файла, именовании, стоимости и длительности записываются в экземпляр класса AudioRecordingDto в соответствующие поля. Также присваивается статус PENDING, означающий, что файл не был проверен администрацией приложения.

Когда все данные записаны в AudioRecordingDto, он передаётся в метод createRecording(AudioRecordingDto recordingDto) для сохранения всей информации в базе данных проекта.

Физическое же сохранение файла происходит в специальной директории проекта при помощи метода saveAudio(MultipartFile file, Long fileId), в котором помимо обычного сохранения происходит присваивание ему уникального имени для поддержки уникальности каждого нового файла. Оно, в свою очередь, состоит из оригинального наименования файла и присвоенного ему базой данных ID, как на рис. 2.

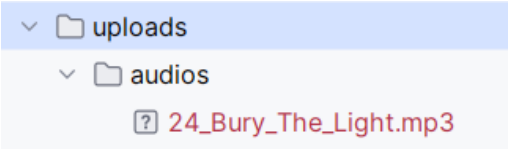


Рис. 2. Пример сохраненного файла

2. Реализация проверки файла администратором

Уже сохраненный файл имеет по умолчанию статус PENDING, то есть требующий проверки. В роли проверяющего выступает администратор приложения, который имеет доступ ко всем аудиозаписям. Для получения всех записей, требующих проверки, используется метод getRecordingsByStatus(@PathVariable String status) из контроллера AudioRecordingController, который на стороне клиента выглядит, как запрос на рис. 3.

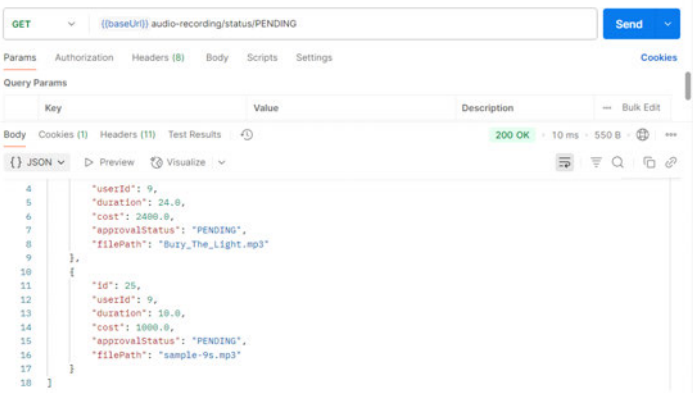


Рис. 3. Пример получения всех файлов со статусом PENDING

После получения полного списка файлов администратор выбирает один из файлов для прослушивания, а затем присваивает ему один из двух статусов: APPROVED (файл успешно прошел проверку) или REJECTED (файл запрещен для прикрепления в пробелы вещания). И

вызывается метод updateRecordingStatus(@PathVariable Long id, @RequestParam String status) для обновления статуса определенного файла в базе данных через запрос на рис. 4.

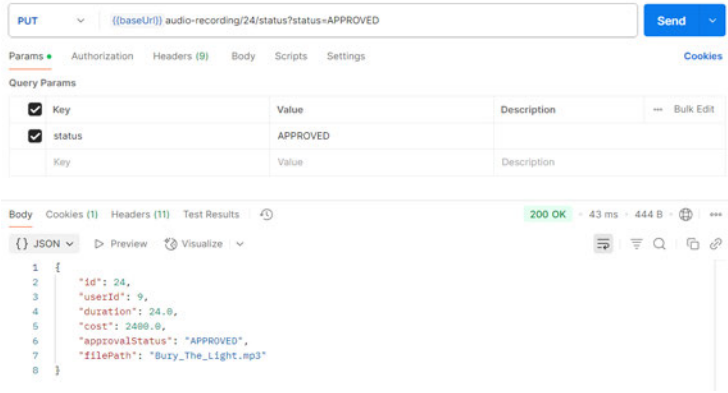


Рис. 4. Пример запроса по изменению статуса аудиозаписи

3. Реализация прикрепления аудиофайла к пробелам в вещании

Поскольку вручную прикреплять один и тот же аудиофайл к нескольким слотам вещания – задача избыточная, этот процесс был автоматизирован.

Со стороны клиентской части поступает информация от рекламодателя: сколько средств он готов потратить и выбрал ли он для своего файла повышенный приоритет. Повышенный приоритет означает, что рекламодатель хочет, чтобы его аудио объявление воспроизводилось в «прайм-тайм» – с 07:00 до 09:00, с 12:00 до 14:00 и с 17:00 до 19:00.

Далее, используя информацию о стоимости файла, мы вычисляем количество вхождений для него:

$$N = \frac{B}{C}, \tag{3}$$

где B – денежная сумма, предоставленная пользователем, C – цена прикрепляемого файла в рублях.

Затем из контроллера BroadcastSlotController вызывается метод getAvailableSlotsByRadioStationIdWithPriority(@PathVariable Long radioStationId, @RequestParam boolean highPriority), который возвращает список всех доступных пробелов вещания для определённой радиостанции по выбранному приоритету. Пример запроса на рис. 5.

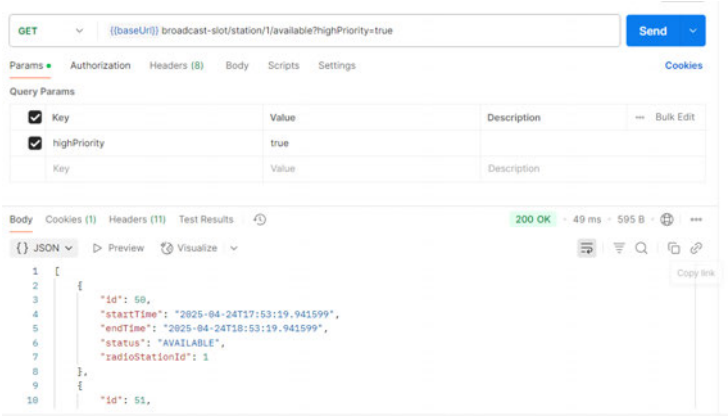


Рис. 5. Пример запроса списка доступных пробелов в вещании для радиостанции с ID 1 и высоким приоритетом

Из данного списка осуществляется случайная выборка *N* пробелов, если их меньше, то часть средств возвращается пользователю. В эти пробелы при помощи запроса к методу createPlacement(@RequestBody PlacementDto placementDto) из PlacementController осуществляется прикрепление файла, где PlacementDto хранит информацию о ID аудиофайла и пробела. Запрос на рис. 6.

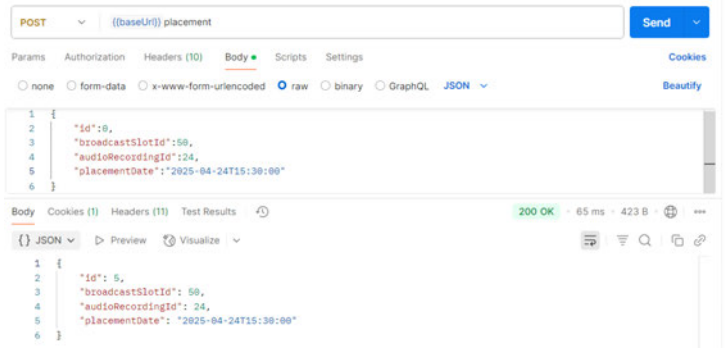


Рис. 6. Пример запроса на прикрепление файла к пробелу вещания

Также в процессе прикрепления аудиофайла вызывается метод splitBroadcastSlot(@RequestParam Long slotId, @RequestParam Long audioId), который выполняет разбиение исходного слота по переданному

идентификатору. Новый слот создаётся на основе времени, оставшегося после вычитания длительности аудиофайла. Пример запроса на рис. 7.

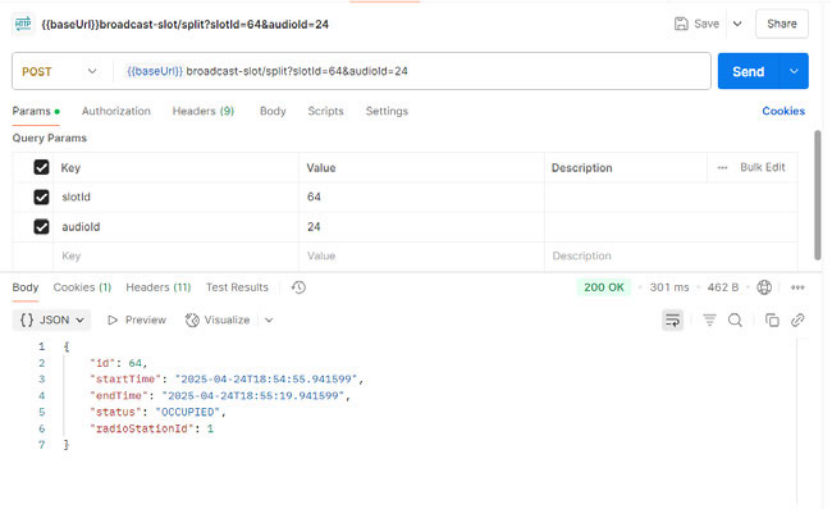


Рис. 7. Пример запроса на разбиение пробела в вещании

Заключение

В рамках работы была полностью реализована серверная часть приложения на базе Spring Boot, обеспечивающая полный жизненный цикл рекламного аудиофайла: приём и валидацию MP3-загрузок, вычисление длительности и стоимости размещения, сохранение метаданных в PostgreSQL [3] через Spring Data JPA [4] и Flyway [5], административную модерацию статусов и динамическое прикрепление в вещательные слоты с учётом приоритетов.

Разработанные модули доказали свою надёжность и масштабируемость:

– Загрузка и предварительный анализ аудио обеспечивают корректное вычисление длительности ролика и стоимости его размещения для точного планирования прикрепления к пробелам в вещании.

– Проверка файлов администратором гарантирует безопасность и контроль качества содержимого аудиофайла.

– Автоматическое прикрепление в доступные пробелы и их разбиение по длительности ролика повышают эффективность использования эфирного времени.

Таким образом, разработанный API демонстрирует гибкость и надёжность архитектуры, позволяя радиостанциям автоматизировать размещение рекламы и повысить доходность эфирного времени.

Список литературы

1. Pivotal Software, Inc. Spring Boot Reference Guide [Электронный ресурс] : руководство пользователя / Pivotal Software, Inc. – Режим доступа: https://docs.spring.io/spring-boot/index.html

2. Javadoc.io. com.googlecode.soundlibs:mp3spi [Электронный ресурс] : API-документация / javadoc.io – Режим доступа: https://javadoc.io/doc/com.googlecode.soundlibs/mp3spi/latest/index.html

3. PostgreSQL Global Development Group. PostgreSQL Documentation [Электронный ресурс] / PostgreSQL – Режим доступа: https://www.postgresql.org/docs/current/

4. Spring Data. Spring Data JPA [Электронный ресурс] / Spring Data – Режим доступа: https://docs.spring.io/spring-data/jpa/reference/index.html

5. Redgate Flyway Documentation [Электронный ресурс] / Redgate – Режим доступа: https://flywaydb.org/documentation/

Анализ и применение структуры данных "Дерево палиндромов" в алгоритмах обработки строк

Д. И. Шульженко

Студент бакалавр

Д. И. Соломатин

Старший преподаватель

Введение

В сфере алгоритмов обработки строк палиндромы - подстроки, обладающие свойством одинакового чтения в прямом и обратном направлениях, - занимают важное положение благодаря их симметричным характеристикам. Обнаружение и анализ таких подстрок имеют существенное значение для решения задач в биоинформатике (например, выявление симметричных последовательностей в ДНК), сжатия данных, криптоанализа, текстового поиска и других дисциплин [1-2]. Также очень много задач на строковые алгоритмы, встречающихся в олимпиадах по спортивному программированию, связаны именно с палиндромами.

Классическим подходом к выявлению палиндромов является алгоритм Манакера, разработанный в 1975 году [3]. Он позволяет определить все палиндромные подстроки, центрированные на каждом символе исходной строки, за линейное время. Однако данный алгоритм не сохраняет полную структуру палиндромов и обладает ограниченной масштабируемостью при решении задач с динамическим вводом данных и множественными запросами.

Тогда в 2015 году Михаил Рубинчик и Арсений Шур представили эффективную структуру данных - дерево палиндромов (англ. Eertree), предназначенное для обработки строк в режиме реального времени и хранения всех уникальных палиндромных подстрок [1]. Предложенная структура обеспечивает высокую скорость построения, компактное хранение данных и выполнение разнообразных запросов, таких как поиск палиндрома максимальной длины, подсчет палиндромов, их факторизация и сравнение строк.

© Шульженко Д.И., Соломатин Д.И., 2035

В данной работе рассматриваются устройство, алгоритмы и практическое применение дерева палиндромов, а также его сравнение с алгоритмом Манакера и другими.

1. Структура дерева палиндромов

Дерево палиндромов (EerTree) представляет собой направленный ациклический граф, вершины которого соответствуют уникальным палиндромным подстрокам. Данная структура оптимизирована для обработки данных в режиме реального времени, что предполагает последовательное добавление символов к строке. Рассмотрим подробнее каждый его элемент.

В структуре используются два специальных корня: $root_{even}$ и $root_{odd}$. И поэтому в реальности структура представляет собой не одно, а два дерева, одно нужно для палиндромов четной длины, а второе - для нечетной. Корни этих деревьев соответственно и будут называться $root_{even}$ и $root_{odd}$. Для верной работы структуры принято, что $root_{odd}$ соответствует палиндрому длины -1, а $root_{even}$ соответствует фиктивному палиндрому длины 0.

Поскольку структура представляет собой дерево, начнем анализ с её вершин. Каждая вершина соответствует уникальному палиндрому, который определяется двумя параметрами: длиной палиндрома и позицией его окончания в исходной строке. Кроме того, в вершине содержатся ссылки для переходов по ребрам, где ребро, обозначенное символом x, соединяет вершину u с вершиной v только в том случае, если строка v’ может быть выражена как xu’x, где u’ - строка, соответствующая вершине u. В вершине может храниться дополнительная информация, специфичная для решения конкретной задачи.

Помимо вершин и рёбер в дереве палиндромов также присутствуют *суффиксные ссылки*. На рисунке ниже суффиксные ссылки обозначены прерывистыми линиями.

Для каждой вершины u её суффиксная ссылка ведет в такую вершину w, что w′ является наибольшим суффиксом строки u’ относительно других вершин [1]. Заметим, что суффиксная ссылка из вершины одного дерева может вести как в то же, так и в другое дерево.

На рисунке для примера представлено полностью сформированное дерево палиндромов для строки "aabcba". Таким образом, дерево палиндромов гарантирует компактное представление всех уникальных палиндромных подстрок.

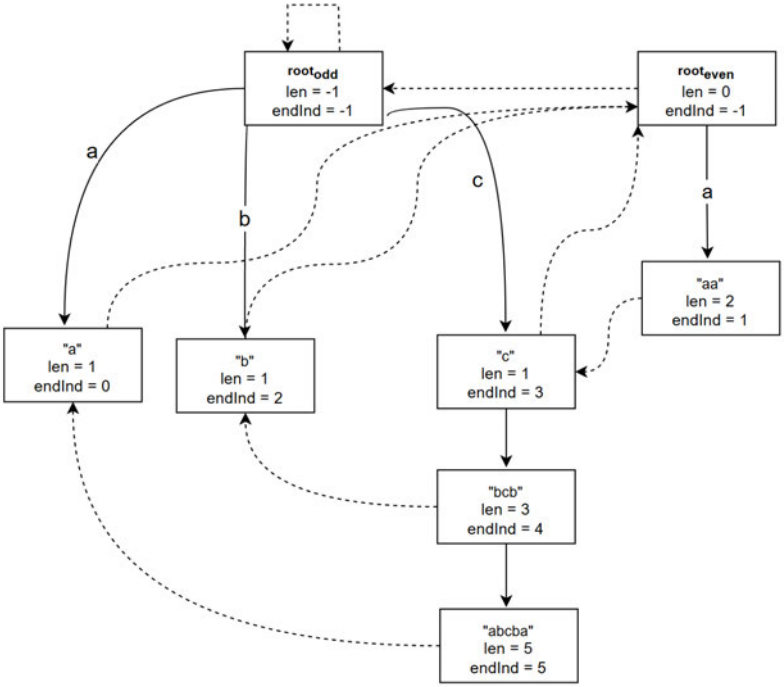


Рисунок. Дерево палиндромов для строки "aabcba"

2. Принцип работы дерева палиндромов

Построение дерева происходит по мере поступления каждого символа. Структура поддерживает указатель на последний добавленный палиндром. Алгоритм работает по следующему принципу [1, 4]:

1. Переход по суффиксным ссылкам до тех пор, пока не найдется палиндром P, такой что xPx - тоже палиндром.
2. Если такой палиндром уже существует - происходит переход по существующему ребру.
3. Иначе - создается новая вершина.
4. Суффиксная ссылка новой вершины устанавливается на xSx, где S - максимальный палиндромный суффикс P.
5. Увеличиваются счетчики посещения, если они необходимы для решения задачи.

Каждое добавление символа занимает амортизированное время O(logσ), где σ - размер алфавита. Общее время построения дерева длины n составляет O(n*logσ), а память - O(n) [1].

3. Применение дерева палиндромов для решения задач

Рассмотрим несколько примеров применения данной структуры для решения задач и сравним их с результатами, получаемыми с использованием алгоритма Манакера, если это применимо.

В первой задаче нам *необходимо определить число новых подпалиндромов, порождаемых после добавления символа x в конец строки s.* Учитывая, что добавление одного символа может привести к появлению не более одного нового палиндрома, мы будем строить дерево палиндромов посимвольно и для каждого добавленного символа определять, появился ли новый палиндром, подсчитывая общее число вершин в структуре.

В другой задаче, где *требуется определить число подпалиндромов, которые содержатся в переданной строке,* мы можем воспользоваться как алгоритмом Манакера, так и деревом палиндромов, они будут работать за одинаковую асимптотику. Для этого в структуре дерева палиндромов необходимо дополнительно рассчитывать длину цепочки суффиксных ссылок от каждой вершины до корня (включая саму вершину). Затем это значение будет прибавляться к общему результату для каждого нового символа t по мере их добавления.

4. Сравнение с алгоритмом Манакера

Алгоритм Манакера находит самый длинный палиндром за O(n) [3]. Он строит массив радиусов палиндромов вокруг каждого центра и эффективен для задачи нахождения наибольшей палиндромной подстроки. Однако он не возвращает структуру или явный список всех различных палиндромов.

Основной недостаток алгоритма Манакера заключается в его неспособности к расширению для решения более широкого круга задач, в отличие от дерева палиндромов [1, 5]. Дерево палиндромов хранит множество палиндромов в явном виде и позволяет быстро отвечать на запросы о числе и количестве их вхождений.

В таблице отображена краткая информация для сравнения данных двух алгоритмов. Основными характеристиками при сравнении являются время работы, а также занимаемая память при работе алгоритма.

Таблица

Сравнение алгоритма Манакера с деревом палиндромов

Характеристика	Алгоритм Манакера	Дерево палиндромов
Время работы	O(n)	O(n)
Занимаемая память	O(n)	O(n)

Окончание таблицы

Хранение всех палиндромов	Нет	Да
Потоковая обработка	Нет	Да
Простота реализации	Средняя	Средняя

По временной и пространственной эффективности EerTree сопоставим с суффиксными деревьями и автоматами [2, 4]. Однако благодаря своей специализации на палиндромах он зачастую имеет лучшие константы и проще в реализации. В отличие от комбинированных решений на базе Манакера и Укконена, EerTree не требует построения сложных вспомогательных структур и остаётся линейным по времени и памяти [1].

Таким образом, дерево палиндромов предпочтительнее для задач, требующих всех палиндромов, но избыточно для поиска только самого длинного.

5. Оптимизации и расширения дерева палиндромов

Несмотря на линейную асимптотическую сложность и простоту базовой реализации, в ряде практических задач возникают требования, превышающие возможности исходного алгоритма. Исследования современных модификаций и оптимизаций структуры направлены на повышение её универсальности, снижение вычислительных затрат и адаптацию к новым вычислительным платформам [4, 5]. В рамках анализа будут рассмотрены четыре модификации: работа с динамическими строками, адаптация дерева палиндромов для скользящего окна, создание двустороннего дерева палиндромов и параллельная обработка символов.

Изначальный вариант дерева палиндромов ориентирован на монотонное добавление символов в конец строки. Однако во многих задачах необходимо поддерживать удаление символов - например, при скользящем окне в потоках данных или при откате операций во взаимодействии с пользователем [1, 5]. Для таких целей реализуются модификации с поддержкой отката состояний, аналогично техникам rollback в суффиксных автоматах. Каждое добавление символа сопровождается сохранением текущего состояния дерева, включая активную вершину, что позволяет восстановить его при необходимости.

Современные системы требуют масштабируемых алгоритмов для многоядерных архитектур. Хотя Eertree работает последовательно, возможно распараллеливание через сегментацию строки и построение частичных деревьев с последующим слиянием [4]. Это требует

синхронизации для обработки границ, но эффективно при правильной реализации. Эксперименты, включая CUDA, показывают перспективы для анализа больших текстов или биологических данных.

Одним из значимых расширений является *двустороннее дерево палиндромов*, представленное в 2022 году [5]. Эта модификация позволяет выполнять операции с двусторонней очередью за линейное время, что особенно полезно для диапазонных запросов, таких как подсчет различных палиндромных подстрок или поиск самого длинного палиндрома в заданной подстроке. Такое расширение значительно повышает гибкость структуры в задачах с динамическими запросами.

Еще одним важным дополнением является адаптация EerTree для работы со *скользящим окном* [6]. Эта модификация позволяет поддерживать дерево палиндромов для окна фиксированной длины, что делает его применимым для потоковой обработки данных. Существуют алгоритмы, обеспечивающие временную сложность O(d*σ'), где d - длина окна, а σ' - максимальное число различных символов в окне, что делает эту оптимизацию эффективной для реальных приложений.

Эти улучшения и расширения значительно повышают функциональность дерева палиндромов, превращая его в универсальное средство для обработки строк в динамических и параллельных сценариях. Благодаря таким доработкам EerTree становится конкурентоспособным инструментом для решения сложных задач в биоинформатике, анализе текстов и других областях.

Заключение

Дерево палиндромов (EerTree) - значимая структура данных в алгоритмической обработке строк. Оно компактно хранит все палиндромные подстроки, поддерживает их динамическое обновление в реальном времени и превосходит классические методы, такие как алгоритм Манакера, по гибкости.

EerTree работает в режиме реального времени с линейной пространственной сложностью, решая сложные задачи, включая подсчет уникальных палиндромов и k-факторизацию. Это делает его востребованным в биоинформатике, криптоанализе и спортивном программировании. Современные модификации, такие как двусторонние деревья и адаптация для скользящих окон, усиливают его применимость к большим данным и параллельным вычислениям. Реализация EerTree требует небольших усилий, но его эффективность и универсальность оправдывают затраты, делая его сильным инструментом для решения актуальных задач в различных областях.

Список литературы

1. Rubinchik, M. M. EERTREE: An Efficient Data Structure for Processing Palindromes in Strings / M. M. Rubinchik, A. M. Shur // Proc. 26th Int. Workshop on Combinatorial Algorithms (IWOCA 2015). – Cham : Springer, 2016. – Vol. 9538 (Lecture Notes in Computer Science). – P. 321–333. – Режим доступа: https://ar5iv.org/html/1506.04862v2

2. Groult, R. Counting distinct palindromes in a word in linear time / R. Groult, E. Prieur, G. Richomme // Information Processing Letters. – 2010. – Vol. 110. – Iss. 20. – P. 908-912

3. Manacher, G. A new linear-time on-line algorithm finding the smallest initial palindrome of a string / G. Manacher // Journal of the ACM. – 1975. – Vol. 22. – Iss. 3. – P. 346-351.

4. Kosolobov, D. Finding distinct subpalindromes online / D. Kosolobov, M. Rubinchik, A. M. Shur // Prague Stringology Conference 2013 : Proceedings. – Prague : Czech Technical University, 2013. – P. 63–69. – Режим доступа: https://arxiv.org/abs/1305.2540

5. Wang, Y. Double-Ended Palindromic Trees: A Linear-Time Data Structure and Its Applications / Y. Wang, M. Yang, D. Zhu // arXiv preprint. – 2022. – Режим доступа: https://ar5iv.org/html/2210.02292

6. Palindromic Trees for a Sliding Window and Its Applications / T. Mieno [et al.] // Information Processing Letters. – 2022. – Vol. 173. – Article 106174. – DOI: 10.1016/j.ipl.2021.106174

Использование квантовых алгоритмов для решения задач дискретной оптимизации

Е. О. Юров

Студент бакалавр

В. В. Коротков

Ст. преподаватель

Введение

Дискретная оптимизация представляет собой фундаментальную задачу, востребованную в самых разных областях – от планирования маршрутов в логистике и распределения ресурсов в проектировании до формирования инвестиционных портфелей в финансах и анализа сложных социальных или коммуникационных графов. Суть таких задач заключается в выборе наилучшего решения среди конечного, но невероятно большого множества возможных комбинаций. При увеличении размера исходных данных – например, числа узлов в графе или объема параметров – традиционные алгоритмы сталкиваются с экспоненциальным ростом вычислительных затрат, что делает многие практические задачи NP-трудными и зачастую неразрешимыми в разумное время классическими методами.

Квантовые вычисления нацелены на преодоление этих ограничений за счет использования явлений суперпозиции и квантового параллелизма. В этой работе основное внимание уделяется квантовому алгоритму приближённой оптимизации (QAOA) для решения комбинаторной задачи максимального разреза. Мы ставим перед собой задачу сравнить QAOA с классическими эвристиками – генетическим алгоритмом и поиском с запретами – чтобы оценить практическую состоятельность квантовых подходов.

Поскольку современные квантовые устройства относятся к классу шумных промежуточных квантовых систем (NISQ) и ограничены как по числу кубитов, так и по допустимой глубине схем, эксперименты проводятся преимущественно в среде симуляции, однако с перспективой запуска на реальном квантовом оборудовании. Это позволяет выяснить теоретические преимущества квантовых алгоритмов.

© Юров Е. О, Коротков В. В., 2025

1. Квантовые вычисления и квантовые платформы

Для реализации квантовых вычислений используются различные технологии:

Сверхпроводящие кубиты: основаны на эффектах Джозефсона, где сверхпроводящая цепь с двухэлектронным туннелированием образует два взаимодействующих состояния. Такие устройства работают при сверхнизких температурах и обладают относительной простотой интеграции и масштабирования, но требуют снижения уровня шума и повышения когерентности.

Ионные ловушки: отдельные атомы (ионы) удерживаются в электромагнитном поле и манипулируются лазерными импульсами. Ионные кубиты демонстрируют высокую когерентность и точность управления, однако масштабирование на большое число кубитов осложнено требованиями к сложной вакуумной и оптической системе.

Квантовый отжиг: системы используют процессы медленного охлаждения для решения задач оптимизации, постепенно вводя и снимая взаимодействия между кубитами. Хотя они не обеспечивают универсальный набор квантовых гейтов, они эффективны для комбинаторных задач и масштабируются до тысяч условных «кубитов».

Такие технологические подходы определяют основные ограничения устройств: число кубитов, время когерентности, глубину реализуемых схем и уровень шумов. На основе этих платформ строятся облачные сервисы для разработки и тестирования квантовых алгоритмов.

Из табл. 1, видно, что наиболее доступной и мощной платформой для проведения квантовых вычислений является IBM Quantum Experience. На была выбрана как основная платформа для тестирования и в будущем под возможность масштабирования для проведения тестов на реальных квантовых машинах.

Таблица 1

Современные доступные коммерческие платформы для квантовых исследований

Название платформы	Технология квантового процессора	Кол-во кубитов	Программный интерфейс	Доступ
IBM Quantum Experience	Сверхпроводящие кубиты	127 – 156	Qiskit	Открытый
IonQ	Ионные ловушки	11 – 36	Qiskit, Cirq	Открытый
D-Wave Systems	Квантовый отжиг	>1000	Ocean SDK	Закрытый

2. Задача о максимальном разрезе

Задача максимального разреза – NP-трудная задача на взвешенном неориентированном графе: нужно разбить вершины на два множества так, чтобы максимизировать сумму весов рёбер между ними. Применяется в кластеризации сетей, финансовом анализе зависимостей активов и выявлении сообществ в социальных графах. Решение кодируется битовой строкой, где 0 – это первая группа, а 1 – вторая группа вершин, что удобно для классических эвристик и прямой интеграции в квантовые схемы (каждый бит ≈ кубит). Кроме того, благодаря чёткому графовому представлению и возможности генерировать случайные тестовые экземпляры небольших размеров, задача максимального разреза служит идеальным тестом для сравнения производительности и масштабируемости классических и квантовых алгоритмов на реальных и симуляционных квантовых системах.

3. Методы решения задачи максимального разреза: классические и квантовые алгоритмы

Для сравнения эффективности были реализованы классические эвристики и гибридный квантово-классический алгоритм.

Реализован классический генетический алгоритм, который на каждом шаге проводит селекцию по рулетке (выбор пропорционально целевой функции), одноточечный кроссовер и мутацию (инверсия случайного бита). Популяция фиксированного размера обновляется на протяжении всех поколений, а изначальная генерация случайна.

Поиск с запретами начинает с произвольного решения и на каждой итерации переходит к лучшему соседу (инверсия одной вершины), избегая повторений с помощью фиксированного табу-списка недавних операций. При отсутствии улучшений в течение заданного числа шагов алгоритм останавливается.

Оба метода демонстрируют высокую стабильность и масштабируемость.

Квантовый алгоритм приближенной оптимизации (QAOA) – частный случай вариационного квантового алгоритма. Он сочетает многократный запуск параметризованной квантовой схемы с классическим оптимизатором.

Квантовая часть формирует цепочку гейтов, зависящую от вектора параметров θ. Каждый параметр управляет, например, углом поворота или фазовым сдвигом в гейтах. После начинается процесс применения оператора стоимости и смешивания, поочередно в зависимости от кол-ва слоев.

Оператор стоимости U_c кодирует гамильтониан задачи максимального разреза: в модели Изинга вес рёбер переводится в взаимодействия между кубитами.

$$U_c = \sum_{i,j} Q_{ij} Z_i Z_j \quad (1)$$

Оператор смешивания U_m внедряет суперпозиции, позволяя переходить между конфигурациями.

$$U_m = \sum_{i,j} b_i Z_i \quad (2)$$

После каждого запуска схемы производится многократное измерение кубитов, на основании частоты результатов вычисляется значение целевой функции. Классический оптимизатор (SPSA, COBYLA, Nelder-Mead) получает и обновляет параметры, минимизируя энергию, то есть максимизируя разрез.

Увеличение числа слоёв p повышает потенциальную точность, но критично увеличивает шум и время работы. На NISQ-устройствах глубина ограничена свойствами аппаратуры.

Таким образом, QAOA представляет собой гибридную оптимизацию: квантовая часть генерирует решения и смешивает эти генерации, а классическая – эффективно направляет поиск к оптимальному разрезу.

4. Подбор параметров и тестирование на задаче максимального разреза

Для сравнения эффективности классических и квантовых алгоритмов в задаче максимального разреза был проведён параметрический эксперимент на графах с 5 и 10 вершинами. Все графы генерировались по единой стратегии для обеспечения сопоставимых условий.

Для настройки алгоритмов использовалась библиотека Optuna: целевая функция вычисляла среднее значение максимального разреза на 50 графах одинаковой генерации, а оптимизация проводилась по 30 пробам. Такой подход позволял точно подбирать параметры под каждую размерность графа. В табл. 2 предоставлены результаты подбора параметров, на которых производили финальные замеры для каждого алгоритма на представленных размерностях.

Таблица 2

Макропараметры тестируемых алгоритмов

Размер графа	Алгоритм	Параметры
5	Генетический алгоритм	population=90, mutation_rate=0.1235, generations=223
	QAOA	optimizer=NELDER_MEAD, reps=2, maxiter=107
	Поиск с запретами	max_iter=348, tabu_size=38
10	Генетический алгоритм	population=178, mutation_rate=0.2255, generations=178
	QAOA	optimizer=SPSA, reps=1, maxiter=235
	Поиск с запретами	max_iter=271, tabu_size=28

Далее были произведены замеры решений классических и квантовых алгоритмов с представленными макропараметрами. Результаты продемонстрированы на рисунках 1 и 2. По этим данным можно сделать вывод, что на малых размерах данных (5 вершин) квантовый алгоритм показывает себя на уровне классических алгоритмов, по качеству решения задачи. Если размер задачи увеличивается (10 вершин), то квантовый алгоритм проигрывает в качестве решения классическим алгоритмам.

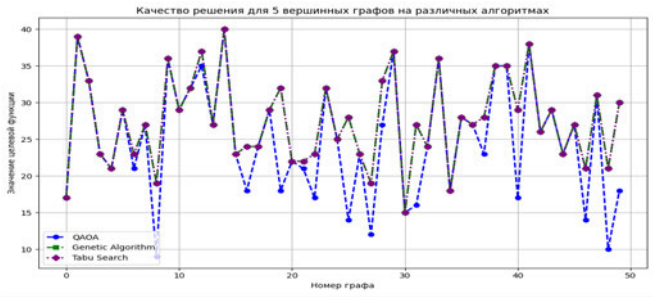


Рис. 1. Качество решения для 5-ти вершинных графов

По итогу тестирования квантовых алгоритмов на симуляторах, можно сказать что они проигрывают классическим алгоритмам по многим параметрам, но стоит учесть, что бралось минимальное кол-во слоев для ускорения вычислений. На данный момент, исходя из тестов на реальных квантовых устройствах, можно сказать, что задачи

дискретной оптимизации более масштабируемы, но менее стабильны при использовании квантовых алгоритмов.

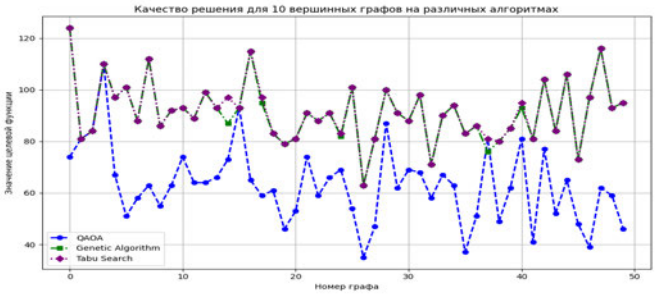


Рис. 2. Качество решения для 10-ти вершинных графов

Заключение

В рамках работы были исследованы как классические, так и квантовые методы решения задачи максимального разреза – типичной задачи дискретной комбинаторной оптимизации. Основное внимание уделялось сравнению эффективности алгоритмов на малых размерностях графов (5 и 10 вершин), с учётом качества решений, масштабируемости и чувствительности к параметрам.

Квантовые алгоритмы, в частности Алгоритм приближённой квантовой оптимизации, находятся на этапе активного изучения. Тесты на симуляторах показали, что они способны находить приемлемые приближённые решения на небольших размерах задачи. Однако на практике такие методы сильно ограничены: ограниченное число кубитов, высокая чувствительность к шуму и потребность в точной настройке параметров затрудняют их эффективное применение. Масштабируемость остаётся слабым местом, особенно при переходе от 5 к 10 вершинам.

С другой стороны, классические эвристические алгоритмы – генетический алгоритм и поиск с запретами – продемонстрировали высокую стабильность, предсказуемость и способность находить оптимумы или близкие к ним решения. Эти методы хорошо адаптируются под структуру графа, не требуют специализированного оборудования и масштабируются значительно лучше в рамках текущих задач.

Таким образом, на данном этапе классические методы остаются наиболее надёжным инструментом для практического решения задач оптимизации. Квантовые алгоритмы, несмотря на свои теоретические

перспективы, пока выступают скорее как исследовательская альтернатива. Однако с развитием квантового оборудования и улучшением точности симуляторов можно ожидать, что квантовые подходы займут значимое место в будущем арсенале оптимизационных средств.

Список литературы

1. An investigation of IBM quantum computing device performance on combinatorial optimisation problems / M. T. Khumalo [и др.] // Neural Computing and Applications. – 2025. – № 37. – P. 611-626.
2. Perez-Ramirez, D. F. Variational Quantum Algorithms for Combinatorial Optimization [Электронный ресурс] / D. F. Perez-Ramirez. – arXiv preprint, 2024. – 10 с.
3. Benchmarking the quantum approximate optimization algorithm / M. Willsch, D. Willsch, F. Jin [и др.] // Quantum Information Processing. – 2020. – Т. 19. – Ст. 197.
4. Multi-angle quantum approximate optimization algorithm / R. Herrman [и др.] // Scientific Reports. – 2022. – Т. 12. – Ст. 6781.
5. Georgopoulos, K. Quantum computer benchmarking via quantum algorithms [Электронный ресурс] / K. Georgopoulos, C. Emary, P. Zuliani. – 2021. – 19 с. – Режим доступа: https://arxiv.org/pdf/2112.09457
6. Systematic benchmarking of quantum computers: status and recommendations [Электронный ресурс] / J. M. Lorenz – 2025. – 36 с. – Режим доступа: https://doi.org/10.48550/arXiv.2503.04905
7. Pokharel, B., Lidar, D. A. Demonstration of Algorithmic Quantum Speedup [Текст] / B. Pokharel, D. A. Lidar // Phys. Rev. Lett. – 2023. – Т. 130, № 21.

Статьи студентов магистратуры

Блочное хеширование для построения массива нечетких граней

Р. Б. Анохин

Студент магистрант

С. Д. Махортов

Заведующий кафедрой

Введение

Обработка строк с допустимыми ошибками важна при анализе биологических данных, в частности, при работе с результатами высокопроизводительного секвенирования [1]. В таких задачах возникает необходимость учитывать возможные неточности в сопоставлении фрагментов последовательностей. Одной из важных задач в этом контексте является построение массива граней строки — структуры, описывающей максимальные совпадающие префиксы и суффиксы на каждом префиксе строки [2]. В условиях наличия ошибок задача требует модификации классических алгоритмов и допуска отклонений между префиксом и суффиксом на заданное число замен.

В предыдущей работе [3] была подробно сформулирована данная задача и предложен базовый алгоритм ее решения. Однако прямое сравнение символов влечет за собой квадратичную сложность, а попытки адаптировать классический линейный алгоритм сталкиваются с фундаментальными ограничениями. В данной статье рассматривается реализация алгоритма на основе хеширования. Она позволяет обрабатывать строки с учетом допустимого числа замен за линейное время. Особое внимание уделено организации хеш-функции и оценке производительности полученного результата.

1. Использование хеширования для поиска нечетких граней

Для определенности вначале кратко формализуем задачу построения массива нечетких граней.

Пусть S – строка длины n, k – целое неотрицательное число, H – функция вычисления расстояния Хемминга между строками [4]. Для каждой позиции i от 0 до n требуется найти максимальное L такое, что:

$$H\left(S\left[0..L\right], S\left[L-i..i\right]\right) \le k \qquad (1)$$

© Анохин Р. Б., Махортов С. Д., 2025

В результате образуется массив fbp [0..n], где

$$fbp[i] = max\left\{L \mid H\left(S[0..l], S[i-L..i]\right) \le k\right\}. \quad (2)$$

Для решения задачи поиска массива нечетких граней ранее использовался «наивный» алгоритм [3], который на каждом шаге заново сопоставляет суффикс с добавлением символа в начало и префикс с добавлением одного символа в конец. Этот способ на каждом шаге полностью меняет порядок сравнения, что приводит к необходимости снова проходить по тем же символам, но сравнивать их со сдвигом.

Возможным более эффективным решением задачи является использование хеша, обеспечивающего амортизированное O(1)-обновление. Он должен:

– иметь возможность добавлять символ справа;
– иметь возможность удалять символ слева и справа;
– поддерживать вычисление расстояния до другого хеша.

При наличии такого хеша возможно применение подхода скользящего окна (sliding window), как в алгоритме Рабина-Карпа [5].

Подобный алгоритм представлен в листинге 1. Здесь и далее разработанные алгоритмы будут представлены на языке Си.

Листинг 1

Хеш алгоритм построения массива нечетких граней

```
int *simple_overflow_hash_fbp(const char *str, int n, int k) {
    int *fbp = (int*)malloc(strlen(str) * sizeof(int));
    Hasher *suffix_hash=new_hash(), *prefix_hash = new_hash();
    fbp[0] = 0;
    int left = 1, right = 1, errs = 0;
    while (right < n) {
        if (!is_hash_overflow(prefix_hasher) && ((errs < k) ||
((errs == k) && (str[right] == str[right - left])))) {
            fbp[right] = right - left + 1;
            add_c_to_hash(suffix_hash, str[right]);
            add_c_to_hash(prefix_hash, str[right - left]);
            if (str[right] != str[right - left])
                errs++;
            right++;
        } else {
            remove_last(prefix_hash);
            remove_first(suffix_hash);
            left++;
            errs = calc_errors(prefix_hash, suffix_hash);
        }
    }
    return fbp;
}
```

Данный алгоритм вычисляет для каждой позиции строки длину максимального префикса, приблизительно совпадающего с соответствующим суффиксом при допустимом числе ошибок.

Принцип работы алгоритма:

– последовательная обработка строки, использующая окно сравнения;
– применение хеш-функции для сопоставления префикса и суффикса;
– подсчет числа несовпадений до превышения порога k;
– сдвиг левой границы окна при выходе за предел ошибок;
– сохранение максимальной длины допустимого совпадения для каждой позиции строки.

Все операции над хешем выполняются за время O(B), где B – это количество блоков. Поэтому общая сложность алгоритма составляет O(nB), где n — длина строки. В случае константного числа блоков B=const сложность будет O(n). Этот результат позволяет эффективно обрабатывать длинные последовательности и избегать квадратичных затрат, характерных для «наивных» подходов.

2. Структура блочной хеш функции

Ядром рассматриваемого алгоритма является хеш-функция и набор операций для работы с ней. Для выполнения требований, предъявляемых к хешу, удобно представить его в виде последовательности бит.

В рамках задачи, связанной с анализом ДНК, естественным ограничением служит алфавит из четырех символов — по числу нуклеотидов. Каждый символ можно закодировать двумя битами: «A» — как 00, «T» — 01, «G» — 10 и «C» — 11.

Для хранения закодированных данных используется 64-битный целочисленный тип, он позволяет работать с 32 символами за один такт.

Подход использования блоков для хранения хеша заключается в организации 64-битных хешей в виде массива заданной длины. Такой способ позволяет явно задавать объем хранимой информации, обеспечивая баланс между точностью и скоростью. При увеличении длины массива возрастает точность, но снижается производительность. Напротив, меньший размер ускоряет работу, но может привести к потере точности. Такое соотношение следует из необходимости выполнять операции отдельно для каждого блока в битовой последовательности.

На листинге 2 представлена структура блочного хеша.

Листинг 2

Структура алгоритма блочного хеша

```
typedef struct {
    uint64_t* hashItems; // Массив значений хешей
    uint64_t* maskItems; // Массив битовых масок
    int curItem;         // Индекс текущего элемента
    int curCapacity;     // Длина текущего элемента
    int capacity;        // Общий размер массива
} LongHasher;
```

Хеши хранятся в массиве в виде чисел типа uint64_t, называемых блоками, hashItems – указатель на этот массив. Внутри блока символы располагаются справа налево. Тип uint64_t хранится в памяти как последовательность длиной 64 бит, что позволяет использовать битовые операции для быстрой работы с хешем. Вместе с хешем хранится маска, содержащая единицы на позициях значимых символов. Побитовое «И» маски и хеша позволяет обнулить неиспользуемые биты. В дальнейшем будет показано, как это ускоряет вычисления.

3. Операции добавления и удаления из хеша

При добавлении нового элемента в хеш необходимо произвести битовый сдвиг на две позиции влево и в освободившееся место добавить новый символ. Подобную операцию необходимо произвести только для последнего блока. В листинге 3 приведен код алгоритма.

Листинг 3

Операция добавления символа к хешу

```
void add_c_to_hash(LongHasher *h, char c) {
    if (h->curCapacity >= HASH_ITEM_CAPACITY) {
        h->curCapacity = 0;
        h->curItem += (h->curItem < h->capacity - 1);
    }
    h->hashItems[h->curItem] <<= 2;
    h->hashItems[h->curItem] |= map_char_to_bits(c);
    h->maskItems[h->curItem] <<= 2;
    h->maskItems[h->curItem] |= 0b11;
    h->curCapacity++;
}
```

Несложно также реализовать операцию удаления последнего добавленного элемента remove_last(). Достаточно выполнить битовый сдвиг вправо на 2 бита. Код операции представлен в листинге 4.

Листинг 4

Операция удаления последнего символа из хеша

```
void remove_last(LongHasher *h) {
    if (h->curCapacity == 0 && h->curItem > 0) {
```

Окончание листинга 4

```
        h->curItem--;
        h->curCapacity = HASH_ITEM_CAPACITY;
    }
    h->maskItems[h->curItem] >>= 2;
    h->hashItems[h->curItem] >>= 2;
    h->curCapacity--;
}
```

Для удаления первого добавленного символа требуется убрать ведущие 2 бита в последовательности. Так как это требует последовательного переноса старших битов последующих блоков в младшие биты предыдущих, то эта операция требует использования цикла над всеми блоками, что увеличивает сложность операции до линейной, а всей процедуры — до квадратичной. Рассмотренная операция представлена в листинге 5.

Листинг 5

Операция удаления первого добавленного символа из хеша

```
void remove_first(LongHasher *h) {
    if (h->curCapacity == 0 && h->curItem > 0) {
        h->curItem--;
        h->curCapacity = HASH_ITEM_CAPACITY;
    }
    uint64_t ts =(h->hashItems[h->curItem] & h->maskItems[h-
>curItem]) >>((h->curCapacity - 1) * 2) & 0b11;
    h->maskItems[h->curItem] >>= 2;
    for (int i = h->curItem - 1; i >= 0; i--) {
        uint64_t new_ts = h->hashItems[i] >>
(HASH_ITEM_CAPACITY - 1) * 2 & 0b11;
        h->hashItems[i] <<= 2;
        h->hashItems[i] |= ts;
        ts = new_ts;
    }
    h->curCapacity--;
}
```

4. Операция поиска ошибок между хешами

Оценка числа ошибок между двумя хешами (расстояние Хемминга) выполняется с помощью операции подсчета количества единиц в последовательности бит. Подсчет числа ошибок в одном блоке выполняется за конечное число тактов процессора, что приводит к константной сложности такой операции.

Для подсчета различий между хешами, применяется операция XOR, затем биты разделяются на четные и нечетные с помощью масок 0xAAAAAAAAAAAAAAAA и 0x5555555555555555. После сдвига нечетных битов на один разряд влево и объединения с первой группой в

результирующей последовательности остаются единицы только в тех позициях, где символы различаются. Этот способ позволяет точно подсчитать количество ошибок на уровне символов.

При обработке блоков подобная операция вызывается поэлементно для каждой пары блоков. Итоговая сумма ошибок рассчитывается как сумма значений числа единиц по всем блокам. Этот способ сохраняет линейную сложность алгоритма при умеренном числе блоков.

Схема преобразований блоков представлена на рис. 1, а реализация функции подсчета — в листинге 6.

Рис. 1. Схема битовых операций для поиска числа ошибок между хешами

Листинг 6

Операция поиска числа ошибок между хешами

```
int calc_errors(LongHasher *hasher1, LongHasher *hasher2) {
    int result = 0;
    for (int i = 0; i <= hasher1->curItem; i++) {
        uint64_t x = hash(hasher1, i) ^ hash(hasher2, i);
        uint64_t singleOnes =(x & 0xAAAAAAAAAAAAAAAA) | ((x &
0x5555555555555555) << 1);
        result += (int) _mm_popcnt_u64(singleOnes);
    }
    return result;
}
```

5. Сравнение производительности различных хеш-функций

Для оценки эффективности блочной реализации хеш-функции были проведены бенчмарки на псевдослучайных строках длины от 100 до 100 000 символов. Измерения проводились с использованием Google

Benchmark (3 прогона на каждую точку) [6]. Результаты представлены на графике (рис. 2): каждая кривая соответствует различному количеству блоков хеша.

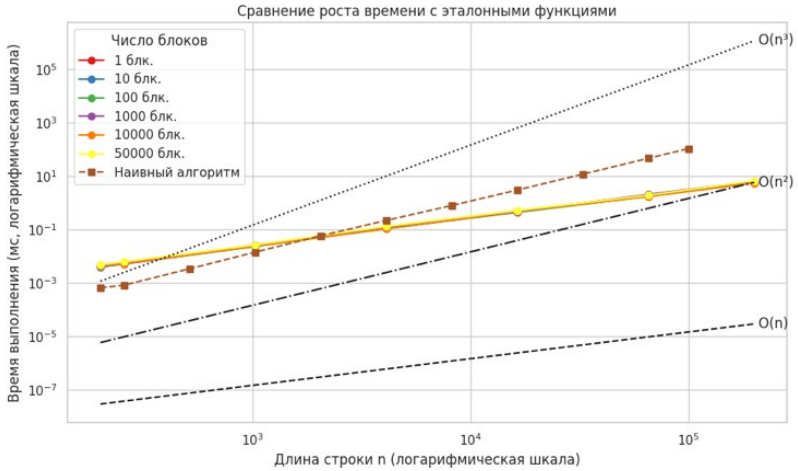


Рис. 2. График роста времени вычислений от длины строки

Как видно, сложность алгоритма остается близкой к линейной — угол наклона кривых на логарифмической шкале соответствует эталонной прямой O(n). При этом ни предел допустимых ошибок k, ни количество блоков до 50 000 практически не влияют на время выполнения.

Так как алгоритм не всегда дает точный результат, то необходимо провести анализ количества ошибок и найти зависимость ошибок от параметров алгоритма. Для оценки точности реализации использовался метод Монте-Карло [7]: алгоритм многократно запускался на случайных строках с различными параметрами, что позволило провести статистическую проверку качества сопоставления. Таблица демонстрирует, что при увеличении числа блоков число ошибок снижается быстрее, чем растет сложность алгоритма. На небольших строках (целевой случай задачи) точность достигает 100% уже при 5 блоках. Верхняя строка в таблице означает процент строк, на которых массив нечетких граней был получен с ошибками. Нижняя строка показывает средний процент ошибок по всем массивам нечетких граней, полученных с ошибкой.

Таблица

Зависимость точности от числа блоков хеш-функции

Размер блч. хеша	10^3 симв. * $4*10^6$ стр.	10^4 симв. * $2*10^5$ стр.	10^5 симв * 10^3 стр.	10^6 симв. На 100 стр.
1	30.323%	6.875%	57%	0%
	45.554%	22.288%	18.274%	0.0004%
5	100%	97.55%	98%	1%
		98.594%	98.045%	10.015%
10	100%	100%	100%	2%
				10.015%
20	100%	100%	100%	3%
				30.045%
>20	100%	100%	100%	-

Таким образом, блочная структура позволяет гибко настраивать компромисс между точностью и производительностью, а при разумном выборе параметров сохраняет как скорость, так и надежность вычислений.

Заключение

Предложенный метод демонстрирует устойчивую линейную сложность на практике даже при большом количестве блоков хеширования. Несмотря на теоретическую возможность квадратичного поведения в худшем случае (при использовании параметра числа блоков сопоставимым с длиной строки), в реальных условиях алгоритм показывает высокую производительность (близкую к линейной) и приемлемую точность. Благодаря гибкости архитектуры хеширования, метод может быть адаптирован под конкретные ограничения по скорости или точности и применяться для эффективного построения массива нечетких граней в задачах с ошибочными или вариативными данными.

Список литературы

1. Single-molecule, quantitative detection of low-abundance somatic mutations by high-throughput sequencing / A.Y. Maslov [et al.] // Science Advances. – 2022. – Vol. 8. – Iss. 14. – P. 1–6. – DOI: 10.1126/sciadv.abm3259.

2. Махортов, С. Д. О выразительных возможностях схем описания структуры строк / С. Д. Махортов // Программная инженерия. – 2018. – № 5. – С. 235–240. – Текст : непосредственный.

3. Анохин, Р. Б. Алгоритм вычисления массива граней с нечетким сопоставлением префикса и суффикса / Р. Б. Анохин, С. Д. Махортов // Материалы Международной научно-практической конференции им. Э. К. Алгазинова «Информатика: проблемы, методы, технологии» (IPMT, 12–13 февраля 2025 г., Воронеж). – 2025. – С. 53–68.

4. Hamming, R. W. Error Detecting and Error Correcting Codes / R. W. Hamming // Bell System Technical Journal. - 1950. - Vol. 29. - Iss. 2. - P. 147-160.

5. Rabin, M. Efficient randomized pattern-matching algorithms / M. Rabin, R. Karp // IBM Journal of Research and Development. - 1987. - Vol. 31. - Iss. 2. - P. 249-260.

6. GoogleTest: Google Testing and Mocking Framework [Электронный ресурс]. – Режим доступа: https://google.github.io/googletest/ (дата обращения: 02.05.2025).

7. The Monte Carlo Method / N. Metropolis [et al.] // Journal of Chemical Physics. - 1949. - Vol. 21. - Iss. 6. - P. 1087-1092.

Деградация современной компьютерной графики в фильмах и видеоиграх

Р. М. Бозоров

Студент магистр

А. Ф. Клинских

Профессор

Введение

Период 2010–2015: Расцвет физически корректного рендеринга (PBR) и реализма. В это время основной акцент делался на повышении детализации и физической достоверности графики.

Ключевые технологии и тренды:

1. Physically Based Rendering (PBR) – стал стандартом в индустрии, обеспечивая реалистичное взаимодействие света с материалами (металлы, пластик, кожа).
2. Тесселяция и displacement mapping – использовались для добавления геометрической детализации без чрезмерной нагрузки на GPU (например, в Crysis 3, 2013).
3. Глобальное освещение (Global Illumination) – внедрение методов, таких как Voxel Cone Tracing (CryEngine 3) и Light Propagation Volumes (Unreal Engine 4), для реалистичного рассеянного света.
4. Фотореализм в кино – фильмы вроде «Аватара» (2009) и «Гравитации» (2013) показали возможности CGI, сочетая motion capture и сложный рендеринг.

Период 2021–2025: Доминирование нейросетей [1–3] и критика «ленивой» графики [4, 5]. Современный этап характеризуется массовым внедрением ИИ, но также и спорами о потере детализации.

Ключевые технологии и тренды:

1. Нейросетевые текстуры и апскейлинг – технологии вроде NVIDIA DLSS 3.0 и AMD FSR 3.0 позволяют рендерить игры в 4K с меньшими затратами, но иногда ценой артефактов.
2. Метавселенные и цифровые двойники – Epic Games MetaHuman и Unreal Engine 5 Nanite обеспечивают фотореалистичные модели без LOD-ов.

© Бозоров Р. М., Клинских А. Ф., 2025

3. Критика «размытой» графики – TAA (Temporal Anti-Aliasing) + DLSS создают эффект «мыльной» картинки (Call of Duty: Modern Warfare III, 2023).

4. Возврат к математическим методам – SDF-моделирование (Unreal Engine 5) и воксельные технологии предлагают альтернативу нейросетям.

1. Ручное моделирование и шейдинг

Период 2010-2015 годов стал золотой эрой ручной работы в компьютерной графике, когда качество визуализации достигалось за счет глубокого понимания физики света и кропотливой работы художников. В это время доминировал подход Physically Based Rendering (PBR), который радикально изменил стандарты реализма в индустрии.

Основу PBR составляло точное моделирование взаимодействия света с поверхностями через уравнение рендеринга:

$$L_0(p,\omega) = L_e(p,\omega_0) + \int_\Omega f_r(p,\omega_i,\omega_0) L_i(p,\omega_i)(n\cdot\omega_i)d\omega_i \quad (1)$$

L_0 – исходящая световая энергия

f_r – BRDF (Bidirectional Reflectance Distribution Function)

L_i – входящее освещение

$(n\cdot\omega_i)$ - косинус угла падения

Ключевым достижением стало внедрение микрофакетных моделей, описывающих поверхность как множество микроскопических граней. Наиболее популярная модель GGX (Trowbridge-Reitz) задавала распределение нормалей:

$$D(m) = \alpha^2 / [\pi((n\cdot m)^2(\alpha^2-1)+1)^2], \quad (2)$$

где α - параметр шероховатости поверхности.

В игровых движках (CryEngine 3, Unreal Engine 4) активно использовалась техника split-sum approximation для эффективного вычисления зеркальных отражений.

2. Гибридные подходы и рендеринг в реальном времени

Период 2016-2020г. ознаменовался революцией в интерактивном рендеринге благодаря трем ключевым технологиям: аппаратной трассировке лучей, нейросетевым методам реконструкции изображения и новым алгоритмам процедурной генерации. Рассмотрим их математические основы.

Нейросетевой апскейлинг (DLSS). Сверточные сети для реконструкции:

$$f(x) = \sigma(W_n \cdot (...\sigma(W_1 \cdot x + b_1)... + b_n), \quad (3)$$

где W_i - обучаемые веса upscale-фильтров.

Функция потерь включала:

$$L = \lambda_1 \cdot L_{MSE} + \lambda_2 \cdot L_{VGG} + \lambda_3 + L_{GAN} \quad (4)$$

Процедурная генерация. Дифференцируемый шум для миров:

$$W(x) = \sum_i^j w_{ij} \cdot B(x / 2^{i-j}), \quad (5)$$

где B – базисные функции.

Этот период показал, как математика и аппаратура совместно решают задачи, которые раньше считались неразрешимыми в реальном времени.

3. Нейросети портят качество

Подмена качества апскейлингом (фейковый 4К).

Современные нейросетевые методы апскейлинга (DLSS/FSR/XeSS) используют сверточные сети для реконструкции изображения из низкого разрешения.

Рис. 1. Изображение одного и того же кадра из видеоигры, справа используются нейросетевые модели, слева изображение оригинальное

При сравнении двух изображений на рис. 1. можно заметить на правом изображении снижение детализации. Это называется ореолы, (артефакты изображения), они чаще всего проявляются при апскейлинге изображения нейросетевыми моделями, если камера и объекты находятся в движении. Изображение на рис.1. было сделано в движении.

Такие модели можно представить математически:

$$I_{hi} = f_0(I_{l0}) + \varepsilon, \quad (6)$$

где f_0 – нейросеть, а ε – артефакты генерации.

Проблема в том, что нейросети склонны к чрезмерному сглаживанию. Вместо физически корректного рендеринга нейросети генерируют правдоподобные, но не точные пиксели. Это особенно заметно в динамических сценах, где возникает «мерцание» из-за несогласованности между кадрами.

Ещё один вид моделей, который используется в разработке видеоигр, – диффузионные. Они позволяют художникам генерировать текстуры (картинки), которые будут отображаться на объекте или материале. Достаточно часто теряется связь с материалом, и проявляются логические ошибки, и если художник не исправит всё вручную, то качество итоговой работы будет плохое.

4. Математические альтернативы нейросетям и сбалансированный подход.

Современные нейросетевые технологии в графике, несмотря на их эффективность, создали три ключевые проблемы:

1. Потерю физической достоверности – ИИ часто «додумывает» детали, нарушая реальные законы света и материалов.
2. Артефакты генерации – размытости, мерцание и «пластиковый» вид изображений.
3. Упрощение творческого процесса – автоматизация лишает художников контроля над деталями.

Какие могут быть альтернативы?

Усовершенствованная растеризация. Технологии вроде Nanite в Unreal Engine 5 показывают, что можно рендерить миллионы полигонов без LOD-упрощений Адаптивная тесселяция сохраняет детализацию только там, где это действительно нужно.

Физически точные алгоритмы сжатия. Вейвлет-сжатие текстур дает лучшее качество чем нейросетевые JPEG AI. Детерминированный апскейлинг на основе частотного анализа сохраняет резкость границ.

Процедурные методы включают в себя параметрическое моделирование материалов, например на основе шумов Перлина. Аналитическая генерация деталей через математические функции.

Выводы

При создании современной компьютерной графики лучше всего использовать комбинированный подход. Таким образом можно будет достичь лучшего качества и производительности.

Где использовать нейросети:

– Постобработка: тонкая коррекция цветов, шумоподавление
– Вторичные элементы: далекая листва, атмосферные эффекты

– Апскейлинг UI и HUD-элементов

Использование алгоритмов:

– Основная геометрия и материалы персонажей
– Ключевое освещение и тени
– Важные визуальные эффекты (вода, стекло, металлы)

Использование такого подхода поможет стабилизировать качество изображения, избежать появления артефактов, обеспечит аппаратную эффективность, так как математические методы часто менее требовательны к видеопамяти GPU, а главное, поможет сохранить авторский стиль. Художники остаются полными хозяевами ключевых визуальных элементов.

Заключение

Индустрия уже движется в этом направлении – примеры вроде Unreal Engine 5.4 демонстрируют, как можно сочетать математически строгие методы с избирательным использованием ИИ без потери качества. Будущее – за разумным балансом между физикой, математикой и нейросетями, где каждая технология применяется там, где она действительно незаменима.

Список литературы

1. Голышева, Е. Н. Основные подходы к генерации изображений с помощью нейронных сетей / Е. Н. Голышева // Инновационная наука. – 2023. – № 11/2. – С. 25–28.

2. Ильинская, Е. В. Применение генеративно-состязательных нейросетей для генерации изображений / Е. В. Ильинская // Научный результат. Информационные технологии. – 2024. – Т. 9, № 1. – С. 73–78.

3. Тимошевич, К. С. Перспективы и проблемы развития нейросетей для генерации изображений и текстов / К. С. Тимошевич, К. А. Котова // Компьютерные системы и сети. – Минск, 2024. – С. 575–577.

4. Ахромеева, Т. С. Искусственный интеллект как проблема культуры / Т. С. Ахромеева // Обсерватория культуры. – 2020. – Т. 17, № 3. – С. 228–241.

5. Дружинина, А. А. Художник и нейросеть: конфликт, диалог, сотрудничество? / А. А. Дружинина // Вестник ЮУрГУ. Серия: Социально-гуманитарные науки. – 2024. – № 2. – С. 58–65.

Система автоматизированного принятия решения для задачи выбора метода идентификации пользователя

О. С. Бондарева

Студент магистр

Е. Ю. Митрофанова

Доцент

Введение

Принятие решений является неотъемлемой частью жизни человека. Когда решения становятся многокритериальными, возрастает сложность выбора. Системы поддержки принятия решений (СППР) помогают объективно оценить альтернативы на основе важных для пользователя критериев.

Процесс автоматизированного принятия решения будет построен на методе анализа иерархии (далее – МАИ) как на интуитивно понятном, универсальном и лаконичным многокритериальном методе принятия решения.

В данной статье будет рассматриваться решение задачи выбора методов идентификации пользователя.

1. Методы идентификации пользователя

Биометрическая идентификация представляет собой процесс сравнения и определения сходства между данными человека и его биометрическим «шаблоном». Биометрия позволяет идентифицировать и провести верификацию человека на основе набора специфических и уникальных черт, присущих ему от рождения [1].

Рассмотрим следующие методы, основанные на геометрии БХЧ пользователя [2, 3].

1. Метод идентификации пользователя по геометрии руки

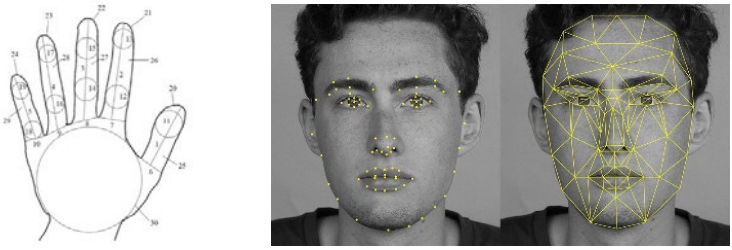
Метод основан на выявлении ключевых особенностей геометрии руки, таких как длина пальцев, длина фаланг, ширина пальцев, периметры фаланг и др. [4].

2. Метод идентификации пользователя по геометрии лица

В работе будет использоваться двумерный подход к идентификации лица. Основными биометрическими признаками, анализируемых при 2D-идентификации показаны на рис. 1. При создании биометрического

© Бондарева О. С., Митрофанова Е. Ю., 2025

шаблона лица учитываются индивидуальные особенности строения черепа, которые остаются неизменными на протяжении длительного времени.



а б

а – геометрии руки человека, б – для 2D-идентификации лица

Рис. 1. Биометрические признаки

2. Метод анализа иерархии

Одним из самых популярных, лаконичных и понятных человеку, является метод анализа иерархии, который был разработан Т. Саати [5].

Рис. 2. Иерархия проблемы выбора метода идентификации пользователя

Основа этого метода – декомпозиция сложной проблемы на простые составные части. Метод анализа иерархии позволяет лицу, принимающему решение (ЛПР), системно оценивать различные альтернативы путем попарного сравнения критериев и вариантов решения. Для решения текущей задачи была разработана следующая иерархия, представленная на рис. 2.

3. Используемые модели идентификации

На рынке существуют уже готовые решения для идентификации человека по его биометрическим характеристикам. Одно из таких решений для идентификации человека по геометрии его руки – предобученная модель компьютерного зрения MediaPipe Hands, разработанная Google [6].

Архитектура этой модели включает в себя сверточные нейронные сети (CNN). Модель была обучена на 30 тысячах изображений человеческих рук.

Модель представляет собой двухэтапный пайплайн:

1. Модель детектирования: на вход модели подается изображение (или видеопоток), на котором определяется область местоположения руки.

2. На обрезанной области ладони регрессионной моделью предсказываются координаты, описывающие местоположение точек.

Сама модель зачастую используется распознавания жестов, в приложениях дополненной реальности, для управления с помощью жестов.

Если модель обнаружила руку, она возвращает координаты точек (меток), показанных на рисунке ниже. Также модель способна предсказать, какую руку ей показывают – левую или правую.

Рис. 3. Метки, определяемые с помощью модели MediaPipe Hands

На вход модели подается видеопоток или изображение произвольного размера с каналами RGB. В результате выполнения модели реализовывалось обнаружение руки, т.е. координаты меток руки.

Помимо модели идентификации руки будет рассматриваться и уже обученная модель для распознавания лиц face_recognition. Стоит упомянуть, что различных моделей и программ распознавания лиц в сети представлено большое количество, однако мы остановимся только на одной. Сравнительный анализ, проведенный автором в статье [7], показал, что рассматриваемая модель обладает высокой точностью распознавания.

Архитектура модели face_recognition представляет собой остаточную сеть ResNet с 29 сверточными слоями. Это версия модели, представленная в статье [8] с несколькими удаленными слоями и уменьшенным вдвое количеством фильтров.

Обучающая выборка составляла порядка трех миллионов изображений. Точность модели (метрика accuracy) составляет 0,9938 на тестовой выборке, которая включает в себя 13233 изображений порядка 6 тысяч людей.

На вход модели подается видеопоток или изображение произвольного размера с каналами RGB. Модель распознает человеческое лицо и выделяет нужную часть изображения, пример представлен на рисунке ниже.

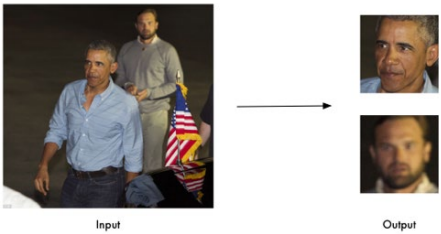


Рис. 4. Идентификация лица человека

После обнаружения лица на изображении модель определяет положение 68 ключевых точек лица.

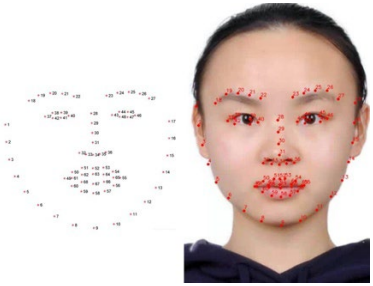


Рис. 5. Ключевые точки для идентификации человека по лицу

На основе взаимного расположения этих точек создается 128-мерный вектор признаков, который и используется для идентификации человека.

Отклонение между векторами признаков одного и того же человека минимально. Таким образом, распознавание лиц возможно, если

сопоставить вектор на выходе модели эталонного изображения с вектором изображения на входе.

Если евклидово расстояние между векторами будет слишком велико, то модель распознает лицо как отличающееся от эталонного (аутентификация таким образом пройдена не будет). Если евклидово расстояние будет составлять пороговое или меньшее (в данной модели оно заявлено как 0,6), то сравниваемая пара векторов будет определяться как принадлежащих одному и тому же человеку.

4. Реализация системы принятия решения

Реализация приложения выбора метода идентификации и непосредственно идентификация пользователя осуществлялась с помощью языка программирования Python, библиотек OpenCV для работы с изображениями и tkinter для разработки интерфейса.

Разработанное приложение выполняется следующим образом:

Рис. 6. Схема работы приложения

После запуска появляется интерфейс, в котором пользователь проводит попарное сравнение критериев между собой. При выборе и нажатии кнопки «Далее» данные с формы записываются в массив для дальнейшей обработки.

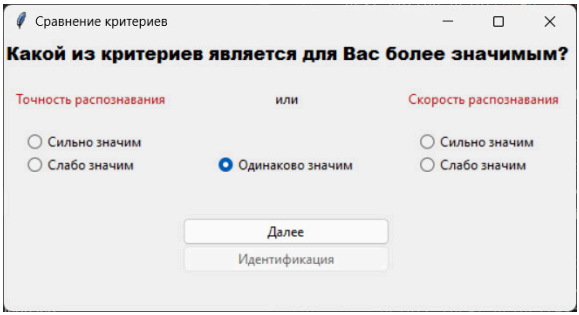
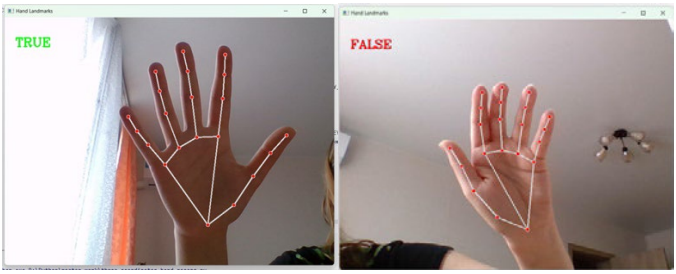


Рис. 7. Попарное сравнение критериев

После завершения сравнения критериев кнопка «Далее» отключается для пользователя и становится активной кнопка «Идентификация».

На рисунке ниже приведен этап идентификации пользователя по геометрии его руки.



а б

а – при совпадении с эталоном, б – при отсутствии совпадения

Рис. 8. Идентификация по геометрии руки

На рис. 9 показан пример, когда доступ пользователю разрешен при распознавании лица, т.е. распознанное лицо сходится с эталоном.

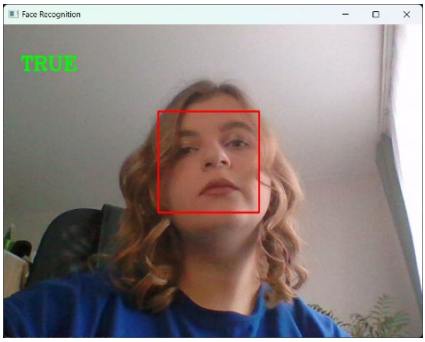


Рис. 9. Идентификация по лицу при совпадении с эталоном

Заключение

Реализована система автоматизированного принятия решения по выбору метода идентификации пользователя. В качестве базового подхода для принятия решения был выбран метод анализа иерархий.

В соответствии с поставленной задачей разработана программа, реализованная с помощью Python. В настоящее время эталонные значения заданы разработчиком в приложении. В дальнейшем система дополнится интерфейсом регистрации эталонов и дополнительной защитой от несанкционированного доступа в виде парольного ввода.

Сравнения как для идентификации по лицу, так и для идентификации по руке сейчас проводятся только с одним эталоном. Предполагается дополнение несколькими эталонами, чтобы повысить точность распознавания обоими методами.

Список литературы

1. Фролова, Е. Ю. Идентификация человека по биометрическим данным: обзор современных технологий / Фролова Е. Ю., Кошлыкова Ю. А. // Северо-Кавказский юридический вестник. – 2022. – №3. – С. 167-174.

2. Биометрическая аутентификация в Windows [Электронный ресурс]. – Режим доступа: https://www.osp.ru/winitpro/2010/03/13002555

3. Спиридонов, И. Н. Многоагентная биометрическая система / Спиридонов И. Н., Гаврюшин С. С., Евгенев Г. Б. // Известия высших учебных заведений. – 2012. – №5. – С. 60-78.

4. Baca, M. Basic Principles and Trends in Hand Geometry and Hand Shape Biometrics / M. Baca, P. Grd, T. Fotak // New Trends and Developments in Biometrics. – InTech. – С. 77–99.

5. Саати, Т. Принятие решений. Метод анализа иерархий / Т. Саати; пер. с англ. Р. Г. Вачнадзе – М.: Радио и связь, 1993. – 320 с.

6. MediaPipe Hands: On-device Real-time Hand Tracking / F. Zhang [et. al.] // arXiv preprint arXiv:2006.10214. – 2020.

7. Каляшов, Е. В. Сравнительный анализ систем распознавания лиц, построенных с использованием блоков стандартных архитектур / Е. Каляшов // Информационные технологии и телекоммуникации. – 2020. – № 3. – С. 94-101.

8. High Quality Face Recognition with Deep Metric Learning [Электронный ресурс]. – Режим доступа: https://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html

Методы распознавания печатного текста в мобильных приложениях на Flutter

В. Э. Борисенков

Студент магистр

В. С. Тарасов

Ст. преподаватель

Введение

В современном мире мобильные приложения становятся неотъемлемой частью повседневной жизни, предоставляя пользователю широкий спектр функциональности. Одним из ключевых аспектов их развития является возможность распознавания печатного текста. Эта функция находит применение в различных сценариях, включая сканирование и распознавание документов, перевод текстов, а также цифровизацию рукописных заметок.

В данной научной работе рассматриваются современные методы распознавания печатного текста в мобильных приложениях, разработанных на платформе Flutter. Flutter, созданный компанией Google, представляет собой кроссплатформенный фреймворк, который позволяет разработчикам создавать высокопроизводительные приложения с единым кодом для iOS и Android.

Основное внимание уделено сравнительному анализу различных технологий распознавания текста, включая Tesseract OCR, Google ML Kit и Firebase ML. Каждая из этих технологий имеет свои уникальные особенности, предоставляя разработчикам инструменты для решения задачи распознавания текста с различной степенью точности, скорости и поддержки языков.

В контексте быстро меняющихся требований информационной безопасности и повсеместного использования мобильных устройств, выбор подходящего метода распознавания текста становится ключевым аспектом для разработчиков и конечных пользователей. Понимание преимуществ и ограничений каждой технологии помогает эффективно интегрировать функциональность распознавания текста в приложения Flutter, обеспечивая высокое качество работы и удовлетворение потребностей пользователей.

© Борисенков В. Э., Тарасов В. С., 2025

1. Tesseract OCR

Tesseract OCR (Optical Character Recognition) — это мощная библиотека с открытым исходным кодом для распознавания текста, разработанная и поддерживаемая сообществом разработчиков. Изначально создана в Hewlett-Packard Laboratories в 1985 году [1], позже была перенесена в Google, где и получила широкое распространение и развитие. В настоящее время Tesseract является одним из самых популярных инструментов для распознавания текста и поддерживает более 100 языков [2]. Распространяется под лицензией Apache 2.0, что позволяет всем пользователям свободно использовать, модифицировать и распространять данную технологию без юридических ограничений. Также технология позволяет использовать в качестве входных данных различные форматы изображений, такие как: JPEG, PNG, TIFF, BMP и другие [3]. В качестве выходных данных может предоставить форматы TXT, HTML, PDF и др.

Принцип работы данной технологии можно наблюдать на рис. 1 [4].

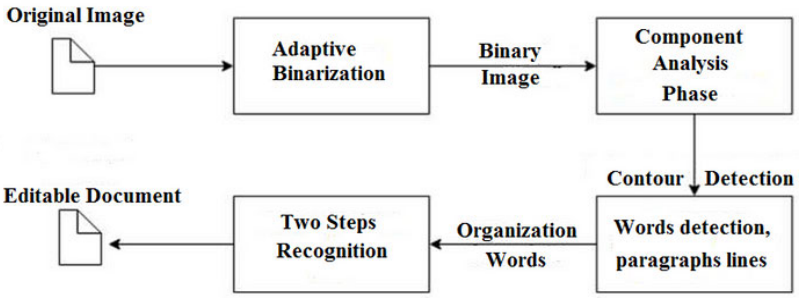


Рис. 1. Принцип работы Tesseract OCR

Он заключается в прохождении нескольких этапов, а именно:

1. Предобработка изображения.

Прежде чем Tesseract начнет распознавание текста, изображение проходит ряд предобработок, чтобы улучшить его качество и сделать текст более читаемым для алгоритмов распознавания. Это включает:

- бинаризацию (преобразование изображения в черно-белое, чтобы выделить текст на фоне. Это помогает устранить шум и улучшить контрастность);
- удаление шума (Применение фильтров для удаления мелких артефактов и шума, которые могут мешать распознаванию текста);

– выравнивание (Коррекция углов наклона текста для выравнивания строк. Tesseract использует алгоритмы для обнаружения и исправления наклона текста);
– нормализацию (Регулировка яркости и контрастности изображения для улучшения четкости символов).

2. Сегментация.

После предобработки Tesseract выполняет сегментацию изображения, чтобы разделить его на отдельные элементы. Сегментация происходит в следующем порядке: сегментация строк, сегментация слов, сегментация символов.

3. Распознавание текста.

На этом этапе Tesseract применяет алгоритмы распознавания для преобразования изображений символов в текст. Используются методы машинного обучения для сопоставления изображений символов с соответствующими буквами, цифрами и другими знаками. Tesseract использует обученные модели для распознавания символов. После этого происходит компоновка распознанных символов в слова. Tesseract использует словари для проверки и исправления распознанных слов.

4. Постобработка текста.

После распознавания текста Tesseract выполняет постобработку для улучшения качества результата. Она состоит из:

– коррекции ошибок (использование языковых моделей и словарей для исправления ошибок в распознанных словах);
– форматирования (Сохранение формата текста, включая пробелы, абзацы и другие элементы разметки).

5. Вывод результатов.

После завершения всех этапов распознанный текст выводится в удобном формате.

Пример небольшого кода, который будет выводить весь текст, который будет найден на изображении, можно увидеть в листинге 1.

Листинг 1

Распознавание текста с изображения на русском и английском языках Tesseract OCR

```
Future<void> _pickImageAndExtractText() async {
    final XFile? image = await _picker.pickImage(source:
ImageSource.gallery);
    if (image != null) {
```

Окончание листинга 1

```
      String text = await
FlutterTesseractOcr.extractText(image.path, language:
"eng+rus");
      setState(() {
        _extractedText = text;
      });
    }
  }
```

2. Google ML Kit

Google ML Kit — это мощный набор инструментов и API для мобильных приложений, разработанный компанией Google. Он предоставляет возможность интегрировать в свои приложения функции машинного обучения, такие как распознавание текста, обнаружение лиц, классификация изображений, отслеживание объектов и многие другие. ML Kit предназначен для упрощения и ускорения внедрения передовых технологий машинного обучения в мобильные приложения, обеспечивая при этом высокую производительность и точность.

Также особенностью данной технологии является кроссплатформенность, что позволяет без проблем использовать один инструментарий как в случае разработки под iOS, так и под Android [5].

ML Kit предоставляет набор предварительно обученных моделей машинного обучения с возможностью использования как локально, так и через подключение к облачному серверу, которые охватывают распространенные случаи использования, такие как распознавание текста, сканирование штрих-кода или обнаружение и отслеживание объектов. При этом можно использовать и собственные модели. Возможно либо обучить свою собственную модель с помощью TensorFlow или TensorFlow Lite, либо использовать модель из AutoML Vision Edge или AutoML Natural Language [6]. ML Kit упрощает развертывание и обновление пользовательских моделей с помощью Firebase, а также предоставляет показатели производительности и использования моделей.

Из недостатков можно выделить поддержку меньшего количества языков, в отличие от Tesseract, но это компенсируется более обширными возможностями библиотеки, что позволяет использовать ее в формате «все в одном».

Принцип разборки текста похож на Tesseract, это последовательное сегментирование полученного текста из изображения на блоки (или абзацы), линии (или строки), элементы (или слова) и символы. Более подробная схема представлена на рис. 2 [7].

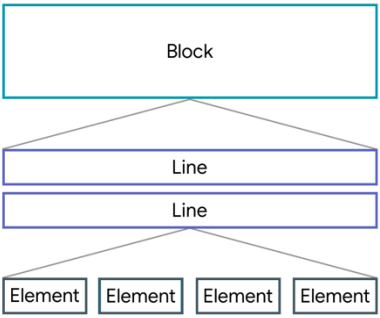


Рис. 2. Принцип работы Google ML Kit

Хоть и ML Kit работает гораздо быстрее Tesseract, существенным минусом является отсутствие поддержки русского языка, что сильно ограничивает использование данной технологии от Google.

С примером кода для взаимодействия с ML Kit можно ознакомиться в листинге 2.

Листинг 2

Пример реализации распознавания текста с использованием Google ML Kit

```
Future<void> _recognizeText() async {
    final XFile? image = await _picker.pickImage(source:
ImageSource.gallery);
    if (image != null) {
      final inputImage = InputImage.fromFilePath(image.path);
      final RecognizedText recognizedText =
          await _textRecognizer.processImage(inputImage);
      setState(() {
        _recognizedText = recognizedText.text;
      });
    }
  }
```

3. EasyOCR

EasyOCR — это инструмент для оптического распознавания символов (OCR), разработанный с использованием библиотек PyTorch и deep learning моделей. Он также позволяет извлекать текст из изображений и поддерживает более 80 языков. EasyOCR разработан компанией Jaided AI и отличается простотой интеграции и высокой точностью распознавания текста. Отличается от других рассматриваемых технологий тем, что реализован на Python, из-за чего требуется дополнительный сервер-обработчик, но без затруднений работает через http запросы.

Присутствие Deep Learning моделей в составе технологии позволяет использовать и дообучать модели посредством таких технологий, как Tensorflow или PyTorch, что расширяет возможности взаимодействия с данной технологией. В качестве недостатков можно выделить требовательность к ресурсам, а именно к наличию хорошей видеокарты, которая влияет на производительность данной технологии при обработке большого объема текста.

Принцип работы обладает практически такими же этапами, как и другие модели и отличается лишь использованием других технологий (рис. 3.) [8]. Самым главным отличием служит возможность вывода результата в разнообразных форматах текстовых файлов благодаря использованию Python.

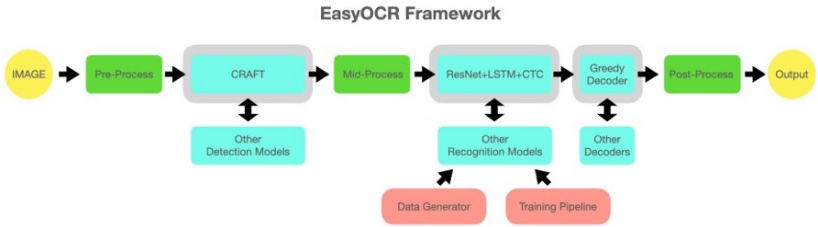


Рис. 3. Принцип работы EasyOCR

Серверную часть технологии, запускаемую на Python с помощью библиотеки Flask, представлена в листинге 3.

Листинг 3

Реализация EasyOCR с помощью Flask

```
def ocr():
    if 'image' not in request.files:
        return jsonify({"error": "No image file provided"}),
400
    try:
        file = request.files['image']
        image = cv2.imdecode(np.frombuffer(file.read(),
np.uint8), cv2.IMREAD_COLOR)
        results = reader.readtext(image)
        extracted_text = "\n".join([result[1] for result in
results])
        return jsonify({"text": extracted_text})
    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

4. Сравнение рассмотренных технологий

Для сравнения Tesseract, Google ML Kit и EasyOCR выбраны следующие метрики:

4Character Error Rate (CER);

5Word Error Rate (WER);

6Throughput (строк или страниц в секунду);

7Avg Confidence (%);

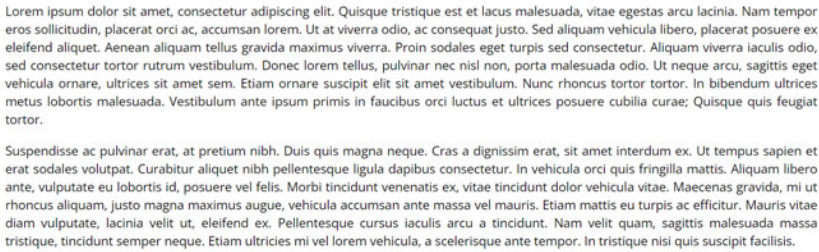
8CPU Usage (%) и RAM Usage (MB);

9Font Coverage (%);

10Supported Languages (количество).

По результатам сравнения будет составлена таблица с итогами, по которой можно будет определить какая библиотека подходит для мобильного приложения.

С примером одного из тестовых изображений, с помощью которых производилось сравнение технологий распознавания, можно ознакомиться на рис. 4.


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque tristique est et lacus malesuada, vitae egestas arcu lacinia. Nam tempor eros sollicitudin, placerat orci ac, accumsan lorem. Ut at viverra odio, ac consequat justo. Sed aliquam vehicula libero, placerat posuere ex eleifend aliquet. Aenean aliquam tellus gravida maximus viverra. Proin sodales eget turpis sed consectetur. Aliquam viverra iaculis odio, sed consectetur tortor rutrum vestibulum. Donec lorem tellus, pulvinar nec nisl non, porta malesuada odio. Ut neque arcu, sagittis eget vehicula ornare, ultrices sit amet sem. Etiam ornare suscipit elit sit amet vestibulum. Nunc rhoncus tortor tortor. In bibendum ultrices metus lobortis malesuada. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Quisque quis feugiat tortor.

Suspendisse ac pulvinar erat, at pretium nibh. Duis quis magna neque. Cras a dignissim erat, sit amet interdum ex. Ut tempus sapien et erat sodales volutpat. Curabitur aliquet nibh pellentesque ligula dapibus consectetur. In vehicula orci quis fringilla mattis. Aliquam libero ante, vulputate eu lobortis id, posuere vel felis. Morbi tincidunt venenatis ex, vitae tincidunt dolor vehicula vitae. Maecenas gravida, mi ut rhoncus aliquam, justo magna maximus augue, vehicula accumsan ante massa vel mauris. Etiam mattis eu turpis ac efficitur. Mauris vitae diam vulputate, lacinia velit ut, eleifend ex. Pellentesque cursus iaculis arcu a tincidunt. Nam velit quam, sagittis malesuada massa tristique, tincidunt semper neque. Etiam ultricies mi vel lorem vehicula, a scelerisque ante tempor. In tristique nisi quis suscipit facilisis.

Рис. 4. Тестовое изображение для сравнения технологий

Разберем значения метрик, их формулы и реализацию на dart.

1. Character Error Rate (CER)

Доля символов, распознанных неверно относительно эталонной (ground truth) строки.

$$CER = \frac{D_{lev}(ref, hyp)}{|ref|} \times 100\%, \quad (1)$$

где D_{lev} — расстояние Левенштейна между эталонной строкой ref и распознанной строкой hyp, а $|ref|$ — длина эталонной строки в символах.

2. Word Error Rate (WER)

Доля слов, распознанных неверно относительно эталонного текста.

$$WER = \frac{D_{lev}(W_{ref}, W_{hyp})}{|W_{ref}|} \times 100\%, \quad (2)$$

где W_{ref} и W_{hyp} — последовательности слов эталона и гипотезы, разделённые пробелами, а $|W_{ref}|$ — число слов в эталоне.

3. Throughput

Скорость обработки текста: число строк (или страниц), распознанных за единицу времени.

$$Throughput = \frac{T_{proc}}{N_{lines}}, \quad (3)$$

где N_{lines} — общее число распознанных строк (или страниц), T_{proc} — время обработки (в секундах).

4. Avg Confidence

Средний процент оценки уверенности модели в каждом распознанном символе или слове.

$$AvgConf = \frac{1}{N} \sum_{i=1}^{N} c_i \times 100\%, \quad (4)$$

где $c_i \in [0,1]$ — confidence-score i-го элемента, N — общее число элементов (символов или слов).

5. Font Coverage

Доля корректно распознанных образцов текста, заданных в разных шрифтах и стилях.

В таблице показаны усреднённые значения каждой из метрик, полученные для трёх OCR-движков (Tesseract, Google ML Kit и EasyOCR) на едином наборе тестовых изображений. Такие результаты позволяют наглядно сравнить необходимость и условия использования каждой из систем.

Таблица

Сравнительная таблица систем

Метрика	**Tesseract OCR**	**Google ML Kit**	**EasyOCR**
CER, %	5.0	3.5	4.0
WER, %	12.0	9.0	10.2
Throughput, стр./с	0.9	1.3	1.1
Avg confidence, %	84	89	87
CPU usage, % (макс.)	60	55	50

Окончание таблицы

RAM usage, MB	110	140	160
Latency p90, мс	650	500	550
CER при +20 % Gaussian noise, %	10.0	8.0	8.5
Font coverage, % корректно	75	88	80
Количество поддерживаемых языков	100	25	80
End-to-end F1 (таблицы/формы), %	60	68	65

Заключение

В данной статье был рассмотрен небольшой спектр методов и технологий для распознавания печатного текста в мобильных приложениях на основе Flutter. Был проведён детальный сравнительный анализ трёх технологий: Tesseract OCR, Google ML Kit и EasyOCR. На основе полученных результатов можно сделать вывод, что среди рассмотренных алгоритмов наиболее подходящим для использования в мобильных приложениях является Google ML Kit.

Выбор Google ML Kit обоснован следующими аргументами:

– Высокая точность распознавания текста, подтверждённая наиболее низкими показателями CER (3,5%) и WER (9,0%) по сравнению с Tesseract OCR и EasyOCR.
– Наивысшая производительность (Throughput — 1,3 стр./с, Latency — 500 мс), что критично для мобильных приложений, требующих оперативного взаимодействия.
– Высокий показатель уверенности (89%), демонстрирующий стабильность и надежность системы.
– Лучшая поддержка различных шрифтов (Font Coverage — 88%), что важно при работе с документами разнообразного формата и стиля.

Не смотря на довольно маленькое количество поддерживаемых языков, Google ML Kit является довольно качественным инструментом для обработки текста локально на устройстве. Однако при необходимости распознать какие-то языки, которые не входят в состав Google ML Kit, может быть целесообразно применять комбинированный подход с использованием Tesseract OCR в том случае, когда приложение планируется масштабировать на дополнительные языки или работать с узкоспециализированными текстами. Это позволит эффективно

расширить функциональные возможности приложения, сохранив при этом высокий уровень точности и производительности.

Таким образом, в рамках локального распознавания Google ML Kit OCR является оптимальным решением: сочетая низкие показатели CER и WER, высокую производительность и простоту интеграции во Flutter-приложение, он закрывает основные требования проекта.

Список литературы

1. Википедия — свободная энциклопедия [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Tesseract
2. Klippa [Электронный ресурс]. Режим доступа. Режим доступа: https://www.klippa.com/en/blog/information/tesseract-ocr/
3. GitHub [Электронный ресурс]. Режим доступа: https://github.com/tesseract-ocr
4. Medium [Электронный ресурс]. Режим доступа: https://medium.com/nanonets/a-comprehensive-guide-to-ocr-with-tesseract-opencv-and-python-fd42f69e8ca8
5. Хабр [Электронный ресурс]. Режим доступа: https://habr.com/ru/companies/yoomoney/articles/461867/
6. Fritz ai [Электронный ресурс]. Режим доступа: https://fritz.ai/build-custom-machine-learning-models-for-android/
7. Google for Developers [Электронный ресурс]. Режим доступа: https://developers.google.com/ml-kit/vision/text-recognition/v2?hl=ru
8. GitHub [Электронный ресурс]. Режим доступа: https://github.com/JaidedAI/EasyOCR

Реализация рекомендательной системы на основе двухстороннего вариационного автоэнкодера и GraphRag

Д. С. Бурнашев

Студент магистр

М. Е. Залыгаева

Старший преподаватель

Введение

С каждым годом объём цифровой информации стремительно растёт, что обуславливает необходимость в интеллектуальных алгоритмах, способных эффективно обрабатывать и интерпретировать большие и разнородные данные. Рекомендательные системы становятся важнейшим инструментом, помогающим пользователям находить релевантный контент — от фильмов и книг до товаров и сервисов — среди множества доступных вариантов.

В данной работе рассматривается реализация гибридной рекомендательной системы, объединяющей два современных подхода: двухсторонний вариационный автоэнкодер (BiVAE) и GraphRAG.

BiVAE применяется для построения персонализированных рекомендаций на основе истории взаимодействия пользователя с фильмами, моделируя скрытые предпочтения пользователей и характеристики фильмов в латентном пространстве.

В свою очередь, GraphRAG используется для генерации рекомендаций по текстовому описанию, вводимому пользователем — например, когда он описывает, какой фильм хотел бы посмотреть. Эта модель сочетает графовую структуру данных и механизм извлечения внешней информации (retrieval-augmented generation), что позволяет выдавать рекомендации, опираясь не только на структуру графа, но и на семантику пользовательского запроса.

Такой подход позволяет охватить как сценарии с известной историей взаимодействия, так и cold-start-сценарии, где пользователь вводит только своё описание желаемого контента.

© Бурнашев Д. С., Залыгаева М. Е., 2025

1. Двунаправленный вариационный автоэнкодер для коллаборативной фильтрации (BiVAE-CF)

BiVAE-CF представляет собой расширение классического вариационного автоэнкодера (VAE) [1], адаптированное под задачу рекомендаций на разреженных матрицах «пользователь × объект». В контексте данной работы объектами являются фильмы, а матрица содержит оценки или взаимодействия пользователей с фильмами (например, из набора MovieLens 100K).

Классические методы коллаборативной фильтрации (SVD, NMF) сводят всю информацию о пользователях и объектах к единым точечным векторным представлениям, что ограничивает гибкость модели и не учитывает неопределённость в предпочтениях. BiVAE-CF же вводит по два латентных вектора: для каждого пользователя u — θ_u и для каждого фильма i — β_i.

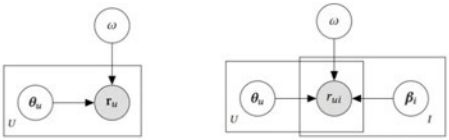


Рис. 1. Схематичное изображение VAE (слева) и BiVAE (справа)

Пользовательский энкодер $q(\theta_u \mid r_{u*})$, отображает вектор всех оценок пользователя $(r_{u1},...,r_{um})$ в апостериорное распределение латентных факторов θ_u. Таким образом, энкодер пользователей формирует аппроксимированное апостериорное распределение:

$$q(\theta_u \mid r_{u*}) = \mathrm{N}(\mu_\psi(r_{u*}), diag[\sigma_\psi(r_{u*})^2]) \quad (1)$$

где $r_{u*} = (r_{u1},...,r_{um})$ – вектор оценок пользователя по всем фильмам. Параметры μ_ψ и σ_ψ вычисляются нейросетевыми слоями энкодера.

Энкодер фильмов $q(\beta_i \mid r_{*i})$, преобразует вектор оценок по фильму i от всех пользователей $(r_{1i},...,r_{ni})$ в распределение латентных признаков β_i. Таким образом, энкодер фильмов задаёт апостериорное распределение:

$$q(\beta_i \mid r*_i) = \mathrm{N}(\mu_\phi r*_i), diag[\sigma_\phi (r*_i)^2]), \quad (2)$$

где $r*_i = (r_{1i},...,r_{ni})$ – вектор взаимодействий всех n пользователей с фильмом i. Такой двунаправленный подход позволяет одновременно учитывать информацию со стороны пользователя и со стороны фильма.

Декодер (предсказатель рейтинга) вычисляет ожидаемую оценку пользователя u для фильма i как скалярное произведение латентных векторов:

$$\hat{r}_{ui} = \theta_u^T \beta_i \quad (3)$$

Обучение модели основано на максимизации нижней оценки логарифма маргинального правдоподобия (ELBO):

$$L_{ELBO} = \mathrm{E}_{q(\theta_u)q(\beta_i)}\left[\log p(r_{ui} \mid \theta_u, \beta_i)\right] - \quad (4)$$

$$-KL(q(\theta_u \mid r_{u*}) \| p(\theta_u)) - KL(q(\beta_i \mid r*_i) \| p(\beta_i)),$$

где априорные распределения $p(\theta_u)$ и $p(\beta_i)$ принимаются стандартными нормальными $\mathrm{N}(0, I)$.

Для обеспечения дифференцируемости процесса семплирования применяется приём репараметризации [2].

$$\theta_u = \mu_\psi(r_{u*}) + \sigma_\psi(r_{u*}) \otimes \varepsilon, \quad \beta_i = \mu_\phi(r*_i) + \sigma_\phi(r*_i) \otimes \varepsilon, \ \varepsilon \sim \mathrm{N}(0, I) \quad (5)$$

BiVAE обеспечивает более глубокое и гибкое моделирование взаимодействий между пользователями и фильмами за счёт вероятностных латентных представлений. Это позволяет учитывать неопределённость, устойчиво работать с разреженными данными и достигать баланса между точностью и обобщающей способностью благодаря оптимизации через ELBO и KL-регуляризацию.

2. Модификация GraphRag для рекомендаций фильмов на основе введенного описания

GraphRAG представляет собой гибридную архитектуру, сочетающую возможности языковых моделей и графов знаний для извлечения, структурирования и поиска информации [3]. Система работает в два этапа. На этапе индексации из текстов автоматически извлекаются сущности и их взаимосвязи с использованием LLM. Проводится разрешение дубликатов и профилирование сущностей, после чего формируется граф знаний, где узлами выступают как текстовые фрагменты, так и сущности (персонажи, темы,

эмоциональные состояния и др.), а рёбрами — семантически значимые связи между ними.

На этапе поиска пользовательский запрос преобразуется в векторное представление, из которого извлекаются как низкоуровневые элементы (факты, объекты), так и высокоуровневые категории (темы, эмоции, концепты). На основе этих ключей осуществляется навигация по графу для выявления релевантных узлов, отношений и исходных контекстов.

Каждая сущность в графе описывается именем, типом и кратким пояснением, отражающим её роль в тексте. Пары взаимосвязанных сущностей дополняются описанием характера их связи, оценкой её силы и тематическими ключевыми словами. Дополнительно формируются обобщающие концепции высокого уровня, описывающие основные сюжетные и эмоциональные доминанты.

Для реализации рекомендации фильмов для введенного пользователем описания используются короткие описания-тизеры, что позволяет сопоставить каждый фильм отдельному текстовому чанку (текстовой части), который используется для получения сущностей, связей и ключевых слов.

Пример выделенных сущностей, связей и ключевых слов из описания фильма, полученного с помощью разработанного программного модуля представлен на рис. 2.

В рамках финального этапа каждый блок оценивается по степени релевантности запросу, после чего ранжируется и отбираются наиболее информативные. Суммарный ранг используется для выбора оптимального набора фильмов, формирующего локальный контекст для LLM. Модель генерирует обоснованный ответ с финальным списком релевантных идентификаторов фильмов, который затем передаётся в микросервис для формирования пользовательских рекомендаций с описаниями, названиями и постерами.

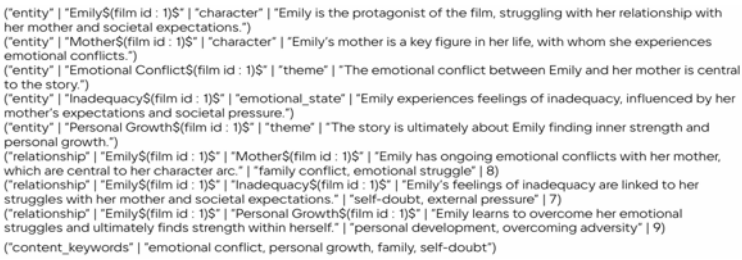

("entity" | "Emily$(film id : 1)$" | "character" | "Emily is the protagonist of the film, struggling with her relationship with her mother and societal expectations.")
("entity" | "Mother$(film id : 1)$" | "character" | "Emily's mother is a key figure in her life, with whom she experiences emotional conflicts.")
("entity" | "Emotional Conflict$(film id : 1)$" | "theme" | "The emotional conflict between Emily and her mother is central to the story.")
("entity" | "Inadequacy$(film id : 1)$" | "emotional_state" | "Emily experiences feelings of inadequacy, influenced by her mother's expectations and societal pressure.")
("entity" | "Personal Growth$(film id : 1)$" | "theme" | "The story is ultimately about Emily finding inner strength and personal growth.")
("relationship" | "Emily$(film id : 1)$" | "Mother$(film id : 1)$" | "Emily has ongoing emotional conflicts with her mother, which are central to her character arc." | "family conflict, emotional struggle" | 8)
("relationship" | "Emily$(film id : 1)$" | "Inadequacy$(film id : 1)$" | "Emily's feelings of inadequacy are linked to her struggles with her mother and societal expectations." | "self-doubt, external pressure" | 7)
("relationship" | "Emily$(film id : 1)$" | "Personal Growth$(film id : 1)$" | "Emily learns to overcome her emotional struggles and ultimately finds strength within herself." | "personal development, overcoming adversity" | 9)
("content_keywords" | "emotional conflict, personal growth, family, self-doubt")

Рис. 2. Пример выделенных сущностей, связей и ключевых слов из описания фильма

Пример графического пользовательского интерфейса системы рекомендаций фильмов по введенному сюжету от пользователя на основе разработанного программного модуля представлен на рис. 3.

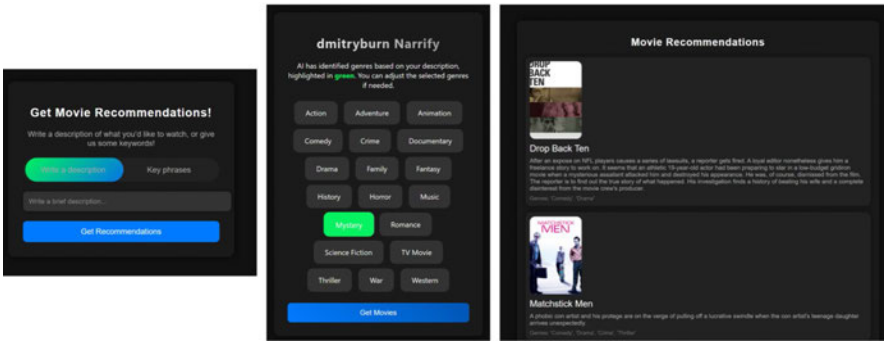


Рис. 3. Графический интерфейс сервиса рекомендаций фильмов на основе введенного пользователем желаемого сюжета

3. Архитектура рекомендательной системы

Рекомендательная система предоставляет пользователям персонализированные рекомендации фильмов, опираясь на их предпочтения и историю взаимодействия [4]. В процессе регистрации и онбординга пользователи указывают любимые жанры, что позволяет системе предварительно адаптировать рекомендации. Система динамически уточняет предпочтения за счёт пользовательских оценок просмотренных фильмов, повышая точность рекомендаций. Через пользовательский интерфейс осуществляется интеграция с модулем GraphRAG, который обрабатывает текстовые запросы, извлекает ключевые сущности и формирует релевантный контекст для генерации ответов. Вся информация о предпочтениях, оценках и истории взаимодействий хранится в централизованной базе данных PostgreSQL, обеспечивая надёжность и масштабируемость системы.

Графический пользовательский интерфейс основных элементов рекомендательной системы представлен на рис. 4, рис. 5 и рис. 6.

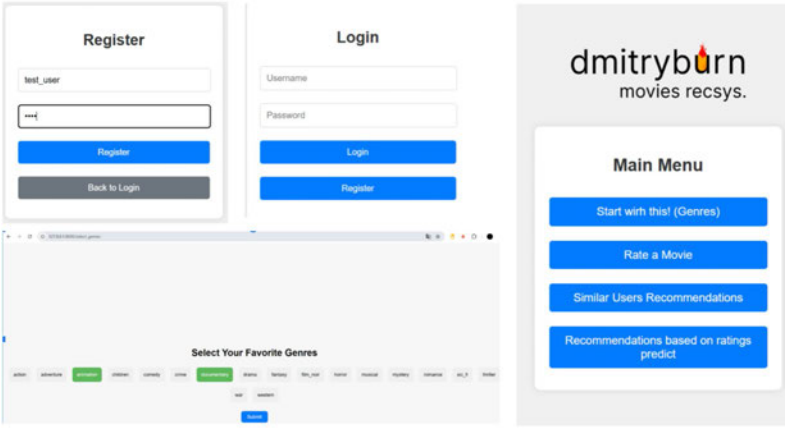


Рис. 4. Графический интерфейс регистрации, онбординга и основного меню

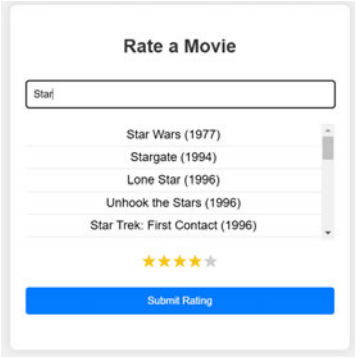


Рис. 5. Графический интерфейс оценки фильма

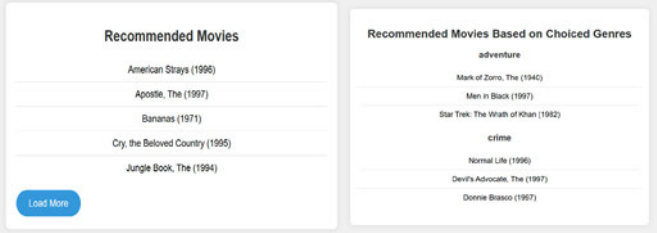


Рис. 6. Графический интерфейс получения рекомендаций.

4. Оценка качества рекомендаций для модели на основе двунаправленного автоэнкодера

Для оценки качества рекомендаций на тестовой выборке использовались следующие метрики:

$$\text{Precision}@k = \frac{\text{relevant}@k}{k} \tag{6}$$

где relevant@k — число релевантных элементов в топ-K

$$MMR = \frac{1}{|U|}\Sigma\frac{1}{rank_u} \tag{7}$$

где $rank_u$ — позиция первого релевантного элемента для пользователя u.

Результат оценки качества работы модели на основе метрик в сравнении с старым подходом на основе матричной факторизации представлен на рис. 6.

Метрика	BiVAE (новый подход)	SVD (старый подход)
Precision@K	0.208	0.049
MRR	0.35	0.15

Рис. 7. Пример локального изменения яркости

BiVAE значительно превосходит классическую матричную факторизацию (SVD) по всем метрикам качества рекомендаций. Повышенные значения Precision@K и MRR свидетельствуют о большей релевантности и точности выдачи, особенно в топ-K результатах.

Заключение

Данная статья посвящена разработке и реализации гибридной рекомендательной системы, сочетающей вероятностную коллаборативную фильтрацию на основе двунаправленного вариационного автоэнкодера (BiVAE) и генеративный модуль поиска GraphRAG. Такой подход позволил объединить преимущества персонализированного моделирования пользовательских предпочтений и семантически обоснованного ответа на текстовые запросы. Проведённые эксперименты показали, что BiVAE существенно превосходит классические методы матричной факторизации по метрикам точности рекомендаций. Внедрение модуля GraphRAG расширило функциональные возможности системы, обеспечив качественные рекомендации даже при отсутствии пользовательской истории (cold start). Представленная архитектура демонстрирует

высокую адаптивность, масштабируемость и точность, что делает её перспективным решением для современных интеллектуальных рекомендательных систем.

Список литературы

1. Truong, Q.-T. Bilateral Variational Autoencoder for Collaborative Filtering / Q.-T. Truong, A. Salah, H. W. Lauw // Proceedings of the 14th ACM International Conference on Web Search and Data Mining. – 2021.

2. Salah, A. C.: A Comparative Framework for Multimodal Recommender Systems / A. C. Salah, Q.-T. Truong, H. W. Lauw // Journal of Machine Learning Research. – 2020. – Vol. 21, No 95. – PP. 1–5.

3. Han, H. Retrieval-Augmented Generation with Graphs (GraphRAG) [Электронный ресурс] : arXiv preprint / H. Han, Y. Wang, H. Shomer. – 2024. – Режим доступа: https://arxiv.org/abs/2501.00309

4. Бурнашев, Д. С. Реализация рекомендательной системы на основе сингулярного разложения в Python / Д. С. Бурнашев // Сборник трудов молодых исследователей факультета компьютерных наук ВГУ. – Воронеж : ВГУ, 2023. – С. 11–19.

Разработка программного обеспечения для анализа состояния приемников и проведения базового тестирования

Д. А. Быченко

Студент магистр

С. А. Самодуров

Доцент

Введение

Современные радиотехнические системы требуют эффективного анализа и тестирования оборудования, используемого в различных областях, таких как радиолокация, радионавигация и связь. Одним из важнейших компонентов таких систем являются приемники, которые играют ключевую роль в обработке поступающих сигналов. Для обеспечения надежности и качества работы этих устройств необходимо регулярно проверять их корректность и эффективность работы в реальных условиях.

Существуют различные типы антенн и приёмных модулей, которые могут использоваться как в одиночных конфигурациях, так и в составе антенных решёток. Так, например, пирамидальные ТЕМ-рупоры с диэлектрическими линзами могут применяться в качестве направленных элементов в решётках для СВЧ диапазона [1]. В свою очередь, многоканальные приёмные системы могут быть использованы не только в задачах радиолокации, но и в современных системах беспроводной связи, включая архитектуры MIMO-OFDM. Для них актуальны методы оценки канала и адаптивного подавления помех, требующие согласованной и синхронной работы нескольких приёмников [2].

В настоящее время существует множество инструментов и программного обеспечения, предназначенного для тестирования и мониторинга работы приемников. Однако большинство из них ограничено по функциональности или сложны в настройке и использовании. Для решения этих проблем требуется создание универсального и доступного программного решения, которое обеспечит базовое тестирование приемников, анализ их состояния и корректности работы при взаимодействии в связке.

© Быченко Д. А., Самодуров С. А., 2025

В рамках данной работы предлагается разработка программного обеспечения, которое будет проводить анализ состояния подключенных приемников, а также выполнять базовые тесты для оценки их работоспособности в сети. Ожидается, что данная система позволит эффективно наблюдать за работой приемников, выявлять возможные ошибки и дефекты в их функционировании, а также проводить диагностику для обеспечения бесперебойной работы.

1. Анализ предшествующих работ

В настоящее время существует ряд программных решений, позволяющих осуществлять прием и анализ радиосигналов. Однако большинство из них ориентировано на визуализацию спектров или общую обработку сигнала, а не на комплексное тестирование приемников и анализ их согласованной работы в связке. Рассмотрим несколько наиболее известных решений.

Одним из популярных инструментов является SDRangel — кроссплатформенное программное обеспечение, предназначенное для приема и обработки радиосигналов в реальном времени. SDRangel поддерживает широкий спектр устройств, включая RTL-SDR, и предоставляет пользователю мощные средства визуализации, такие как спектрограмма и диаграмма мощности. Однако работа с программой предполагает значительное участие оператора и ручную настройку, а функциональность автоматического тестирования или оценки работы нескольких приемников одновременно в системе отсутствует.

Другим распространенным решением является CubicSDR. Это легкое и простое в использовании приложение, предназначенное для визуализации и управления SDR-приемом. Оно предоставляет базовые инструменты для наблюдения спектра сигнала и не требует сложной настройки. Тем не менее, CubicSDR не поддерживает работу с несколькими приемниками одновременно. Также программное обеспечение не предназначено для проведения тестов оборудования и не предоставляет информации о техническом состоянии подключенных устройств.

Еще одной системой является GNURadio — модульная платформа с открытым исходным кодом, ориентированная на построение систем цифровой обработки сигналов. Она предоставляет обширный набор инструментов и библиотек, позволяющих реализовать практически любые алгоритмы анализа. GNURadio может использоваться для параллельной обработки сигналов с нескольких источников. Основным недостатком этой системы является высокая сложность освоения, необходимость ручной настройки блоков и отсутствие

специализированных средств для мониторинга и диагностики состояния приемников.

Таким образом, существующие программные решения обладают рядом недостатков, ограничивающих их применение для задач автоматизированного тестирования и анализа работы нескольких радиоприемников. Среди основных проблем можно выделить отсутствие синхронной обработки данных с разных устройств, слабую или отсутствующую поддержку анализа технического состояния оборудования, а также высокую сложность настройки и эксплуатации большинства систем.

Разрабатываемое программное обеспечение должно отличаться своей доступностью, простотой использования и возможностью масштабируемого анализа данных от нескольких приемников. Основной задачей является реализация системы, способной автоматически тестировать подключенные устройства, оценивать их корректность работы в связке и предоставлять результаты в удобной форме без необходимости сложной конфигурации.

2. Методы и материалы

В качестве приемных устройств использовались радиоприемники типа RTL-SDR. Это компактные USB-устройства, предназначенные для приема радиосигналов в диапазоне от 25 МГц до 1,7 ГГц. Они обеспечивают цифровой вывод сигнала и подключаются напрямую к вычислительной машине, что делает их удобными для построения тестовых стендов. Ключевым преимуществом RTL-SDR является их низкая стоимость, простота интеграции собственного программного обеспечения и возможность подключения нескольких приемников одновременно, что критично для анализа их согласованной работы [3].

Разработка программного обеспечения осуществлялась на языке Python. Этот язык отличается простым синтаксисом, кроссплатформенностью и широкой поддержкой. Благодаря наличию множества библиотек, Python позволяет быстро реализовывать алгоритмы цифровой обработки сигналов, проводить численный анализ и строить наглядные графики. Выбор Python также обусловлен его возможностью запуска на различных операционных системах, что позволит тестировать программу на разных вычислительных устройствах.

Для взаимодействия с приемниками использовалась библиотека pyrtlsdr. Она предоставляет удобный интерфейс для настройки параметров RTL-SDR, таких как частота дискретизации, центральная частота и усиление. Кроме того, pyrtlsdr поддерживает параллельную работу с несколькими устройствами, что делает возможным синхронный

прием сигналов с разных антенн. Это является необходимым условием для анализа фазовых сдвигов и задержек между каналами [4].

Численная обработка данных выполнялась с использованием библиотек NumPy и SciPy. NumPy применялась для операций с массивами комплексных чисел, необходимых при работе с сигналами в I/Q-формате, а также для выполнения базовых математических операций. SciPy использовалась для фильтрации сигналов с помощью цифровых фильтров, расчета корреляций, оценки задержек и других операций, связанных с анализом временных и частотных характеристик. Эти библиотеки обеспечивают высокую производительность и надежность расчетов.

Для визуализации данных использовалась библиотека Matplotlib. С её помощью строились графики амплитуд сигналов, фазовые диаграммы и полярные диаграммы направленности. Возможность отображения данных как в декартовой, так и в полярной системе координат позволяет более наглядно представить пространственные характеристики сигнала и определить направление на источник радиосигнала.

3. Результаты и их обсуждение

Работа программы начинается с инициализации трёх RTL-SDR приемников. Для каждого устройства устанавливаются одинаковые параметры: частота дискретизации 2,4 МГц, центральная частота 100 МГц и автоматическое усиление. После настройки происходит считывание фиксированного количества отсчетов (в данном случае 1024) с каждого приемника. Эти отсчеты представляют собой комплексные значения (I / Q компоненты), описывающие полученный радиосигнал.

После получения данных выполняется проверка наличия сигнала. Для этого используется булева операция np.any(samples), которая возвращает True, если массив содержит ненулевые значения. Эта проверка позволяет убедиться в исправности приёмников и наличии активного сигнала в выбранном диапазоне. Данный шаг выполняется перед основным анализом, чтобы исключить случаи потери сигнала или аппаратных сбоев.

Затем программа рассчитывает взаимную корреляцию сигналов между каждой парой антенн. Корреляция используется для оценки схожести формы сигналов и наличия общего источника. Она рассчитывается по формуле 1, где $S_i[n]$ и $S_j[n]$ – сигналы с приемников i и j, а $\overline{S_j}$ - комплексное сопряжение.

$$\mathrm{Corr}_{ij} = \sum_n S_i[n] \bullet \overline{S_j[n]} \quad (1)$$

Результат — комплексное число, где модуль отражает степень совпадения сигналов, а фаза — их относительный сдвиг. Для каждой пары приёмников анализируется максимальное значение корреляции.

Следующий этап — расчет фазовой разности между сигналами. Он проводится с использованием точечной операции над массивами по формуле 2.

$$\varphi_{ij}[n] = \arg(S_i[n] \cdot \overline{S_j[n]}) \Rightarrow \overline{\varphi_{ij}} = \frac{1}{N} \sum_n \varphi_{ij}[n] \quad (2)$$

Средняя разность фаз $\overline{\varphi_{ij}}$ показывает смещение фаз фронтов сигналов между антеннами и используется для дальнейшего пространственного анализа. Это значение особенно важно при оценке направленности системы, так как фаза отражает геометрическое положение антенн относительно источника.

На заключительном этапе строится грубая диаграмма направленности. Она реализована по простой модели, отображенной по (3).

$$P(\theta) = \left|\cos(\theta - \overline{\varphi_{12}})\right| + \left|\cos(\theta - \overline{\varphi_{13}})\right|. \quad (3)$$

Здесь θ представляет собой направление в полярной системе координат, а $\overline{\varphi_{ij}}$ - усредненные фазовые сдвиги. Такая диаграмма не претендует на точность фазированной антенной решетки, но позволяет наглядно оценить доминирующее направление прихода сигнала.

В ходе исследования было произведено тестирование программы на трех подключенных приёмниках, результат которого можно увидеть на рисунке 1. Полученные данные подтверждают наличие сигнала на всех приёмниках, что свидетельствует о корректной работе оборудования. Значения корреляции между парами приёмников являются высокими, что говорит о наличии общей структуры в принимаемых сигналах. Разности фаз между антеннами находятся в пределах 0.05 радиан, что указывает на небольшое пространственное смещение. Это также говорит о том, что сигналы были получены синхронно и с одного источника, с незначительной разницей фаз фронта.

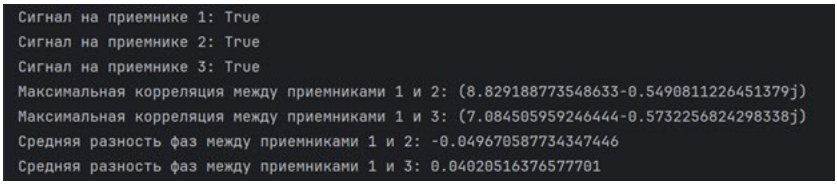


Рис. 1. Результат работы программы на примере трех подключенных приемниках

Таким образом, программа корректно реализует базовые тесты анализа сигналов: проверку сигнала, корреляционный анализ и расчет фазовых смещений. Итоговая визуализация позволяет получить представление о направлении источника, что может служить основой для дальнейшего развития более точных методов пространственной обработки.

Заключение

В рамках проведенной работы было реализовано программное обеспечение, предназначенное для анализа состояния радиоприемников и оценки их согласованной работы в составе единой системы. Основной функционал программы включает в себя прием сигналов с нескольких приёмников, проверку их активности, вычисление взаимной корреляции и средних фазовых сдвигов между сигналами. На основе полученных данных выполняется построение диаграммы направленности, что позволяет делать первичные выводы о направлении на источник сигнала и согласованности работы антенн. Все расчеты основаны на стандартных математических процедурах, адаптированных под радиотехнические задачи.

Разработанная архитектура позволяет расширять систему за счёт подключения новых алгоритмов анализа. Каждый тип теста — будь то анализ спектра, фильтрация, определение направленности или диагностика синхронизации — может быть оформлен в виде отдельного программного модуля. Это обеспечивает гибкость и возможность постоянного расширения функционала без необходимости изменения основной логики программы.

Полученные результаты подтверждают работоспособность предложенного подхода и открывают возможность дальнейшего развития системы, включая внедрение более точных методов пространственной обработки сигналов и аппаратных средств синхронизации приёмников.

Список литературы

1. Возможность применения пирамидального ТЕМ-рупора с круглой диэлектрической линзой / С. А. Антипов [и др.] // Радиотехника. – 2014. – № 3. – С. 7-10.

2. Ромащенко, М. А. Усовершенствованный метод оценки канала с итерационным подавлением помех для многопользовательских систем MIMO-OFDM / М. А. Ромащенко, О. Н. Чирков, П. П. Чураков // Радиотехника. – 2019. – Т. 83, № 6(8). – С. 150-155.

3. RTL-SDR.com [Электронный ресурс]. – Режим доступа : https://www.rtl-sdr.com/about-rtl-sdr/

4. Osmocom [Электронный ресурс]. – Режим доступа : https://osmocom.org/projects/rtl-sdr/wiki

Интеграция игры на Unity с Web3

В. И. Валеев

Студент магистр

В. С. Тарасов

Ст. преподаватель

Введение

NFT игры являются довольно молодым и вместе с этим до сих пор недостаточно раскрытым направлением в разработке. Однако у них видится большой потенциал ввиду ряда факторов: наличия концепции P2E (play-to-earn), то есть заработка во время игры, возможности однозначного подтверждения на владение активом и принципиальной особенности самих NFT – невзаимозаменяемости (отсюда и их название – non-fungible token).

При разработке такого рода приложения разработчику важно разбираться в игровом движке, в Web3 и в их интеграции.

Целью данной работы является выявление наиболее удобного инструмента для работы с Web3 в Unity и реализация ключевых для разрабатываемой NFT игры функций. Также было определено, что данная тематика довольно плохо раскрыта и достаточно проблематично найти актуальные сведения, что в свою очередь подтверждает актуальность исследования.

1. Постановка задачи

В соответствии с поставленной целью следует решить следующие задачи:

– провести сравнительный анализ Web3-платформ;
– выбрать Web3-платформу для интеграции с Unity;
– провести интеграцию Web3-платформы с Unity.

Ниже приведено схематической представление интеграции Web3-платформы с Unity (рис. 1):

© Валеев В. И., Тарасов В. С., 2025

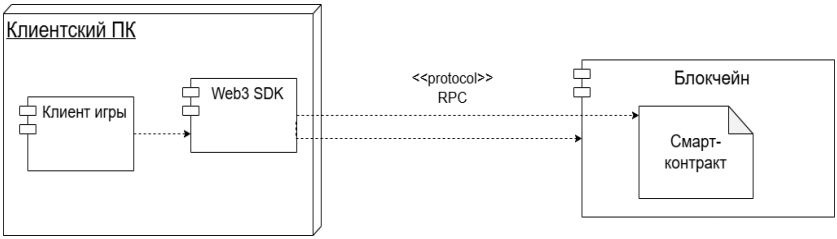


Рис. 1. Схема взаимодействия NFT игры с блокчейном

2. Выбор платформы для работы с Web3

В настоящее время существует ряд платформ для работы с Web3. У каждой есть свои особенности, сильные и слабые стороны. Были рассмотрены следующие варианты для интеграции: Thirdweb [1], Moralis [2], Nethereum [3], Web3 Unity [4]. Ниже приведена сравнительная таблица, в которой отражено соответствие рассматриваемых платформ ключевым критериям.

Таблица

Сравнительная таблица Web3-платформ

	Thirdweb	**Moralis**	**Nethereum**	**Web3 Unity**
Поддержка блокчейнов	Множество	Множество	Ethereum	Ethereum
Простота интеграции	Высокая	Средняя	Средняя	Высокая
Кастомизация	Высокая	Средняя	Высокая	Низкая
Документация	Хорошая	Отличная	Хорошая	Средняя

Простота интеграции измеряется временными затратами на подключение SDK. Кастомизация отражена объемом доступной функциональности и возможностью детально настраивать под нужды проекта. Качество документации определялось по ее доступности, стуктурированности и подробности описания.

По совокупности критериев Thirdweb определен наиболее подходящим, поэтому в дальнейшим будет рассмотрена интеграция этой платформы с Unity.

3. Интеграция Thirdweb с Unity

В первую очередь необходимо скачать SDK, который доступен в репозитории на Github Thirdweb [5]. SDK представляет собой файл с расширением .unitypackage, который добавляется в проект стандартным

для Unity-проектов образом: в строке меню нажать на Assets и выбрать Import Package.

После импорта пакета необходимо будет настроить специальный объект Thirdweb Manager, в котором необходимо установить значения для атрибутов Client ID (API ключа для SDK) и для Bundle ID – (корневого пакета проекта). Пример настройки представлен на рис. 2:

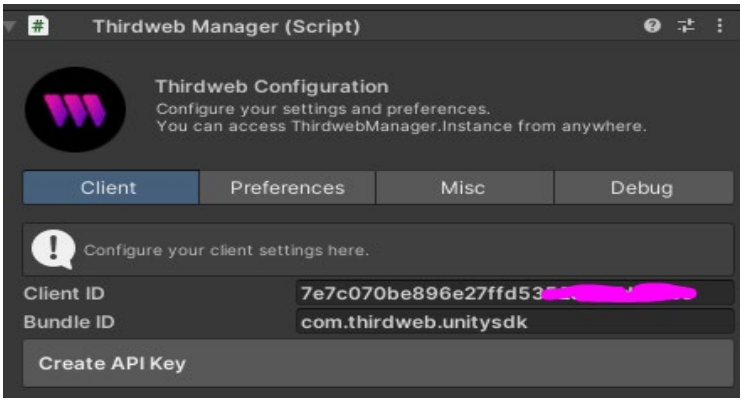


Рис. 2. Окно с конфигурацией клиента SDK

После настройки Thirdweb Manager во всём проекте становится доступным являющийся синглтоном экземпляр ThirdwebManager.Instance, с помощью которого будет происходить взаимодействие с блокчейном. Для NFT игры наиболее важными являются следующие возможности: аутентификация пользователя в блокчейне, чеканка NFT, получение NFT. Их реализации будут продемонстрированы в дальнейшем.

Пример реализации аутентификации, а если точнее, вызова окна аутентификации в криптокошельке Metamask в блокчейне (листинг 1):

Листинг 1

Пример вызова окна аутентификации криптокошелька

```
var walletConnection = new WalletConnection()
            {
                provider = WalletProvider.MetaMask,
                chainId = 137 // polygon's id
            };
string address = await
ThirdwebManager.Instance.SDK.wallet.Connect(walletConnection);
```

Непосредственно аутентификация произойдет в Metamask. С помощью chainId задается идентификатор блокчейна, в данном случае

это Polygon. После аутентификации игровой клиент, имея адрес криптокошелька пользователя, может инициировать различные операции в блокчейне и конкретных смарт-контрактах, в частности действия с NFT, речь о которых пойдет дальше.

Пример реализации чеканки NFT (листинг 2):

Листинг 2

Пример чеканки NFT по стандарту ERC-1155

```
var sdk = ThirdwebManager.Instance.SDK;
var metadata = new NFTMetadata {
                name = "My NFT",
                image = "ipfs url to image"
            };
string uri = await
sdk.storage.UploadText(JsonUtility.ToJson(metadata));

var nftContract = sdk.GetContract("address to contract with
mint function");

var result = await nftContract.ERC1155.Mint(uri);
```

Для успешной чеканки вызываемый смарт-контракт должен поддерживать эту операцию. Допустимо использование других стандартов, например, ERC-721. Сам объект с метаданными, вообще говоря, может быть произвольным и его наполнение зависит только от потребностей логики игры и, возможно, специфичной логики смарт-контракта. Также стоит подчеркнуть, что на самом деле вызывается окно криптокошелька, в котором пользователь должен подтвердить или отклонить транзакцию с чеканкой. Таким образом полученный result необходимо корректно обрабатывать, исходя из каждого возможного исхода.

Примеры получения всех NFT, которые были созданы смарт-контрактом и NFT конкретного пользователя (листинг 3):

Листинг 3

Примеры получения NFT

```
var sdk = ThirdwebManager.Instance.SDK;

var nftContract = sdk.GetContract("address to contract with
NFTs");

// Получение всех NFT контракта

NFT[] allNFTs = await nftContract.ERC1155.GetAll();

// Получение всех NFT пользователя
```

Окончание листинга 3

```
string address = await sdk.wallet.GetAddress();

NFT[] ownedNFTs = await nftContract.ERC1155.GetOwned(address);
```

Как можно заметить, NFT возвращаются массивом, для получения конкретного экземпляра нужно искать его по идентификатору, который, соответственно, должен быть одним из атрибутов метаданных.

Заключение

В ходе работы был проведен сравнительный анализ Web3-платформ и выбрана наиболее подходящая для интеграции с Unity. Были реализованы ключевые функции, требующие работы с блокчейном: вызов окна аутентификации пользователя, чеканка и получение NFT.

Стоит подчеркнуть, что рассмотренный Thirdweb SDK на самом деле имеет гораздо больше возможностей, чем было продемонстрировано, и с его помощью можно решать гораздо менее тривиальные задачи вплоть до совершенно произвольных с многоступенчатой логикой. Однако рассмотренных функций вполне достаточно для MVP версии разрабатываемого продукта.

Результаты данной статьи будут полезны разработчикам NFT игр не только на Unity, но и на прочих аналогах, таких как Unreal Engine 5, так как выбор Web3-платформы проводился изолированно от конкретной среды разработки игр, а API SDK в целом одинаков.

Список литературы

1. Thirdweb docs [Электронный ресурс]. – Режим доступа: https://portal.thirdweb.com/

2. Moralis for Developers - Enterprise-Grade Web3 APIs [Электронный ресурс]. – Режим доступа: https://developers.moralis.com/

3. Nethereum [Электронный ресурс]. – Режим доступа: https://nethereum.com/

4. Introducing ChainSafe Gaming | Chainsafe Gaming Docs [Электронный ресурс]. – Режим доступа: https://docs.gaming.chainsafe.io/

5. Web3 Unity SDK powered by thirdweb. [Электронный ресурс]. – Режим доступа: https://github.com/thirdweb-dev/unity

Разработка программного обеспечения анализа устойчивости оптимального решения задачи нахождения критического пути с нечеткими параметрами

И. В. Горбачева

Студент магистр

Н. А. Алейникова

Доцент

Введение

В современном мире одной из важнейших составляющих успешного управления проектами является точное определение продолжительности их выполнения. Этот процесс требует учета множества факторов, начиная от количества работ, их продолжительности и заканчивая доступными ресурсами. Правильное планирование сроков позволяет избежать задержек, перерасхода бюджета и гарантирует выполнение всех этапов проекта в установленные сроки.

Однако классические методы сетевого планирования, основанные на детерминированных данных, не учитывают неопределенность, присущую многим проектам. Реальная продолжительность выполнения отдельных работ зачастую подвержена колебаниям, что может существенно повлиять на сроки завершения всего проекта.

Недостаток, связанный с учетом неопределенностей, можно преодолеть с помощью теории нечетких множеств, позволяющей представлять временные параметры задач в виде нечетких треугольных чисел. В этой работе рассматривается задача поиска критического пути как задача линейного программирования, где продолжительности операций представлены в форме нечетких W-чисел. В работах [1-2] предложена W-алгебра для решения задач с нечеткими параметрами.

Целью этой работы является разработка алгоритма анализа устойчивости решения задачи поиска критического пути в условиях неопределённости и реализация его в виде программного обеспечения. Под устойчивостью решения будем понимать сохранение состава работ, входящих в критический путь, при изменении α-уровня нечеткого W-числа. Это означает, что при различных значениях параметра α,

© Горбачева И. В., Алейникова Н. А., 2025

который отвечает за степень принадлежности временных оценок, критический путь должен оставаться неизменным.

1. Постановка задачи поиска критического пути как задачи линейного программирования в условиях неопределенности

Одним из главных компонентов эффективного управления проектами является понятие критического пути. Критический путь представляет собой последовательность взаимосвязанных работ или операций в проекте, полная продолжительность выполнения которых определяет минимальное время выполнения всего проекта. Получается, что любые задержки в работах, определяющие критический путь, неизбежно приводят к увеличению общей продолжительности выполнения всего проекта.

Для формализации задачи поиска критического пути в виде задачи линейного программирования рассмотрим сетевую модель, которую представим в виде сетевого графа, который обозначим как $G = (n, P)$, где n – множество событий (вершин), P – множество работ (дуг). В таком графе для описания каждой работы будем использовать пару событий i и j, связанных работой с длительностью t_{ij}. При формализации задачи будем учитывать тот факт, что время свершения события j должно быть не меньше времени свершения события i плюс длительность работы t_{ij}. Так как критический путь является минимальным временем выполнения всего проекта, то для этого необходимо минимизировать разницу между временем свершения последнего события y_n и временем первого события y_1. Тогда математическую запись задачи поиска критического пути можно представить следующим образом:

$$Z = y_n - y_1 \to \min, \tag{1}$$

$$y_j - y_i \geq t_{ij}, \forall (i, j) \in P, \tag{2}$$

где y_i – моменты начала операции (i, j).

Но на стадии планирования проекта зачастую бывает затруднительно точно указать сроки выполнения всех работ. И тогда в таких случаях удобно использовать средства нечеткой логики, а именно нечеткие треугольные числа [2-3]. Эти числа позволяют учесть неопределенность и вариативность оценок, предоставляя возможность представить время выполнения работ в виде диапазона возможных значений. Тогда время выполнения каждой работы можно записать в виде нечеткого треугольного числа следующим образом: $\tilde{t}_{ij} = (t_{ij}^L, t_{ij}^m, t_{ij}^R)$,

где t_{ij}^m – мода (наиболее возможная продолжительность операции), t_{ij}^L, t_{ij}^R – наименьшая и наибольшая наименее возможная продолжительность работ соответственно [4]. Но при использовании нечеткой логики возникают сложности, связанные с большими вычислениями, искажение естественных свойств и отношений классических моделей. В работе [5] для устранения таких недостатков предлагается использовать W-алгебру и W-числа.

W-число – это вектор, который состоит из двух компонентов $(v^L(\alpha); v^R(\alpha))$, где $v^L(\alpha) \in X$, $v^R(\alpha) \in X$ и выполняется условие равенства при α=1: $v^L(1) = v^R(1)$, α – вещественный параметр, $\alpha \in [0;1]$.

Тогда, временные параметры задач также могут быть записаны в виде W-числа:

$$\tilde{t}_{ij} = (t_{ij}^L(\alpha), t_{ij}^R(\alpha)) = (t_{ij}^L + (t_{ij}^m - t_{ij}^L)\alpha, t_{ij}^R + (t_{ij}^m - t_{ij}^R)\alpha), \quad (3)$$

В работе [6] написано, что решение задач сетевого планирования в нечеткой постановке с использованием W-алгебры сводится к решению трех простых задач в четкой постановке. Тогда задача (1)-(2) может быть записана с использованием нечетких треугольных чисел:

$$\tilde{Z} = \tilde{y}_n - \tilde{y}_1 \to \min, \quad (4)$$

$$\tilde{y}_j - \tilde{y}_i \geq \tilde{t}_{ij}, \forall (i, j) \in P. \quad (5)$$

где $\tilde{t}_{pi}$ – нечеткие моменты начала операций.

В рассматриваемой задаче нахождения критические пути времена работ записаны с помощью W-чисел, и зависят от параметра α. И также иногда при решении задач поиска критического пути в нечеткой постановке с использованием W-чисел может возникнуть существенный недостаток: не на всех α-уровнях оптимальное решение задачи может оставаться одинаковым (быть устойчивым). В работе [7] под устойчивостью предполагается сохранение оптимального решения на всех α-уровнях нечетких чисел от 0 до 1. В контексте задачи поиска критического пути устойчивость будет обозначать сохранение состава работ, формирующих критический путь. Если на всех уровнях α от 0 до 1 критический путь составляют одни и те же работы, то такое решение считается устойчивым, иначе неустойчивым. Так как в данной работе W-числа представляют собой интервальные оценки временных параметров событий и работ, и при изменении уровня α эти интервалы могут сужаться или расширяться, что непосредственно влияет на временные

ограничения и, соответственно, на структуру критического пути. В результате, хотя на одном уровне значимости критический путь может включать определенный набор работ, при переходе к другому уровню этот путь может измениться, в него могут начать входить другие работы.

2. Использование аппарата двойственности для анализа устойчивости

Для исследования устойчивости решения задачи поиска критического пути в условиях неопределенности в работе будем использовать аппарат двойственности, который позволяет связать исходную задачу с двойственной к ней. Аппарат двойственности дает возможность проанализировать чувствительность оптимального решения к изменению параметров модели [8]. В [8] представлены правила, с помощью которых можно перейти от прямой задачи к двойственной к ней.

Тогда задачу (4)-(5) назовем прямой и с помощью этих правил перейдем к двойственной к ней задаче и составим ее математическую модель. В нечеткой постановке она запишется следующим образом:

$$\tilde{F} = \sum_{(i,j)\in P} \tilde{t}_{ij}\tilde{x}_{ij} \to \max, \tag{6}$$

$$\begin{cases} \sum\limits_{\substack{i \\ (i,j)\in P}} \tilde{x}_{ij} - \sum\limits_{\substack{k \\ (j,k)\in P}} \tilde{x}_{jk} = 0, \\ \sum\limits_{\substack{j \\ (1,j)\in P}} \tilde{x}_{1j} = 1, \quad \sum\limits_{\substack{i \\ (i,n)\in P}} \tilde{x}_{in} = 1. \end{cases} \tag{7}$$

где $x_{ij} \in 0,1$ – бинарная величина потока, который проходит по дуге (i, j), если работа входит в критический путь, то $\tilde{x}_{ij} = 1$, иначе $\tilde{x}_{ij} = 0$.

Теоремы двойственности в линейном программировании помогают установить взаимосвязи между прямыми и двойственными задачами. Каждая из этих теорем имеет свой экономический смысл и помогает провести анализ устойчивости оптимального решения.

Первая теорема двойственности: если у прямой задачи существует оптимальное решение, то у двойственной задачи также существует оптимальное решение, и значения их целевых функций равны [8]: $F_{\min} = Z_{\max}$ или $F(X^*) = Z(Y^*)$, где X^*, Y^* – оптимальные решения взаимодвойственных задач.

Также, соотношения в прямой и двойственной задачи имеют компоненты оптимального решения. Если переменная исходной задачи является положительной (ненулевой) оптимального решения, то

соответствующая ей в двойственной задаче будет равна нулю и наоборот. В контексте задачи поиска критического пути это означает, что если некоторая работа находится на критическом пути (активное ограничение в прямой задаче), то соответствующая двойственная переменная этой работы будет ненулевой. Это позволяет идентифицировать работы, влияющие на продолжительность выполнения всего проекта.

Из этого соотношения следует соотношение между основными переменными одной из задач и неосновными переменными другой задачи.

Вторая теорема двойственности утверждает, что компоненты оптимального решения двойственной задачи, то есть значения двойственных переменных, соответствуют коэффициентам исходной задачи, выраженными через неосновные переменные [8].

Из третьей теоремы можно понять, что величина объективно обусловленной оценки показывает, на сколько изменится оптимальное значение целевой функции прямой задачи при изменении свободного члена в одном из ограничений на единицу [8].

На основе аппарата двойственности можно разработать алгоритм анализа устойчивости решения задачи поиска критического пути, с помощью которого можно провести исследование чувствительности решения в зависимости от параметра α нечеткого W-числа:

1. Представим параметры прямой (3)-(4) и двойственной (6)-(7) задач в виде W-чисел, тогда мы получаем задачу параметрического программирования, где параметром является α.
2. Найдем оптимальное решение прямой задачи (3)-(4) при $\alpha = 1$.
3. Определим основные и неосновные переменные оптимального решения двойственной задачи (6)-(7), пользуясь второй теоремой двойственности.
4. Рассмотрим целевую функцию двойственной задачи с коэффициентами, представленными левыми компонентами W-чисел.
5. Выразим основные переменные ограничений двойственной задачи (7) через неосновные.
6. Выразим целевую функцию двойственной задачи (6) через неосновные переменные оптимального решения.
7. Определим из системы неравенств, при каких значениях α оптимальное решение двойственной задачи на максимум не изменится. Для этого достаточно, чтобы все коэффициенты при неосновных переменных оптимального решения оставались неположительными.

8. Аналогично проходим шаги 4-7 для случая, когда коэффициенты целевой функции представлены правыми компонентами W-чисел.

9. Если диапазон изменения α, при котором оптимальное решение двойственной задачи на максимум не изменяется, включает отрезок [0; 1], то делаем вывод об устойчивости решения. Если это не выполняется, то определяем значения α, при которых устойчивость нарушается.

3. Программная реализация алгоритма

На основе ранее описанного алгоритма было разработано программное обеспечение «Анализ задачи сетевого планирования», в котором пользователь имеет возможность расчета продолжительности критического пути, просмотра составляющих его работ и получение результата анализа устойчивости полученного оптимального решения.

Приложение было разработано на объектно-ориентированном языке программирования Python. При разработке были использованы следующие библиотеки: CVXPY, SymPy, SciPy, Pandas, Tkinter.

На рис. 1 представлена Use-Case диаграмма для разработанного приложения «Анализ задачи сетевого планирования».

Основные акторы:

– Пользователь: лицо, использующее программное обеспечение для анализа задач сетевого планирования.

Варианты использования:

– Ввод данных: пользователь вводит исходные данные для анализа: информацию о количестве событий, работ, их длительности и зависимости.

– Определение критического пути: система производит расчет критического пути на основе введенных данных.

– Анализ устойчивости: система проводит анализ устойчивости полученного оптимального решения по ранее разработанному алгоритму.

– Дефаззификация результатов: система приводит полученные результаты в четкие числа для удобства восприятия.

– Вывод результатов: система отображает окончательные результаты расчетов: информацию о критическом пути и его устойчивости.

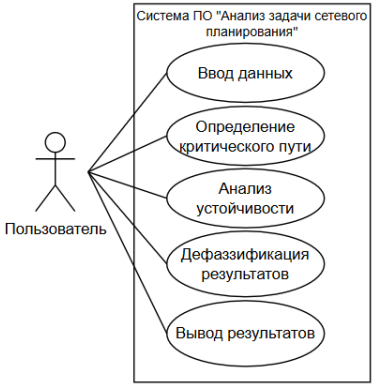


Рис. 1. Use-Case диаграмма ПО «Анализ задачи сетевого планирования»

На рис. 2 представлен пример окна вывода полученных результатов.

Результаты

```
Результат решения:
Минимальный время выполнения проекта (для минимальных времен): 56.0
Минимальный время выполнения проекта (для наиболее вероятных времен): 70.0
Минимальный время выполнения проекта (для максимальных времен): 88.0

Минимальный время выполнения проекта в виде нечеткого треугольного числа: (56.0, 70.0, 88.0)

Минимальный время выполнения проекта в виде W-числа: (56.0 + 14.0α; 88.0 - 18.0α)

Минимальный время выполнения проекта в дефаззифицированном виде: 71.33

Таблица результатов решения прямой задачи:
Операция   yL   ym   yR                    W-число  Дефаззифицированное
      y1  0.0  0.0  0.0      (0.0 + 0.0α; 0.0 - 0.0α)                 0.00
      y2  2.0  4.0  5.0      (2.0 + 2.0α; 5.0 - 1.0α)                 3.67
      y3  3.0  5.0  8.0      (3.0 + 2.0α; 8.0 - 3.0α)                 5.33
      y4  4.0  6.0 11.0     (4.0 + 2.0α; 11.0 - 5.0α)                 7.00
      y5  9.0 13.0 20.0     (9.0 + 4.0α; 20.0 - 7.0α)                14.00
      y6 18.0 25.0 34.0    (18.0 + 7.0α; 34.0 - 9.0α)                25.67
      y7 22.0 28.0 37.0    (22.0 + 6.0α; 37.0 - 9.0α)                29.00
      y8 28.0 36.0 47.0   (28.0 + 8.0α; 47.0 - 11.0α)                37.00
      y9 44.0 56.0 72.0  (44.0 + 12.0α; 72.0 - 16.0α)                57.33
     y10 34.0 43.0 56.0   (34.0 + 9.0α; 56.0 - 13.0α)                44.33
     y11 44.0 56.0 72.0  (44.0 + 12.0α; 72.0 - 16.0α)                57.33
     y12 56.0 70.0 88.0  (56.0 + 14.0α; 88.0 - 18.0α)                71.33

Критический путь (последовательность работ):
(1, 3) -> (3, 4) -> (4, 5) -> (5, 7) -> (7, 8) -> (8, 9) -> (9, 11) -> (11, 12)

Целевая функция может оставаться оптимальной в левой части нечеткого числа при: Interval(-2.0, 4.0)
Целевая функция может оставаться оптимальной в правой части нечеткого числа при: Interval(-oo, 1.5)
Слева решение устойчиво
Справа решение устойчиво
```

Рис. 2. Окно вывода результатов

Заключение

Данная работа была посвящена разработке алгоритма анализа устойчивости оптимального решения задачи поиска критического пути в

условиях статистической неопределенности и соответствующего ему программному обеспечению на языке программирования Python. Разработанный алгоритм и программное обеспечение позволяет оценивать интервалы, на которых критический путь остается неизменным и соответственно сам результат, устойчиво оптимальное решение или нет.

Список литературы

1. Шевляков, А. О. Нечеткие W-числа и W-алгебра для решения задач с нечеткими параметрами / А. О. Шевляков, М. Г. Матвеев // Актуальные проблемы прикладной математики, информатики и механики : сб. тр. Международной научно-технической конференции (Воронеж, 18-20 декабря 2017 г.). – Воронеж, 2017. – С. 960-967.
2. Матвеев, М. Г. Анализ и решение задач выбора с параметрической нечеткостью / М. Г. Матвеев // Вестник Южно-Уральского государственного университета. Сер.: Математическое моделирование и программирование. – 2015. – Т. 8, № 4. – С. 14-29.
3. Зацаринный, А. А. Моделирование процессов сетевого планирования портфеля проектов с неоднородными ресурсами в условиях нечеткой информации / А. А. Зацаринный, В. В. Коротков, М. Г. Матвеев // Информатика и ее применения. – 2019. – Т. 13, № 2. – С. 92-99.
4. Павлов, А. Н. Принятие решений в условиях нечеткой информации : учеб. пособие / А. Н. Павлов, Б. В. Соколов. – ГУАП – СПб : 2006. – 72 с. – ISBN 5-8088-0162-1
5. Шевляков, А. О. Сравнение различных нечетких арифметик / А. О. Шевляков, М. Г. Матвеев // Искусственный интеллект и принятие решений. – 2017. – No4. – С. 60–68.
6. Матвеев, М. Г. Математическое моделирование задачи сетевого планирования с помощью нечеткой математики / М. Г. Матвеев, Н. А. Алейникова // Вестник ВГУ, серия: системный анализ и информационных технологии. – 2018. – № 3. – С. 155–162
7. Воронцов, Я. А. Устойчивость решения в задаче о критическом пути с нечёткими параметрами / Я. А. Воронцов, М. Г. Матвеев // Вестник Воронежского государственного технического университета. – 2014. – Т. 10, № 6. – С. 40–43.
8. Кремер, Н. Ш. Исследование операций в экономике : учебник для вузов / Н. Ш. Кремер. – 2-е изд., перераб. и доп. – Москва : Юрайт, 2011. – 430 с.

Исследование методов обработки естественного языка для автоматического резюмирования интервью

Н. С. Грошев

Студент магистр

М. Е. Залыгаева

Старший преподаватель

Введение

В последние годы стремительный рост объёма текстовой информации в сети ставит перед исследователями и пользователями задачу быстрого и качественного извлечения главного из больших массивов данных [1]. Особенно остро эта проблема проявляется при работе с англоязычными текстами: необходимость оперативно оценивать их содержимое порождает интерес к автоматическим методам резюмирования, способным упростить понимание и сэкономить время. В момент разработки проекта доступ к мощным генеративным моделям, таким как GPT, был ограничен, а отечественных аналогов — ещё не существовало, что сделало исследование альтернативных подходов особенно актуальным.

Цель настоящей работы — исследование современных методов автоматического резюмирования текстов (NLP) и создание удобного веб-сервиса для их применения. Для достижения этой цели были поставлены следующие задачи:

1. Изучение существующих методов NLP
2. Реализация выбранных моделей для генерации резюме
3. Проведение сравнительного анализа эффективности моделей
4. Создание веб-сервиса
5. Развертывание моделей на сервере

Таким образом, сочетание теоретического обзора, практической реализации и эмпирической оценки позволит не только выявить сильные и слабые стороны различных подходов к резюмированию, но и предложить готовое решение для быстрой ориентации в большом массиве текстов.

© Грошев Н. С., Залыгаева М. Е., 2025

1. Датасет

Для исследования был выбран датасет с диалогами на английском языке. Причина выбора именно диалогов заключается в их сложности: диалоги включают разговорные сокращения, неформальные выражения и специфическую лексику, что представляет дополнительный вызов как для понимания человеком, не являющимся носителем, так и для работы нейронных сетей. Датасет насчитывает ~ 16 000 диалогов и столько же эталонных резюме.

2. Алгоритм TextRank

Для решения поставленной задачи были выбраны три разных подхода: алгоритм TextRank, рекуррентная нейронная сеть LSTM и трансформер BART [2, 4].

Алгоритм TextRank помогает автоматически выделять главные предложения или ключевые слова в тексте путём разбиения его на части и построения графа: сначала текст делится на отдельные элементы (предложения или слова), которые становятся вершинами, а связи между ними отражаются ребрами. Значимость каждой вершины вычисляется по формуле (1): чем сильнее она связана с другими важными вершинами, тем выше её рейтинг. Сходство вершин оценивается по формуле (2), учитывающей число общих слов и нормирующей его через логарифмы длин соответствующих частей текста. В конце из элементов с наивысшими рейтингами формируется итоговое резюме.

$$S(V_i) = (1-d) + d \cdot \sum_{V_j \in \ln(V_i)} \frac{S(V_j)}{|\mathrm{Out}(V_j)|}, \tag{1}$$

$$\mathrm{Similarity}(S_i, S_j) = \frac{|\{w_k \mid w_k \in S_i \,\&\, w_k \in S_j\}|}{\log(|S_i|) + \log(|S_j|)}, \tag{2}$$

где $S(V_i)$ – значимость вершины V_i, d – коэффициент затухания(обычно 0.85), $\ln(V_i)$ – множество вершин, указывающих на V_i, $Out(V_j)$ – количество рёбер, исходящих из вершины V_j; S_i и S_j – два предложения, представленные множествами слов, w_k – слово, принадлежащее как S_i, так и S_i, $|S_i|$ и $|S_j|$ – количество слов в предложениях S_i и S_j, $|\{w_k \mid w_k \in S_i \,\&\, w_k \in S_j\}|$ – количество общих слов между предложениями S_i и S_j.

3. Модель LSTM

LSTM (Long Short-Term Memory) – это тип рекуррентной нейронной сети, способной учитывать долгосрочные зависимости

в тексте за счет специальных механизмов памяти и управления информацией.

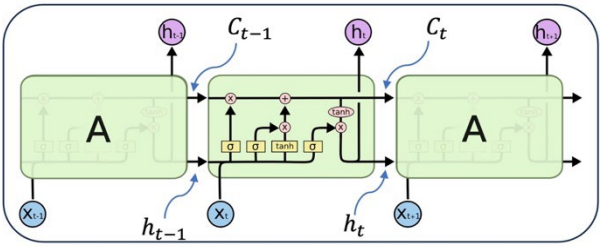


Рис. 1. Повторяющийся модуль LSTM. В LSTM используются четыре компонента: вентиль забывания информации, вентиль обновления (что запомнить), обновление состояния памяти и выходной вентиль (какую информацию передать на выход сети).

Основной элемент LSTM (рис. 1) — ячейка памяти C_t, способная сохранять нужную информацию долгое время. Управление данными происходит через три вентиля: забывания f_t, ввода i_t и вывода информации.

Вентили LSTM

Вентили – это механизмы, которые избирательно пропускают информацию. Они состоят из слоя нейронной сети с сигмоидальной функцией активации и покомпонентной операции умножения [3].

Первый вентиль – вентиль забывания (рис. 3). Он выбирает, какую информацию модель должна забыть. Вектор выходных значений показан на формуле (3):

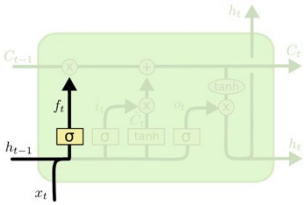


Рис. 2. Вентиль забывания

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{3}$$

где f_t – вектор выходных значений, σ – сигмоидальная функция активации, W_f – матрица весов, $[h_{t-1}, x_t]$ – конкатенация выходного вектора с предыдущего шага h_{t-1} и текущего входного вектора x_t, b_f – вектор смещений (bias).

Следующий этап – определение новой информации, которую нужно сохранить в состоянии ячейки. Этот процесс состоит из двух частей:

Входной вентиль (input gate) на основе сигмоидальной функции активации решает, какие значения будут обновлены.

Слой с функцией активации tanh создает вектор новых кандидатов $\tilde{C}_t$, которые могут быть добавлены в состояние ячейки. Математическая запись этого этапа показана на формулах 4 и 5.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{4}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \tag{5}$$

где $\tilde{C}_t$ – вектор новых кандидатов для состояния ячейки на шаге t, значения которого находятся в диапазоне [-1, 1], tanh – гиперболический тангенс, который сжимает значения в диапазон [-1, 1], W_c – матрица весов, связанная с вычислением новых кандидатов.

Далее происходит обновление старого состояния ячейки C_{t-1} до нового состояния C_t. Ранее мы определили, что нужно сохранить и что удалить, теперь надо это объединить. На формуле (6) показано, как происходит обновление состояния ячейки целиком.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t, \tag{6}$$

где C_t – новое состояние ячейки, f_t - вектор значений вентилей забывания, определяющий, что нужно удалить и что сохранить, i_t – вектор значений входных вентилей, определяющий, какие части новых данных добавить в состояние ячейки, $\tilde{C}_t$ – вектор новых кандидатов для добавления в состояние ячейки.

На последнем этапе необходимо определить, что будет выведено из сети. Этот вывод основывается на состоянии ячейки C_t, но является его отфильтрованной версией. Тут также два этапа: сначала слой сигмоидальной функции активации решает, какие части состояния ячейки будут выведены, это показано на формуле (7).

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{7}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{8}$$

Затем состояние ячейки C_t проходит через функцию активации tanh. Это состояние умножается на o_t, чтобы оставить только те части, которые были выбраны для вывода (формула (8)).

4. BART (Bidirectional and Auto-Regressive Transformer)

Теперь перейдем к BART. BART – это нейронная сеть, используемая для глубокого анализа и генерации текстов [5]. Модель состоит из двух частей: двунаправленного кодера, глубоко анализирующего входной текст, и авторегрессионного декодера, последовательно генерирующего текст (в нашем случае, резюме). В основе обеих частей лежит механизм внимания (Attention), позволяющий сети «фокусироваться» на значимых частях текста, учитывая важность и контекст слов. Пример архитектуры модели показан на рис. 3.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V, \tag{9}$$

Механизм внимания вычисляет веса, определяющие важность элементов текста, по формуле 9. В кодере используется двунаправленное внимание (самовнимание), помогающее каждому слову учитывать контекст с двух сторон, а в декодере применяется маскированное самовнимание(чтобы при генерации очередного токена модель видела только уже сгенерированные ранее слова) и перекрестное внимание (чтобы декодер мог заглядывать в выходы кодера и извлекать из них релевантную информацию). BART сначала предобучают на больших текстовых корпусах, а затем дообучают на конкретной задаче. Благодаря комбинации механизмов внимания и двунаправленной обработки, BART показывает отличные результаты в генерации качественных и содержательных резюме.

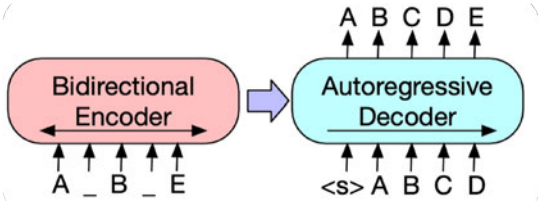


Рис. 3. Архитектура модели BART

5. Результаты

Для оценки моделей я использовал оценку ROUGE и BERTScore. ROUGE показывает, какая доля слов из эталонного резюме совпадает со словами из сгенерированного резюме, то есть, по сути, измеряет полноту (recall). Существенный недостаток ROUGE заключается в том, что эта метрика не учитывает перефразированные или синонимичные выражения и понимает лишь прямые совпадения слов.

BertScore, в свою очередь, учитывает не только прямые совпадения, но и смысловую близость слов и предложений. Метрика рассчитывает косинусное сходство контекстуальных векторов слов из эталонного и сгенерированного текста, позволяя более точно оценить качество абстрактивного резюмирования.

Таблица

Результаты работы моделей

	Rouge1	Rouge2	RougeL	PBERT	RBERT	FBERT
TextRank	0.25	0.07	0.20	0.12	0.33	0.21
LSTM	0.84	0.71	0.83	0.77	0.74	0.75
BART	0.53	0.28	0.47	0.61	0.89	0.72

Как мы можем заметить по результатам из таблицы, TextRank с задачей справился плохо. Он не умеет сокращать или переформулировать текст, а просто извлекает наиболее значимые, по его мнению, предложения, а для данной задачи этого недостаточно. Модель LSTM демонстрирует отличные результаты, так как сгенерированные тексты очень близки к эталонным. А модель BART генерирует резюме своими словами, что снижает её показатель ROUGE (так как меньше прямых совпадений слов с оригиналом). Но при этом метрика BertScore показывает более объективные результаты, поскольку учитывает смысловую близость слов, а не прямое совпадение.

Заключение

В ходе проведённого исследования были рассмотрены и реализованы три подхода к задаче автоматического резюмирования текстов: алгоритм TextRank, рекуррентная нейронная сеть на базе LSTM и современная абстрактивная модель BART. Для каждого из методов выполнена математическая формализация, реализованы прототипы на языке Python и проведена их оценка с помощью комбинации метрик ROUGE и BertScore.

Экспериментальные результаты показали, что классический алгоритм TextRank обеспечивает лишь базовое извлечение наиболее «весомых» предложений, но не справляется с перефразированием и генерацией нового текста. Модель LSTM продемонстрировала высокие показатели ROUGE благодаря близости к эталонным референсам, однако уступает BART в плане гибкости формулировок. BART, в свою очередь, генерирует обобщённые и стилистически разнообразные резюме, что отражается более высоким BertScore, но более низкими значениями ROUGE из-за меньшего лексического совпадения.

Список литературы

1. Orasan, C. Automatic Summarisation : 25 Years On / C. Orasan // Natural Language Engineering. – 2019. – Vol. 25. – No 6. – PP. 735–751.

2. Гольдберг, Й. Нейросетевые методы в обработке естественного языка / Й. Гольдберг ; пер. с англ. А. А. Слинкина. – М. : ДМК Пресс, 2019. – 282 с. : ил. – ISBN 978-5-97060-754-1.

3. Goyal, P. Deep Learning for Natural Language Processing : Creating Neural Networks with Python / P. Goyal, S. Pandey, K. Jain. – New York : Apress, 2018. – 294 p. – ISBN 978-1-4842-3684-0.

4. Хобсон, Л. Обработка естественного языка в действии / Л. Хобсон, Х. Хапке, Х. Коул. – СПб. : Питер, 2020. – 576 с. : ил. – (Серия «Для профессионалов»). – ISBN 978-5-4461-1371-2.

5. Jurafsky, D. Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition / D. Jurafsky, J. H. Martin. – 2nd ed. – Upper Saddle River, NJ : Prentice Hall, 2008. – 1024 p. – ISBN 978-0-13-187321-6.

Информационные сервисы на электронной торговой площадке

П. В. Дрюкова

Студент магистр

М. Г. Матвеев

Профессор

Введение

С развитием современных технологий многие направления в торговле не просто изменились, а получили совершенно новые формы. В последние годы быстрыми темпами растет популярность электронных торговых площадок (маркетплейс), которых на рынке становится все больше. Они формируют информационное пространство для эффективного взаимодействия продавцов и покупателей в рамках конкретной отрасли.

Маркетплейс призван помочь покупателю найти необходимые ему товары и услуги, выбирая лучшее по таким параметрам как: производитель, цена, качество и другие. А также, помочь продавцам с выбором выставляемого на продажу товара и его параметров.

Продавец, формируя свое предложений в базе данных маркетплейса, должен ориентироваться на покупательский спрос. Для этого продавцу необходима рекомендательная система, способная оценивать уровень соответствия покупательского спроса и предложения продавца. Информационную основу такой системы обеспечивает маркетплейс, заинтересованный в привлечении продавцов адекватных сформировавшемуся спросу. Чем больше будет таких продавцов, тем больше заработает маркетплейс, финансируемый этими продавцами [1].

В данной работе предлагаются информационные технологии создания рекомендательного сервиса для продавца, с учетом спроса покупателей. Рекомендательная система поможет продавцу подобрать такие параметры товара, которые обеспечивают наибольшую выгоду. В основу рекомендательной системы положено векторное представление характеристик спроса и предложения с последующим сравнительным анализом (матчингом) этих векторов и выбором наилучшего варианта сделки.

© Дрюкова П. В., Матвеев М. Г., 2025

1. Формализованное представление информации о товарах

Информационные технологии являются совокупностью средств и методов сбора, обработки и передачи данных (первичной информации) для получения информации нового качества о состоянии объекта, процесса или явления (информационного продукта).

Информация, необходимая для рекомендательного сервиса формируется за счет парсинга из базы данных маркетплейса, доступ к которой обеспечивается с помощью API.

Для того чтобы формировать информационные технологии, необходимо иметь единую форму всех информационных процессов, которые происходят на маркетплейсе, например, процесса обработки информации о товарах. Чтобы охарактеризовать любой товар, нужен некоторый набор параметров, который задается товарными классификаторами маркетплейса [2]. Товарный классификатор – это структура ассортимента, которая распределена на схожие по определенным критериям группы товаров. Товарные классификаторы формируются на основании иерархического, фасетного и/или смешанного методов.

В работе предлагается построить алгоритм сервиса на основе соответствия характеристик товара, выставляемого продавцом, покупательскому спросу. Для такого соответствия удобно использовать товарные классификаторы фасетного типа. Далее будем использовать вектор характеристик товара вместо аналогичного ему фасетного классификатора.

Вектор товарных характеристик имеет следующий вид:

$$q = (q^1(x); q^2(x); ...; q^n(x)), \quad (1)$$

где q^n – имя n-го параметра, характеризующего товар из однородной группы; x – значение этого параметра.

Далее под однородными товарами будем понимать товары, имеющие одинаковое функциональное назначение, одинаковую структуру параметров, но различающиеся значениями этих параметров Такие классификаторы составляют основу нормативно-справочной информации маркетплейса.

Выделим среди товарных характеристик характеристики спроса, которые часто задаются не конкретными значениями, а некоторым диапазоном значений, например, желаемая цена. Следуя [3], будем отображать пожелания в виде нечетких параметров спроса:

$$\tilde{g}_{ij} = (\tilde{g}_{ij}^1; \tilde{g}_{ij}^2 ... \tilde{g}_{ij}^N), \quad (2)$$

где $\tilde{g}_{ij}^{1}$ - лингвистическая переменная, отображающая n-й параметр i-го товара j- го покупателя. Каждая лингвистическая переменная имеет кусочно-линейную функцию принадлежности $f_g(x) \in [0;1]$, носители которых $x_{\min} \le x \le x_{\max}$ отражают допустимые значения параметра, а значения функции – предпочтения покупателей. Для дискретных значений носителя функции предпочтений будут иметь дискретный вид.

Пусть каждый продавец формализует свои возможности и желания в отношении однородного товаров виде векторов с именами лингвистических характеристик, таких же, как и у покупателя:

$$\tilde{q}_{ij} = (\tilde{q}_{ij}^{1}; \tilde{q}_{ij}^{2} ... \tilde{q}_{ij}^{N}), \tag{3}$$

где $\tilde{q}_{ij}^{n}$ - лингвистическая переменная, отображающая n-й параметр i-го товара j- го продавца. Желания и возможности продавца также отображаются функциями принадлежности $f_q(x) \in [0;1]$. Функции принадлежности $f_g(x)$ и $f_q(x)$ имеют следующий вид:

$$f(x) = \begin{cases} \dfrac{x-a}{b-a}, a \le x \le b \\ \dfrac{c-x}{c-b}, b \le x \le c \\ 0, x > d, x < a. \end{cases} \tag{4}$$

Вычислять соответствие товара индивидуальному спросу покупателей практически невозможная задача. Поэтому предлагается ориентироваться на обобщенный спрос. Правомерность такого похода показана в работе [3].

Обобщенный спрос имеет ту же структура, что и индивидуальный, только каждая компонента вектора спроса представляет собой свертку индивидуальных компонент:

$$\tilde{g} = (\tilde{g}^{1}; \tilde{g}^{2} ... \tilde{g}^{N}), \tilde{g}^{n} = \bigcup_{j=1}^{N} \tilde{g}_{j}^{n}. \tag{5}$$

Функция принадлежности обобщенного спроса по компоненте n-вектора спроса будет иметь вид:

$$f_g^n(x) = \sum_{j=1}^{J} f_j^n(x) w_j^n, \tag{6}$$

где w_j^n - весовые коэффициенты.

Для оценки соответствия векторов (3) и (5) необходимо ввести числовую меру такого соответствия.

2. Разработка методики построения меры соответствия векторов спроса и предложения

Оценку степени соответствия удобно проводить в два этапа. Сначала найти парные локальные соответствия компонент векторов (3) и (5), а затем агрегировать полученные локальные соответствия в единую числовую меру соответствия потребностей продавца и покупателя по рассматриваемому типу товара.

Введем следующую операцию [4], которая позволяет получить достаточно информативную локальную меру соответствия компонент $\tilde{q}_{ij}^n$ и $\tilde{g}_{ij}^n$. Эта локальная характеристика вычисляется как пересечение соответствующих нечетких множеств и служит локальной мерой соответствия однородных пар компонент:

$$f^n(x) = \min\{f_q^n(x); f_g^n(x)\}. \quad (7)$$

В частном случае, когда, например, продавец выставляет однозначное (четкое) значение x компоненты, выражение (7) преобразуется к виду (8):

$$f^n(x) = \min\{1; f_g^n(x)\}. \quad (8)$$

Полученные локальные меры соответствия компонент векторов (3) и (5) образуют векторы соответствия условий обобщенного спроса и предложения:

$$f_{ij} = (f_{ij}^1; ...; f_{ij}^n; ...; f_{ij}^N). \quad (9)$$

Иллюстрация пересечения обобщенного спроса и предложения показана на рисунке 1, где результат пересечение выделен жирной линией. Следует обратить внимание на то, что определение максимального локального соответствия не представляет проблем.

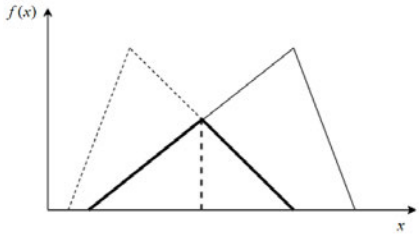


Рис. 1. Пересечение графиков функции принадлежности продавца и покупателя

Локальные соответствия по всем компонентам на втором этапе должны быть агрегированы в совокупную характеристику соответствия потребностей i - го продавца и k - го покупателя.

Для построения процедуры агрегирования целесообразно носители функций принадлежности привести к единой шкале для всех параметров от 1 до N. Для этого используем следующее простое преобразование:

$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \in [0;1], \tag{10}$$

где $x_{\min}; x_{\max}$ - соответственно нижняя и верхняя граница объединения носителей функций принадлежности параметров продавца и покупателя.

В дальнейшем верхний индекс (*) у преобразованной переменной будем опускать.

Оператором агрегирования обычно называют функцию, присваивающую некоторому кортежу действительных чисел из интервала [0;1] одно действительное число из этого интервала [5]:

$$agr : \bigcup_{n=1}^{N} [0;1]^n \to [0;1]. \tag{11}$$

Следуя методике, представленной в работе [5], предлагается в качестве оператора агрегирования использовать метод идеальной точки [6]. Решение задачи агрегирования сводится к максимизации критерия – квадрат расстояния в пространстве соответствий до идеальной точки:

$$\sum_{n=1}^{N} w_i (f_{\max}^n - f^n(x_i))^2 \to \max_x, \tag{12}$$

где w_i - весовые коэффициенты; $f_{\max}^n$ - идеальная точка, соответствующая максимальному значению функции $f^n(x)$.

Такая постановка задачи должна учитывать, что множество значений параметров дискретно и ограничено. Такое предположение вполне оправдано, у продавца для выбора оптимального предложения имеется ограниченный набор товаров с конкретными значениями параметров. Формально ограничения выглядят следующим образом:

$$x \in D, \tag{13}$$

где множество D включает конечный набор значений характеристических параметров товаров, которыми располагает продавец.

3. Численная апробация предложенного алгоритма

Рассмотрим рынок конкретного однородного товара с конкретными характеристическими параметрами. Пусть таким рынком будет рынок обуви.

В качестве конкретных характеристических параметров возьмем размер обуви, цену и качество. Шкалой цен являются рубли, шкалой размера обуви является общепринятый российский размер, а шкалой качества – шкала от 1 до 5.

Покупатель формулирует желаемые значения параметров либо как промежуток количественных значений, либо как совокупность качественных значений. Например, покупатель хочет приобрести обувь размера 42 или 43, в ценовом промежутке от 1000 до 3000 тысяч рублей, среднего или лучшего качества. Гораздо удобнее работать не с естественными шкалами, а с нормированными. Для этого нормируем описанные ранее шкалы, используя формулу (10). После применения правила нормировки мы получаем общую шкалу $[0;1]$.

Вектор характеристических параметров имеет следующий вид:

$$q_j = (q_j^1, q_j^2, q_j^3), \tag{14}$$

где компоненты интерпретируются как q^1 – цена, определяемая на нормированной шкале $[0;1]$; q^2 – размер, определяемый на нормированной шкале $[0;1]$; q^3 – качество, определяемое наименованием производителя, которое расположено на нормированной шкале $[0;1]$.

Предположим, что на рынке имеется четыре мелкооптовых покупателя $k = 4$, чьи объемы запрашиваемого товара соответственно равны: $v_1 = 10, v_2 = 4, v_3 = 6, v_4 = 5$.

Формализуем покупательский спрос, т.е. вектор лингвистических переменных, четырех покупателей в виде:

$$\tilde{g}_k = (\tilde{g}_k^1, \tilde{g}_k^2, \tilde{g}_k^3). \tag{15}$$

Аналогично спросу, продавец должен формализовать свои возможности по цене, размерам и качеству, т.е. вектор лингвистических переменных, в следующем виде:

$$\tilde{q} = (\tilde{q}^1, \tilde{q}^2, \tilde{q}^3). \tag{16}$$

Представим промежуток [0;1] с помощью регулярного разбиения на достаточно большое число малых промежутков. Пусть число малых промежутков будет 10.

В полученных узлах необходимо рассчитать с помощью формулы для нахождения значений обобщенного спроса покупателей по каждой из компонент:

$$f_g^n(x) = \sum_{k=1}^{k} f_{gk}^n(x) \cdot w_k, n = 1...N. \quad (17)$$

Для того чтобы определить характеристики однородного товара, который бы больше всего соответствовал спросу на рынке и предложению продавца, было решено использовать в качестве критерия выбора компромисс между максимумом вероятности сделки и максимумом ее выгодности. Для этого необходимо найти пересечение функции принадлежности компонент векторов обобщенного спроса и функции принадлежности компонент вектора возможностей продавца и выбрать максимальные точки пересечения этих функций принадлежности по формуле (12). Полученные максимальные точки пересечения представлены на рис. 2:

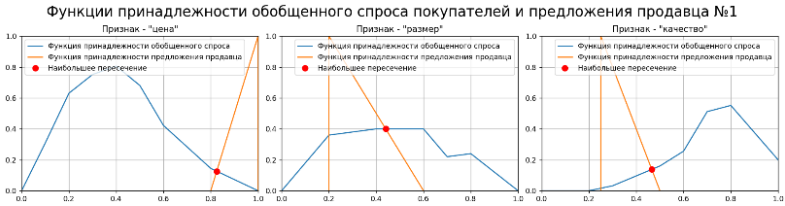


Рис. 2. Функции принадлежности обобщенного спроса покупателей и предложения продавца

Необходимо учитывать то, что в пересечения могут не попадать реальные значения признаков товаров, необходимые покупателю, поэтому будут подбираться наиболее приближенные к реальности значения. Для этого будет использовано Евклидово расстояние, рассчитанное для каждого товара по формуле:

$$d(x^{\max}, x^s) = \sqrt{(x_1^{\max} - x_1^s)^2 + (x_2^{\max} - x_2^s)^2 + (x_3^{\max} - x_3^s)^2}. \quad (18)$$

Далее полученные расстояния необходимо отсортировать по возрастанию, после чего можно составить таблицу рейтинга товаров. Товары, находящиеся в начале таблицы, будут больше всего соответствовать покупательскому спросу на маркетплейсе.

Заключение

Данная статья посвящена автоматизированию процесса формирования товарного предложения продавца на маркетплейсе, которое в большей степени соответствовало бы покупательскому спросу.

В процессе выполнения были решены следующие задачи:

– Проведена формализация спроса и предложения на основе нечетких множеств и лингвистических переменных.

– Реализован алгоритм для формирования товарного предложения продавца

– Полученный алгоритм в дальнейшем может быть использован как один из виджетов на сервисе поддержки продавца, в котором на вход будут поступать данные анализа пользовательского поведения, желания продавца и его товары, а на выходе продавец будет получать сформированное предложение в виде рейтинга товаров, которые будет иметь высокий спрос на электронной торговой площадке среди покупателей.

Список литературы

1. Шендрик, П. В. Теоретические особенности функционирования маркетплейсов / П. В. Шендрик // Международный научно-исследовательский журнал. – 2022. – № 10. – С. 74.

2. Сысоева, С. В. Управление ассортиментом в рознице. Категорийный менеджмент / С. В. Сысоева, Е. А. Бузукова – СПБ. : Питер, 2010. – 330 с.

3. Матвеев, М. Г. Информационные технологии формирования товарного предложения на электронной торговой площадке с технологией маркетплейс / М. Г. Матвеев // Экономика и математические методы. – 2021. – Т. 57. – № 1. – С. 105-112.

4. Матвеев, М. Г. Анализ взаимодействия в системе продавец-покупатель на основе числовой меры парного соответствия / М. Г. Матвеев // Вестник ВГУ. Серия «Системный анализ и информационные технологии». – 2019. – №3. – С. 94-103.

5. Леденева, Т. М. Обзор основных классов операторов порядкового взвешенного агрегирования / Т. М. Леденева, И. Н. Левкина // Вестник ВГУ. Серия «Системный анализ и информационные технологии». – 2022. – №1. – С. 5-31.

6. Машунин, Ю. К. Теория и методы принятия оптимальных решений по множеству критериев в инженерных системах / Ю. К. Машунин. – Русайнс, 2023. – 340 с.

Дообучение модели естественного языка BERT с использованием GPU

А. А. Зюзин

Студент магистр

А. В. Швырева

Старший преподаватель

Введение

Современные системы обработки естественного языка (NLP) всё чаще применяются для автоматизации задач, связанных с анализом, преобразованием и сокращением текстовой информации. Одним из наиболее эффективных инструментов в этой области являются трансформерные языковые модели, в частности BERT и её локализованные версии, такие как ruBERT. Однако, несмотря на высокое качество обобщённых моделей, их применение к специфическим задачам часто требует дополнительной адаптации к целевому корпусу данных.

Одной из таких задач является автоматическое извлечение ключевых предложений из длинных пользовательских текстов (extractive summarization) с целью их адаптации под ограничения платформ, таких как Telegram, ограничивающий длину сообщений 1024 символами. Перенос длинных постов из других источников, например, из ВКонтакте, становится нетривиальной задачей, требующей интеллектуальной обработки текста.

Цель данной работы — исследовать возможность дообучения предобученной языковой модели ruBERT на пользовательском корпусе данных с использованием ограниченных вычислительных ресурсов, в частности потребительского GPU. В процессе работы демонстрируется, как с помощью дообучения можно существенно повысить качество извлекаемых кратких версий текстов, сохраняя их смысловую целостность и логическую связность.

Кроме того, в статье рассматривается влияние среды выполнения на эффективность процесса дообучения. Были протестированы различные варианты — от облачной платформы Google Colab до локального оборудования с графическим процессором — с целью

© Зюзин А. А., Швырева А. В., 2025

оценки производительности, времени обучения и удобства настройки. Такой сравнительный анализ позволяет сделать выводы о применимости доступных решений в условиях ограниченных вычислительных ресурсов.

Работа охватывает следующие аспекты: подготовка и автоматическая разметка датасета, настройка среды дообучения, выбор архитектуры и параметров обучения, сравнительный анализ сред выполнения, а также оценка качества результатов до и после адаптации модели.

1. Обзор модели

В качестве базовой модели для решения задачи извлечения ключевых предложений из длинных русскоязычных текстов была выбрана предобученная трансформерная модель ruBERT-base, являющаяся адаптацией архитектуры BERT под русский язык и предоставляемая в рамках проекта ai-forever [1]. Модель основана на механизме многоголового бидирекционального внимания, что обеспечивает учёт контекста как слева, так и справа от обрабатываемого слова, тем самым улучшая интерпретацию сложных синтаксических конструкций. Архитектура ruBERT-base включает 12 трансформерных слоёв с размерностью скрытых представлений 768 и 12 головами внимания, при общем числе параметров порядка 180 миллионов. Такая конфигурация обеспечивает баланс между вычислительной эффективностью и качеством моделирования языка [2]. Применение ruBERT обусловлено её предварительным обучением на больших русскоязычных корпусах и полной совместимостью с современными библиотеками глубокого обучения, такими как Transformers и PyTorch, что делает её удобной для последующего дообучения на ограниченных пользовательских данных [3]. В рамках настоящего исследования модель была адаптирована к задаче экстрактивной суммаризации, формулируемой как бинарная классификация предложений относительно их релевантности сокращённой версии исходного текста.

2. Подготовка набора данных

Для проведения дообучения модели была сформирована специализированная выборка на основе текстов из открытой группы факультета компьютерных наук в социальной сети «ВКонтакте». В корпус вошли 200 пользовательских постов, для каждого из которых вручную или полуавтоматически была подготовлена краткая версия длиной не более 1024 символов, что соответствует ограничению на размер одного сообщения в мессенджере Telegram. Таким образом, были получены 200 пар текстов в формате «полный текст ↔ сокращённый

текст», что позволило сформировать основу для задачи экстрактивной суммаризации. Особенность данной выборки заключается в её тематической однородности и наличии реальных примеров пользовательской письменной речи, что позволяет более точно адаптировать модель к целевому сценарию применения. На рис. 1 приведены примеры исходных и обработанных данных.

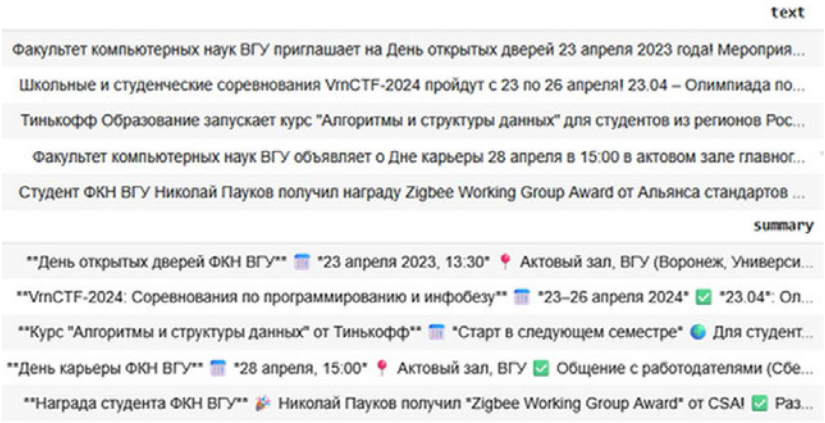


Рис. 1. Пример исходных и обработанных данных

3. Обработка и разметка данных

Для приведения корпуса в пригодный для обучения вид была реализована процедура автоматической разметки предложений на основе эвристического сопоставления с сокращённой версией текста. Полные тексты предварительно сегментировались на отдельные предложения с использованием библиотеки *NLTK* (Natural Language Toolkit), после чего каждое предложение сравнивалось с соответствующим сокращённым вариантом. Предложения, дословно встречающиеся в сокращённом тексте, маркировались меткой 1 (релевантное), остальные – меткой 0 (нерелевантное). Такой подход позволил без ручной аннотации автоматически разметить около 3000 предложений, что является достаточным объёмом для дообучения модели на задачу бинарной классификации. Использование эвристической разметки значительно ускоряет процесс подготовки данных и обеспечивает воспроизводимость методики [4].

4. Настройка дообучения на GPU и сравнение с удаленной средой

Дообучение модели ruBERT-base проводилось в различных средах исполнения с целью сравнения производительности и определения

наилучшего соотношения между временем обучения и доступными ресурсами. Были протестированы три конфигурации: Google Colaboratory с использованием CPU, Google Colaboratory с использованием облачного GPU и удалённое подключение Colab к локальному компьютеру с дискретным GPU. В первом случае (Colab + CPU) обучение модели на протяжении трёх эпох заняло около 30 минут, что делает данный вариант малопригодным даже для небольших корпусов. Второй сценарий (Colab + облачный GPU, например Tesla T4) существенно ускорил обучение: полный цикл из трёх эпох выполнялся за приблизительно 3 минуты. Наиболее эффективным оказался третий подход, при котором среда Google Colab использовалась в качестве интерфейса, а фактические вычисления происходили на локальной машине с видеокартой NVIDIA GeForce RTX 4070. В этом случае дообучение завершалось менее чем за 30 секунд. Для реализации такого подключения необходимо было настроить сервер Jupyter Notebook на локальной машине и обеспечить его совместимость с Colab через безопасное удалённое соединение. Настройка включала запуск Jupyter с дополнительными параметрами, разрешающими подключение из внешнего интерфейса, с помощью команды, представленной на рис. 2.

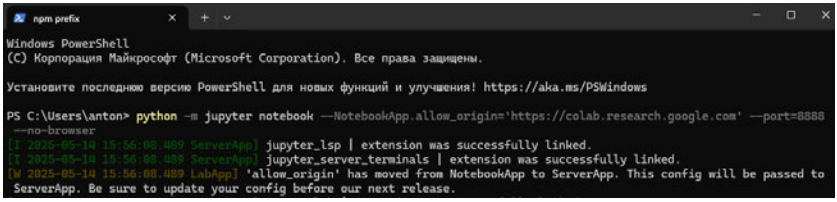


Рис. 2. Запуск среды для подключения

После запуска сервер становился доступным для удалённой сессии. На рисунке 3 представлен интерфейс Colab для подключения.

После запуска сервера Jupyter Notebook достаточно выбрать пункт "Подключиться к локальному средству выполнения", указать URL (например, http://localhost:8888) и ввести автоматически сгенерированный токен авторизации. Для корректной работы требуется предварительная установка всех необходимых библиотек (включая transformers, torch, datasets, nltk) в локальной среде Python.

Такой подход позволяет использовать удобный интерфейс Google Colab при сохранении всех преимуществ локального GPU — высокой скорости, отсутствия ограничений на сессии и полной гибкости в настройке окружения. Результаты тестирования представленные на рисунке 4 подтверждают, что подключение Colab к локальному

оборудованию обеспечивает наиболее эффективное выполнение обучения даже при использовании ресурсоёмких моделей трансформеров.

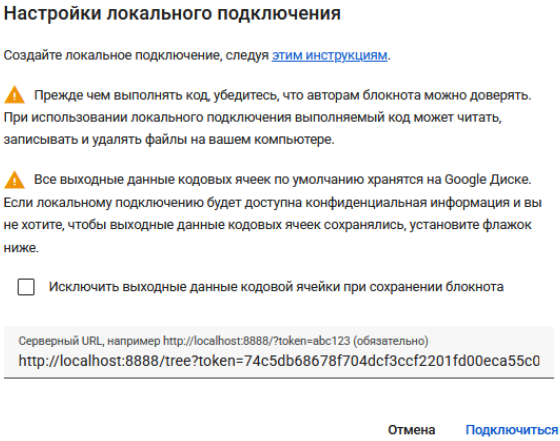


Рис. 3. Интерфейс подключения к локальной среде в Google Colab

Среда обучения	Тип устройства	Время дообучения (3 эпохи, ~3000 примеров)
Google Colab (бесплатный)	CPU	~30 минут
Google Colab (бесплатный)	GPU (Tesla T4)	~3 минуты
Локальная среда (RTX 4070)	GPU	~30 секунд

Рис. 4. Сравнение разных сред выполнения

5. Дообучение модели

Обучающая выборка была сформирована на основе 160 пар полных и сокращённых текстов постов из социальной сети ВКонтакте. Для валидации использовались оставшиеся 40 пар. Разметка данных осуществлялась автоматически путём сопоставления предложений полного текста с сокращённой версией. Предложения, вошедшие в краткий текст, помечались меткой 1, остальные — меткой 0.

К третьей эпохе наблюдалось быстрое схождение функции потерь на валидационном наборе, что свидетельствует о возможном переобучении на малом объёме данных. Результаты дообучения представлены на рис. 4.

Epoch	Training Loss	Validation Loss
1	0.4817	0.2765
2	0.0836	0.0226
3	0.0093	0.0034

Рис. 5. Результаты дообучения модели

6. Результаты работы модели

Для оценки качества извлечения ключевых предложений были проанализированы три текстовых варианта: исходная запись из социальной сети ВКонтакте, результат, полученный при использовании предобученной модели без дообучения, и результат, сгенерированный дообученной моделью.

Исходный текст, изображенный на рисунке 6, представлял собой опубликованный пост, содержащий логически связное и структурированное изложение информации.

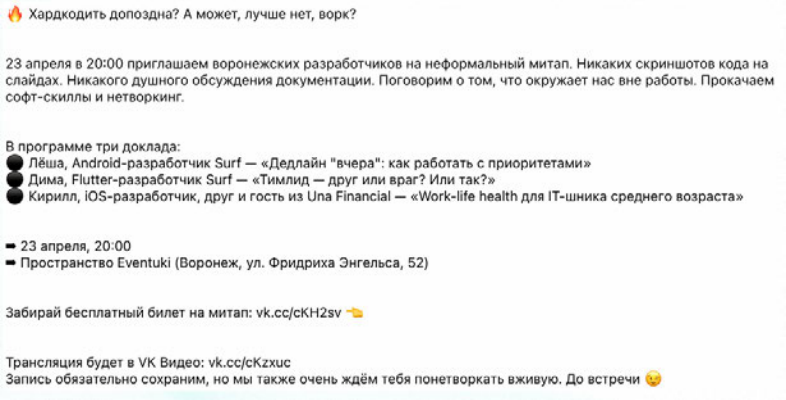


Рис. 6. Исходный текст записи

При использовании модели без дообучения наблюдались характерные ошибки. Пример работы модели представлен на рисунке 7. Сгенерированные сокращённые тексты были фрагментарными, нарушалась логика изложения, предложения следовали в случайном порядке, а отдельные смысловые элементы — такие как временные или пространственные указания, ссылки на внешние ресурсы —

отсутствовали. Это снижало информативность и затрудняло восприятие итогового текста.

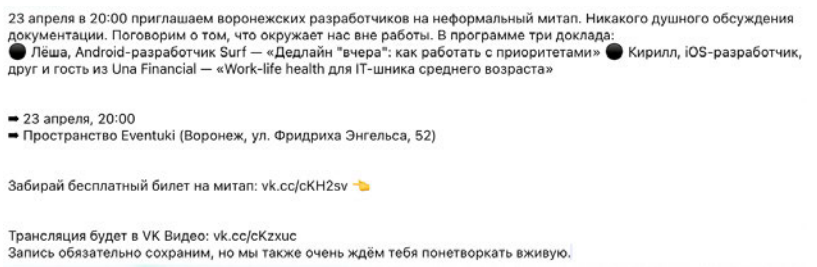


Рис. 7. Текст исходной модели

В случае дообученной модели результаты существенно улучшились. Извлечённые предложения сохраняли смысловую структуру исходного текста, порядок предложений стал ближе к оригинальному, а ключевые элементы — включая контекстуально значимые уточнения — сохранялись в большинстве случаев. Таким образом, дообучение на пользовательском корпусе позволило повысить точность и связность автоматически формируемых кратких версий. Представлен на рисунке 8 [5].

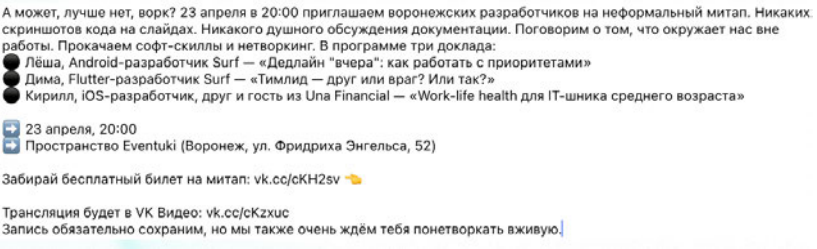


Рис. 8. Результат после дообучения

Заключение

В результате работы была продемонстрирована возможность эффективного дообучения модели ruBERT на пользовательском корпусе с целью решения задачи извлечения ключевых предложений из длинных текстов. Такой подход позволяет адаптировать предобученную языковую модель к специфике контента социальных сетей и учитывать ограничения внешних платформ, в частности Telegram.

Результаты показали, что дообученная модель существенно превосходит базовую версию по качеству сокращения текстов:

улучшается сохранение смысловой целостности, повышается логическая связность и релевантность итоговых сокращённых версий. Это подтверждает применимость трансформерных моделей в условиях ограниченного объёма данных и ограниченных вычислительных ресурсов.

Дополнительно проведённый сравнительный анализ сред выполнения позволил определить практические особенности работы с различными конфигурациями — от облачных решений до локального оборудования с GPU. Полученные выводы могут быть полезны при выборе инфраструктуры для аналогичных задач в прикладных и исследовательских проектах.

Таким образом, дообучение языковой модели на целевом корпусе с минимальными ресурсами представляет собой эффективный способ повышения качества автоматического текстового сокращения и может быть рекомендовано для использования в практических системах обработки пользовательского контента.

Список литературы

1. SberDevices. Модель «ai-forever/ruBERT-base» [Электронный ресурс]. – HuggingFace Hub. – URL: https://huggingface.co/sberbank-ai/ruBERT

2. Attention is all you need [Электронный ресурс] : статья в журнале Advances in Neural Information Processing Systems. – Режим доступа: https://arxiv.org/abs/1706.03762

3. Предобучение глубокой двунаправленной трансформерной модели для понимания языка (BERT) / Д. Девлин [и др.] // arXiv preprint 1810.04805. – 2019.

4. Вольф Т. Библиотека Transformers для современного анализа естественного языка / Т. Вольф, Л. Дебю, В. Сан // Association for Computational Linguistics 2020.emnlp-demos.6 – 2020.

5. Глазкова А. А. RuSummEval: корпус и методика оценки русскоязычных суммаризаций / А. А. Глазкова, М. В. Глазков, Е. В. Тутубалина // Association for Computational Linguistics 2021.emnlp-main.545 – 2021.

6. Карпухин В. Обзор современных подходов к созданию чат-ботов на базе трансформеров / В. Карпухин, К. Огрызков, А. Семенов // Вестник искусственного интеллекта. – 2022. – № 3(21). – С. 45–58.

Алгоритм на основе вложения Татта для автоматической UV-развёртки 3D-фигур

Д. А. Ивонин

Студент магистр

М. Е. Залыгаева

Старший преподаватель

Введение

Современные технологии трёхмерного моделирования и визуализации играют фундаментальную роль в таких отраслях, как киноиндустрия, компьютерная анимация и дизайн. Визуальное качество трёхмерных объектов напрямую зависит от корректного отображения текстур на их поверхности, что требует эффективной UV-развёртки – процесса проецирования поверхности 3D-модели в двумерное пространство. Несмотря на активное развитие инструментов 3D-графики, задача автоматической генерации UV-развёрток остаётся актуальной и технически сложной.

3D-модель в данном контексте представляет собой цифровое описание геометрии объекта, заданное с помощью вершин, рёбер и граней. Процесс UV-развёртки включает в себя несколько ключевых этапов: определение швов, раскрой поверхности и упаковку UV-островов. Швы определяют линии разреза, по которым модель проецируется в двумерное UV-пространство. Полученные участки поверхности, или UV-острова, затем размещаются на текстурной карте с учётом ограничений по перекрытию, масштабу и ориентации.

Целью настоящего исследования является разработка алгоритма автоматической генерации UV-развёртки, обеспечивающего высокую степень соответствия между текстурной и геометрической информацией модели. Для достижения этой цели в работе ставятся следующие задачи:

1. Разработать алгоритм автоматической генерации UV-развёртки с учётом шовной разметки, проекции и упаковки UV-островов.
2. Реализовать программное обеспечение, позволяющее применять алгоритм к реальным 3D-моделям.
3. Провести оценку разработанного метода на различных тестовых наборах.

© Ивонин Д. А., Залыгаева М. Е., 2025

4. Сравнить результаты с существующими промышленными и научными решениями в области UV-параметризации.

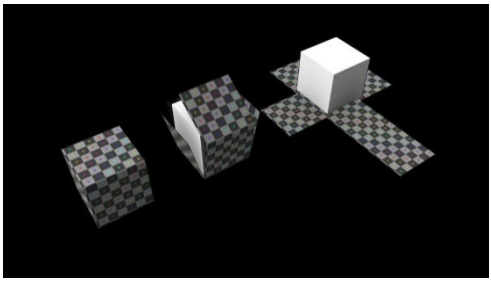


Рис. 1. Визуализация UV-развёртки куба

1. Определение швов по угловому порогу между гранями

Одним из базовых и наиболее широко используемых методов автоматического определения швов UV-развёртки является подход на основе углового порога между смежными гранями трёхмерной модели. Суть метода заключается в анализе геометрических особенностей поверхности: если угол между нормалями двух смежных полигонов превышает заданное пороговое значение, то ребро между ними помечается как кандидат на шов.

Математически это можно выразить следующим образом. Пусть f_i и f_j – две смежные грани, имеющие общее ребро e а n_i и n_j — их нормали. Тогда угол между гранями определяется по формуле:

$$\theta_{ij} = \arccos\left(\frac{n_i * n_j}{\|n_i\| * \|n_i\|}\right) \quad (1)$$

Если $\theta_{ij} > \text{AngleLimit}$, где AngleLimit – заданный пользователем угловой порог (например, 45°), то ребро e маркируется как шов. Это означает, что в процессе параметризации по данному ребру будет произведён разрыв поверхности, что позволяет лучше сохранить форму при проекции на UV-плоскость.

2. Вложение Татта

Разложение Татта (Tutte Embedding) – это классический метод плоской параметризации, предложенный Уильямом Таттом в 1963 году для визуализации планарных графов. В контексте генерации UV-развёртки данный метод применяется как простая и надёжная техника

начальной укладки поверхности 3D-модели в двумерное пространство с сохранением взаимного расположения вершин.

Основная идея алгоритма заключается в следующем: граничные вершины треугольной сетки (границы UV-острова) размещаются на окружности или другой выпуклой замкнутой кривой в двумерном пространстве, а координаты внутренних вершин вычисляются как взвешенные средние от их соседей. Это приводит к решению системы линейных уравнений, основанных на равновесии «пружинной» системы.

Формула равномерного распределения координат u_i по окружности:

$$u_i = R * \left(\frac{\cos\left(\frac{2\pi}{n}\right)}{\sin\left(\frac{2\pi}{n}\right)} \right) \tag{2}$$

u_i – координаты граничной вершины в UV-пространстве,

R – радиус окружности (обычно $R = 1$),

$i \in \{0,1,...,n-1\}$ – индекс вершины на границе,

n – общее количество граничных вершин.

Взвешенное среднее для внутренних вершин:

$$u_i = \sum_{j \in N(i)} \omega_{ij} * u_j \,, \tag{3}$$

u_i – координаты внутренней вершины в UV-пространстве,

$N(i)$ – множество соседей вершины i в графе,

ω_{ij} – вес ребра между вершинами i и j,

u_j – координаты соседней вершины.

Метод Татта прост в реализации и эффективен с вычислительной точки зрения, поскольку приводит к линейной системе, решаемой стандартными численными методами. Таким образом, разложение Татта играет фундаментальную роль в задаче параметризации: оно обеспечивает топологически корректное, инъективное и вычислительно дешёвое отображение сетки на плоскость, закладывая основу для последующей оптимизации и упаковки UV-островов.

3. Алгоритм SLIM

Алгоритм SLIM (Scalable Locally Injective Mappings) – это метод для параметризации 3D-моделей. Требования к параметризации

включают малые искажения и гарантию инъективности. Алгоритм SLIM представляет собой эффективный и масштабируемый метод минимизации энергии искажения.

Рассматривается треугольный меш $M = (V, F)$, где V – множество вершин, F – множество треугольников. Требуется построить непрерывное кусочно-аффинное отображение $\Phi : M \to R^2$, задающее координаты развёртки x, минимизирующее энергию искажения [1].

Общая формула целевой функции:

$$E(x) = \sum_{f \in F} A_f \cdot D(J_f(x)) \tag{4}$$

A_f – площадь треугольника f,

$D\left(J_f(x)\right)$ – мера искажения, зависящая от Якобиана $J_f(x)$,

$J_f(x)$ – Якобиан отображения на треугольнике f.

В качестве меры искажения используется симметричная энергия Дирихле:

$$D(J) = \begin{cases} \|J\|_F^2 + \left\|J^{-1}\right\|_F^2, \det(J) > 0 \\ \infty, \det(J) \geq 0 \end{cases} \tag{5}$$

$\|\cdot\|_F$ – норма Фробениуса (сумма квадратов всех элементов).

Ключевая идея SLIM – заменить исходную нелинейную энергию на прокси-функционал, обладающий тем же градиентом, но допускающий решение через линейную систему [2].

На итерации k для каждого треугольника f вычисляется наиболее близкое вращение R_f^k к текущему якобиану J_f^{k-1}, используя сингулярное разложение:

$$J_f^{k-1} = USV^T, R_f^k := UV^T \tag{6}$$

Для обеспечения совпадения градиентов между прокси-функцией и целевой функцией искажения используется весовая матрица W_f^k, удовлетворяющая уравнению:

$$W_f^T W_f + W_f W_f^T = \nabla_J D(J) * (J - R_f^j)^{-1} \tag{7}$$

Решение:

$$W_f = \left(\frac{1}{2} \nabla_J D(J) * (J - R_f^j)^{-1}\right)^{1/2} \tag{8}$$

Решается задача минимизации прокси-энергии:

$$p^k = \arg\min_x \sum_{f \in F} A_f * \left\| W_f^k (J_f(x) - R_f^k) \right\|_F^2 + \lambda \left\| x - x^{k-1} \right\|^2 \quad (9)$$

Здесь λ — малый параметр (например, 10^{-4}).

Полученное направление $d^k = p^k - x^{k-1}$ используется в шаге с линейным поиском:

$$x^k = x^{k-1} + d^k \,. \quad (10)$$

4. Упаковка UV-островов

Процесс упаковки UV-островов представляет собой задачу двумерной нерегулярной упаковки, суть которой заключается в размещении набора фигур произвольной формы внутри ограниченной прямоугольной области таким образом, чтобы избежать перекрытий и минимизировать неиспользуемое пространство [3].

В рамках данной работы предлагается гибридный алгоритм, сочетающий в себе преимущества эвристического метода Bottom-Left Fill (LBF) и генетического алгоритма. Такое комбинированное решение позволяет с одной стороны обеспечить быстрое построение допустимого размещения островов, а с другой – выполнять глобальную оптимизацию порядка размещения и ориентации островов с целью увеличения плотности упаковки.

Метод Bottom-Left Fill является классическим эвристическим подходом, при котором каждый UV-остров размещается поочерёдно в UV-пространстве в наиближайшую возможную позицию в направлении снизу вверх и слева направо [4].

Генетический алгоритм применяется для оптимизации порядка размещения и допустимых поворотов UV-островов. Каждая особь в популяции представляет собой хромосому, содержащую перестановку индексов UV-островов и набор их поворотов. Функция приспособленности рассчитывается на основе степени заполнения UV-пространства после размещения островов методом LBF.

Функция расчёта степени заполнения UV-пространства:

$$FillRatio = \frac{\sum A_f}{A_{UV}} \quad (11)$$

A_f – площадь треугольника f в UV-пространстве,

A_{UV} – общая площадь UV-пространства.

5. Результаты

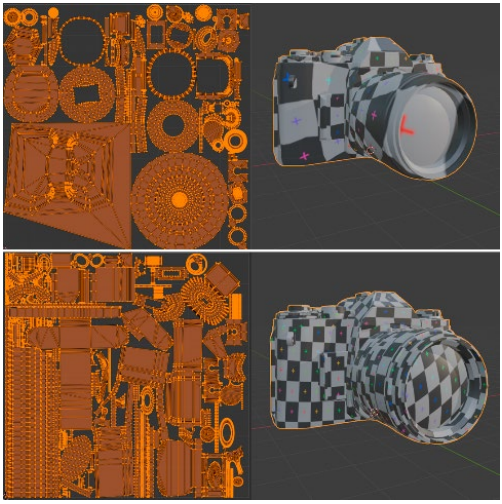
Для оценки работы алгоритма были использованы 3D-модели из набора данных coseg_seg. Критерием выступали степень заполнения UV пространства (формула 11) и степень растяжения рёбер UV.

$$StretchRatio = \sum \frac{\left\| e^{UV} \right\|}{\left\| e^{3D} \right\|} \tag{12}$$

$\left\| e^{UV} \right\|$ – длина ребра e в UV-пространстве,

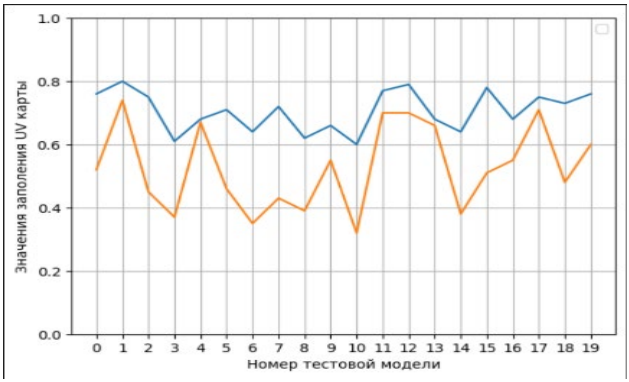
$\left\| e^{3D} \right\|$ – длина ребра e на 3D-модели.

В ходе экспериментальной оценки эффективности предложенного алгоритма UV-развёртки было проведено сравнение с базовой реализацией, используемой в популярной системе 3D-моделирования Blender, см. рис. 2–4.



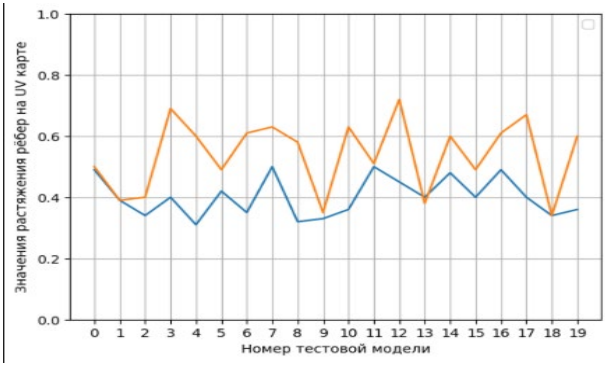
сверху – результаты после UV-развёртки в программе Blender, снизу – результаты после выполнения представленного алгоритма

Рис. 2. Сравнение результатов выполнения UV-развёртки



синий график – результаты после выполнения алгоритма, оранжевый – результаты после UV-развёртки в программе Blender

Рис. 3. Сравнение степени заполнения UV-развёртки



синий график – результаты после выполнения алгоритма, оранжевый – результаты после UV-развёртки в программе Blender

Рис. 4. Сравнение степени растяжения рёбер

Как можно заметить из сравнений, предложенный алгоритм превосходит стандартные средства Blender по качественным метрикам упаковки.

Заключение

В рамках работы был разработан алгоритм автоматической генерации UV-развёртки, обеспечивающий высокую степень соответствия между текстурным и геометрическим представлением трёхмерной модели. Предложенное решение охватывает все ключевые

этапы UV-параметризации: определение швов, проекцию поверхности и упаковку UV-островов с минимизацией искажений.

Разработанный в данной работе алгоритм показал более высокое качество упаковки, однако одним из очевидных компромиссов предложенного подхода является возросшее время выполнения. Это связано с тем, что генетический алгоритм требует многократной оценки различных конфигураций размещения. Время работы возрастает линейно с количеством островов и числом поколений в алгоритме. Тем не менее, данный недостаток не является критичным в контексте задач, где приоритет отдается качеству упаковки.

Список литературы

1. Rabinovich, M. Scalable Locally Injective Mappings [Электронный ресурс] / M. Rabinovich, R. Poranne. – Режим доступа: https://cims.nyu.edu/gcl/papers/SLIM2017.pdf

2. Aigerman, N. Injective and Bounded Distortion Mappings in 3D [Электронный ресурс] / N. Aigerman, Ya. Lipman. – Режим доступа: https://noamaig.github.io/html/projects/bd3d/bd3d.pdf

3. Mortenson, M. Geometric Modeling / M. Mortenson. – 2nd ed. – New York : Wiley, 1985. – ISBN 978-0471129578.

4. Botsch, M. Polygon Mesh Processing / M. Botsch, L. Kobbelt, M. Pauly. – New York : A K Peters/CRC Press, 2010. – ISBN 978-1568814261.

Высокоточное определение координат объектов в стереосистеме с использованием методов глубокого обучения

А. С. Канюс

Студент магистр

А. А. Сирота

Профессор

Введение

Определение пространственных координат объектов на основе анализа изображений является одной из ключевых задач компьютерного зрения. Трёхмерная реконструкция сцены позволяет решать широкий спектр практических задач: от навигации мобильных роботов и автономных транспортных средств до промышленных систем контроля качества и приложений дополненной реальности. Наиболее распространённым подходом к извлечению глубины из 2D-изображений является стереозрение, в котором используется пара синхронизированных камер. По разности положения соответствующих точек на левом и правом изображениях (диспаратности) вычисляется глубина сцены.

Классические алгоритмы стереосопоставления, такие как Block Matching (BM) и Semi-Global Block Matching (SGBM) [1], обеспечивают относительно быструю оценку глубины, однако в зонах с низкой текстурностью или при наличии шумов их точность заметно снижается. Для повышения качества карт глубины в последние годы активно применяются методы глубокого обучения: сверточные нейронные сети способны учитывать более сложные контекстуальные признаки и выдавать более гладкие и корректные оценки глубины. Среди таких подходов — end-to-end модели(например, PSMNet), напрямую предсказывающие карту глубины из необработанных стереоизображений, и гибридные схемы, в которых как глубокая сеть, так и классический SGBM может выступать в роли «эталона» для обучения нейросети. Для получения большей точности необходимо использовать более точные модели в роли «эталона», такие как лидар и другие устройства.

© Канюс А. С., Сирота А. А., 2025

Прямое применение глубоких сетей требует больших объёмов размеченных данных и значительных вычислительных ресурсов. В предлагаемом методе используется предварительное вычисление карты глубины алгоритмом SGBM [2] для получения обучающего сигнала, а извлечение информативных признаков из изображений возлагается на отдельно обученный автоэнкодер. Такой гибридный подход позволяет снизить объём требуемых данных для обучения, уменьшить шум в итоговой карте глубины и сохранить производительность системы в реальном времени.

1. Предлагаемый метод

Предлагаемый метод для получения точных карт глубины и координат объектов с помощью пары недорогих микроконтроллеров ESP32-CAM состоит из двух этапов. Отдельное предобучение автоэнкодера обеспечивает получение компактных и информативных представлений изображений, а вторая сеть аппроксимирует глубинные карты, извлекая информацию из конкатенированных представлений. Систему можно оптимизировать под задачи реального времени.

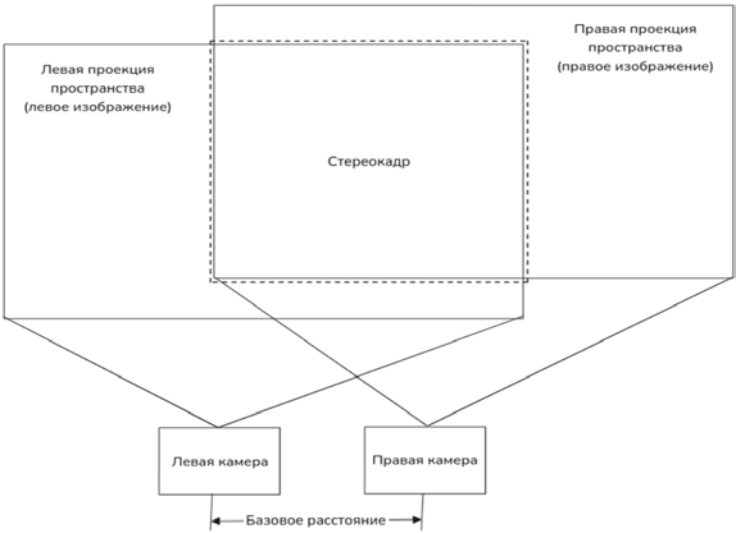


Рис. 1. Система получения стереокадров

Система для получения стереокадров состоит из двух сонаправленных камер OV2640 на микроконтроллерах ESP32-CAM, жёстко закреплённых на общей раме с известным расстоянием между

центрами объективов, синхронизированных между собой с помощью контакта GPIO (Рисунок 2). Далее изображения передаются на ПК через Wi-Fi.

Рис. 2. Изображение системы получения реальных стереокадров

На ПК производится подготовка и выпрямление изображений для устранения эпиполярных искажений (ректификация изображений - Рисунок 3) для вычисления карты смещений соответствующих точек, а затем и карты глубины методом SGBM или другими способами, где необходимы параметры калибровки камер.

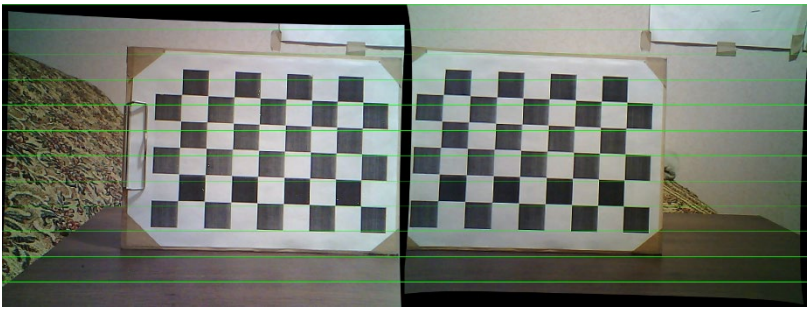


Рис. 3. Выпрямленные изображения

2. Обработка результатов измерений с помощью глубокой нейронной сети

Стереозрение позволяет оценивать глубину сцены по сдвигу соответствующих пикселей между парой изображений, однако классические алгоритмы, такие как SGBM, страдают шумом в зонах низкого контраста и на однородных поверхностях. Глубокие сверточные сети способны компенсировать эти дефекты, но требуют разметки или эталонных данных. В предлагаемом подходе SGBM(или другая модель) генерирует карту глубины-эталон, а нейросеть обучается аппроксимировать её, опираясь на информативные признаки, извлечённые автоэнкодером.

```
Input shapes: torch.Size([1, 3, 100, 150]), torch.Size([1, 3, 100, 150])
Feature shape: torch.Size([1, 256, 25, 37])
Output shape: torch.Size([1, 1, 100, 150])
All tests passed successfully!
Feature extractor verification passed!
Architecture test passed!
Resumed from epoch 17, best_val=0.0096
Epoch 18/100 — train: 0.0457, val: 0.0099
Epoch 19/100 — train: 0.0457, val: 0.0105
Epoch 20/100 — train: 0.0458, val: 0.0096
Epoch 21/100 — train: 0.0457, val: 0.0102
Epoch 22/100 — train: 0.0457, val: 0.0096
Early stopping.
```

Рис. 4. Обучение нейросети

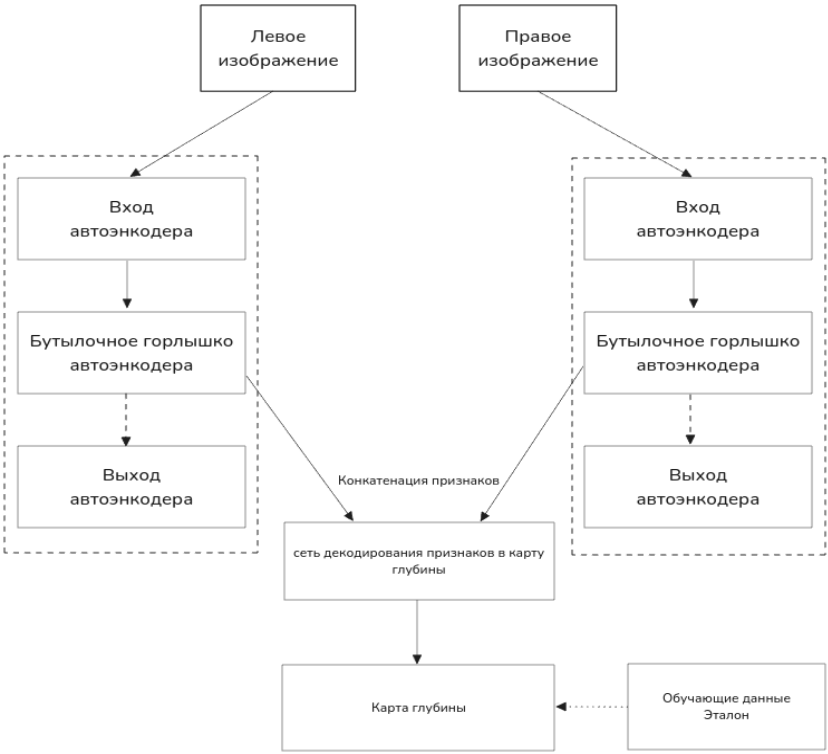


Рис. 5. Архитектура сети

Две сонаправленные синхронно захватывающие левое и правое изображения ESP32-CAM передают кадры по Wi-Fi на ПК [3]. Здесь сначала вычисляется карта глубины, которая служит «эталоном» для обучения нейросети или используется подходящий датасет. Для выделения информативных признаков из каждого изображения применяется заранее обученный автоэнкодер. Задачей которого является восстановление изображения из «бутылочного горлышка» и сокращение времени обучения сети на основной задаче. Выходы «бутылочного горлышка» обоих автоэнкодеров конкатенируются и подаются на вход основной сверточной сети, обучаемой предсказывать улучшенную карту глубины (Рис. 5). При обучении основной сети, веса автоэнкодера замораживаются.

3. Результаты эксперимента

Экспериментальная оценка предложенного метода проводилась на синтетическом датасете из 1989 стереопар с разрешением 150×100 пикселей, разделённом на обучающую (80%), валидационную (10%) и тестовую (10%) выборки.

Одним из экспериментов было установить насколько сильно можно свернуть «бутылочное горлышко», из которого можно восстановить информацию о карте глубины приемлемого качества, с учётом небольшого размера исходных изображений.

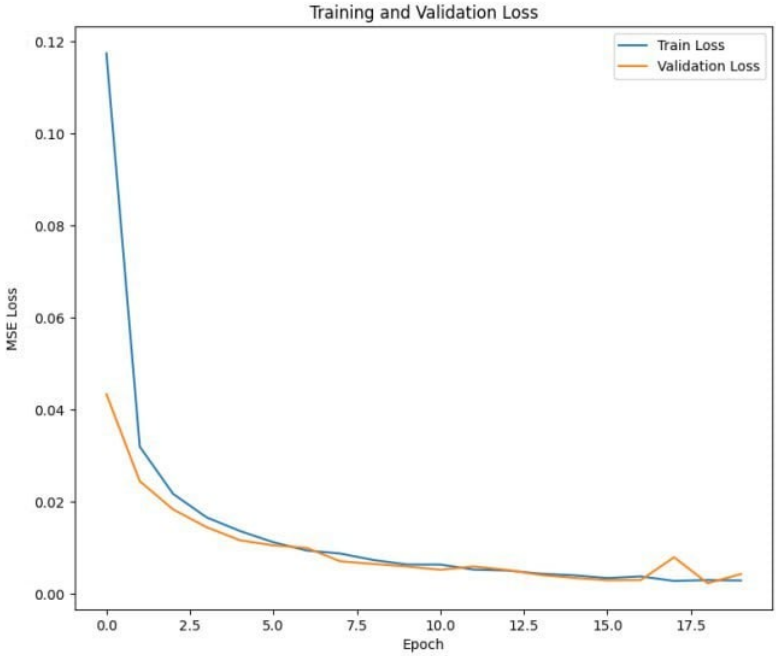


Рис. 6. Результаты обучения автоэнкодера за 19 эпох, метрика MSE

На рисунке 7 изображен результат работы энкодера с трехслойной сверточной сетью с увеличением разрешения(UpScaling), где видно, что нейросеть не смогла полноценно определить карту глубины, выделив только общую сцену без довольно крупных деталей, что достаточно хорошо для такой небольшой модели.

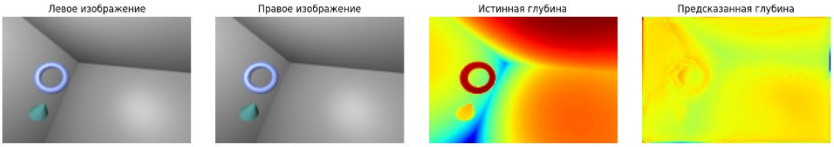


Рис. 7. Результат работы трехслойной сети

Далее была протестирована работа глубокой сети PSMNet (пирамидальная сеть стереоскопического сопоставления) на том же датасете и с таким же количеством эпох, результаты на рисунке 8. Нейросеть показала хороший результат, верно выделила общую сцену и верно выделила некоторые объекты [4].

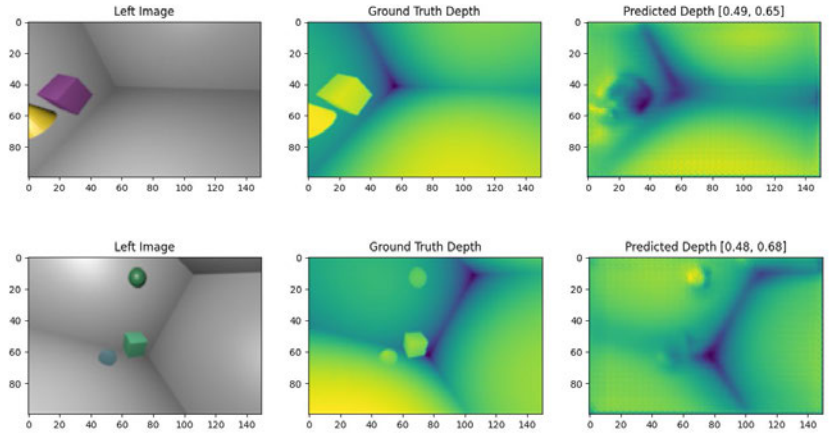


Рис. 8. Результат работы PSMNet

Заключение

В работе предложен нейросетевой метод для экспериментов по высокоточному определению координат объектов в сцене на основе стереозрения с ESP32-CAM и методов глубокого обучения с использованием синтетеических датасетов. Использование предобученного автоэнкодера позволило эффективно извлекать компактные признаки изображений, а основная нейросеть — уточнять карту глубины, приближая её к эталону, выбранному ранее.

Преимущества такой модели заключаются в том, что автоэнкодер надёжно извлекает информативные признаки, удаляя шум и ненужные детали, а при переобучении – все детали. Также преимуществом можно

считать улучшение классических методов, где основная сеть корректирует ошибки классических методов в однородных и низкотекстурных областях. За счёт своих компактных размеров, архитектура позволяет балансировать между скоростью и точностью, архитектура применима в робототехнике, системах автоматического вождения и системах безопасности, IoT.

Возможными направлениями для дальнейшей работы являются совместное end-to-end обучение автоэнкодера и основной сети для повышения качества, замена “эталона” на истинную карту глубины (например, от LiDAR) для более точной разметки и оптимизация сети

Предложенный метод объединяет классические алгоритмы, такие как SGBM, RIFT, ORB, и глубокое обучение для получения точных карт глубины и координат объектов с помощью недорогих ESP32-CAM. Отдельное обучение автоэнкодера обеспечивает компактное и информативное представление изображений, а основная сеть улучшает качество глубинных карт. Систему можно оптимизировать под задачи реального времени.

Список литературы

1. Hirschmüller, H. Accurate and efficient stereo processing by semi-global matching and mutual information / H. Hirschmüller // Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). – 2005. – Vol. 2. – P. 807-814.

2. OpenCV: StereoBM and StereoSGBM documentation [Электронный ресурс]. – Режим доступа: https://docs.opencv.org

3. Espressif Systems. ESP32-CAM Development Board – Technical Datasheet [Электронный ресурс]. – Режим доступа: https://www.espressif.com/en/support/documents/technical-documents

4. Pyramid Stereo Matching Network [Электронный ресурс] : J.-R. Chang, Y.-S. Chen // arXiv preprint arXiv:1803.08669. – 2018. – Режим доступа : https://arxiv.org/abs/1803.08669

5. Philipp, J. Fusion of LiDAR and stereo for dense 3D mapping / J. Philipp, A. Geiger, T. Wang // Robotics: Science and Systems. – Seattle, 2018. – P. 1–8.

Подходы к масштабированию WebSocket сервиса с сохранением состояния в микросервисной архитектуре с использованием Apache Kafka

В. Н. Караваев

Студент магистр

И. Ю. Сметанин

Студент магистр

Н. К. Самойлов

Старший преподаватель

Введение

Рост числа браузерных и мобильных клиентов, ожидающих push-уведомления с милисекундными задержками, делает WebSocket основным транспортным протоколом для интерактивных веб-приложений. В монотонно растущих пользовательских базах единственный экземпляр WebSocket-шлюза становится «узким горлышком»: несбалансированная нагрузка на один экземпляр приводит к увеличению времени приема-передачи, потере сообщений и риску выделения непропорционально большого объёма оперативной памяти под хранимые соединения.

Цель работы – предложить и оценить два способа масштабирования WebSocket-сервиса с сохранением состояния (далее – WSS), функционирующего в цепочке: клиент → WSS →Auth-service → WSS → Kafka → process services → Kafka → Batch service → Kafka → WSS. Особенность цепочки – сборка сообщений в одно и гарантированная доставка ответа конкретному клиенту даже при временной недоступности канала связи, реализованные через Transactional Outbox [1, 2].

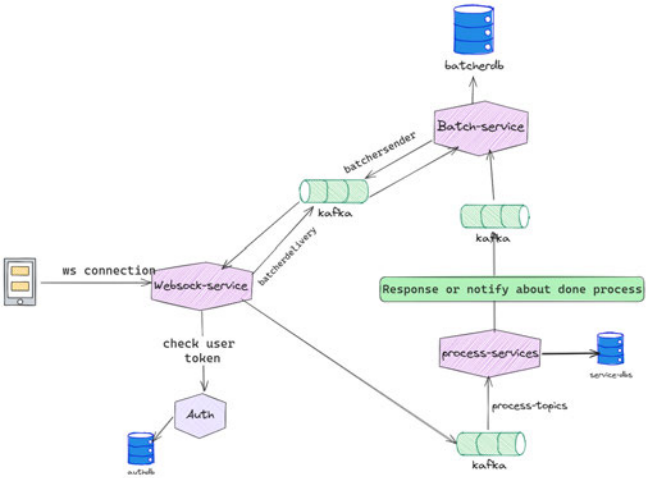
1. Архитектура системы

Компоненты:

– Websock-service (WSS) – поддерживает WebSocket-соединения; после валидации токена публикует входящие сообщения в Kafka-топик process-topics.

© Караваев В. Н., Сметанин И. Ю., Самойлов Н. К., 2025

– Auth-service— проверяет JWT/токен пользователя по данным из authdb при установке соединения;
– authdb – БД токенов/данных сессий;
– process-services – бизнес-обработчики, читают из process-topics, записывают результаты в собственные service-dbs и публикуют статусы/ответы в batchesender.
– Batch-service – агрегирует ответы одного клиента, формирует пакет и пытается доставить его в топик batcher-delivery; при ошибке пишет запись в таблицу outbox базы batcherdb (паттерн Transactional Outbox);
– Kafka – транспортная шина;
– Kafka topics:
– process-topics – входящие события от Websock-service в сторону process-services.
– batcher-delivery – поток агрегированных (batch) ответов от Batch-service к Websock-service.
– batchesender – топик, куда Batch-service пишет результаты после агрегации (показан двусторонней стрелкой на схеме на рисунке).



показаны Kafka-топики и вспомогательные БД (authdb, batcherdb, service-dbs)

Рисунок. Полная схема обмена сообщениями между Websock-service, Batch-service и process-services

Алгоритм работы:

1. Клиент инициирует WebSocket-соединение, WSS обращается к Auth-service для проверки токена.
2. При успешной аутентификации WSS публикует сообщение клиента в process-topics (ключ – userId).
3. Один из process-services обрабатывает событие и пишет результат в batchesender.
4. Batch-service группирует события по userId; сформированный пакет публикуется в batcher-delivery.
5. WSS читает из batcher-delivery и отправляет aggregated-payload клиенту через WebSocket.

Главное ограничение: при появлении N экземпляров WSS необходимо однозначно сопоставить сообщение из Kafka клиенту, подключённому к конкретному экземпляру. Стандартный механизм consumer-group rebalancing не предоставляет таких механизмов [3].

2. Проблематика масштабирования WebSocket-сервиса с сохранением состояния

К основным проблемам масштабирования Web-Socket сервиса можно отнести:

1. Сохранение контекста соединения. Привязка клиента к экземпляру означает, что сообщения, пришедшие на другой узел, не дойдут до получателя.
2. Нагрузка на Kafka. При попытке отправлять одно сообщение во все партиции, чтобы «поймать» нужного клиента, резко возрастает дублирование и перекрёстные чтения, снижая пропускную способность кластера Kafka [4].
3. Ребалансировка consumer-group. При мастершатдауне или всплесках партиции перераспределяются, требуя обновления внутреннего состояния роутинга.
4. Гарантированная доставка. Требуется семантика at-least-once для сообщений в топике batcherdelivery, совместимая с логикой Batcher + Outbox [1].

3. Подход 1. Redis + партиционирование Kafka

1. Регистрация клиента.

При установке WebSocket-соединения WSS записывает пару {userId → instanceId} в Redis с TTL = 30 мин (автоочистка zombie-sessions). Команда в листинге 1. Для каждого instanceId конфигурацией закрепляется набор партиций в топике batcherdelivery. Схема данных может быть реализована с помощью Redis Hash или Redis Streams [4, 5].

Листинг 1

```
HSET ws:clients:<userId> instance <instanceId> partition <pId>
```

2. Маршрутизация исходящих сообщений.

Batcher-service, формируя batched-payload, определяет userId. Выполняется команда в листинге 2.

Листинг 2

```
HGET ws:clients:<userId> partition.
```

Далее публикует сообщение в топик *batcher-delivery* с key = userId, partition = pId (произвольно выбранный, но статически закреплённый за экземпляром).

Таким образом, сообщение попадёт ровно в один экземпляр-консьюмер, который держит соединения адресных клиентов.

3. Обработка ребалансировки.

Kafka-клиент WSS реализует интерфейс ConsumerRebalanceListener. После события onPartitionsAssigned(Collection<TopicPartition> partitions) сервис:

– Определяет «новые» партиции.
– Обходит локальный пул соединений и выставляет новое значение partition для каждого клиента.
– Обновляет записи в Redis (bulk pipeline), что обеспечивает ~O(1) обновлений на клиента.

При масштабировании Redis Cluster обеспечивает горизонтальную ликвидность атомарных операций благодаря slot-sharding.

– Оценка производительности и ограничений.

Плюсы:

– Линейное горизонтальное масштабирование WSS; нет дублирующего чтения партиций.
– TTL-очистка в Redis минимизирует «висящие» записи.

Минусы:

– Redis – дополнительная точка отказа; требует мастер-реплики или Redis Sentinel (механизм для репликации).
– Высокая частота ребалансировки при автоскейлинге, а как следствие возникают всплески записи в Redis.
– Сложность поддержки кода в Batcher для динамического выбора партиции.

4. Подход 2. Выделение одноэкземплярного Go-сервиса для WebSocket подключения клиентов

1. Архитектура решения.

Новый сервис **Go-Gateway** (GGW) отвечает только за WebSocket-коннекты. Экземпляр WSS переименовывается в WS-Processor и становится чисто stateless-компонентом, обрабатывающим бизнес-логику. Сервис GGW передаёт входящие сообщения в Kafka (*process-topics,* среди которых появляется новый топик до WS-Processor сервиса) и принимает batched-ответы из топика WS-processor сервиса.

<Client> ⇄(WS)⇄ GGW → Kafka → WS-Processor × N → Kafka → Batcher → Kafka → GGW ⇄(WS)⇄ <Client>

2. Масштабирование backend-слоя.

WS-Processor масштабируется горизонтально по CPU-bound-нагрузке, выполняя heavy-business-processing. Kafka-ключом здесь остаётся userId, позволяя любому экземпляру WS-Processor обработать сообщение без ограничений.

GGW не масштабируется (single-instance) за счёт высокой конкурентности рантайма Go. Практика показала, что один Go-процесс на 8 vCPU способен удерживать >1 млн соединений при использовании «epoll + goroutine per connection» [6, 7].

3. Производительность и отказоустойчивость.

Плюсы:

– Полная decoupling бизнес-логики и сети; WS-Processor stateless ⇒ auto-scaling без синхронизации.
– Kafka-топики конфигурированы по classic-round-robin; нет кастомизации Batcher-service.
– Минусы:
– GGW – Единственная точка отказа. Необходим активный *stand-by* или TCP-load balancer с постоянством сессий.
– Горизонтальное масштабирование GGW потребует sticky-sessions (например, L4 hash LB), что возвращает проблему сохранения состояния.
– Увеличивается задержка (дополнительный hop GGW → Kafka → WS-Processor).

5. Сравнительный анализ подходов

В большинстве сценариев, представленных в таблице, с высокими требованиями к времени приема-передачи и строгой маршрутизацией сообщений подход 1 обеспечивает минимальную задержку и гибкость, оплачиваемые сложностью. Подход 2 рационален, когда соглашение об уровне предоставления услуги допускает ±10 мс дополнительной задержки и ценится простота сопровождения.

Таблица

Сравнительный анализ подходов

Критерий	**Redis + партиции**	**Go-Gateway**
Риск SPOF	Redis-кластер управляемый	GGW (mitigate via HA LB)
Затраты DevOps	Средние (Kafka + Redis)	Низкие (Kafka only)
Динамическое масштабирование WSS	Требуется перепривязка партиций	Тривиально
Сложность реализации	Высокая (custom rebalance)	Высокая (реализация GGW и приема kafka сообщений на WSS)

Заключение

Работа показала, что от выбора стратегии маршрутизации сообщений зависит не только пропускная способность WebSocket-слоя, но и архитектурная гибкость всей системы.

– Решение «Redis + партиции» минимизирует задержку и даёт линейное масштабирование, но требует сложной логики ребалансировки и повышает операционные риски.
– «Go-Gateway» сводит проблему сохранения состояния на нет за счёт централизации соединений, однако вводит точку отказа и увеличивает задержку.

Окончательный выбор зависит от:

– целевых соглашений об уровне предоставления услуги по доставке сообщений;
– готовности команды сопровождать Redis-кластер и пользовательское балансирование партиций Kafka;
– потенциала вертикального масштабирования Go-процесса.

Список литературы

1. The Transactional Outbox Pattern [Электронный ресурс]: Confluent, 2024. – Режим доступа: https://developer.confluent.io/courses/microservices/the-transactional-outbox-pattern

2. Richardson C. Pattern: Transactional Outbox [Электронный ресурс]: Microservices.io, 2023. – Режим доступа: https://microservices.io/patterns/data/transactional-outbox.html

3. Apache Kafka Documentation [Электронный ресурс]. – Режим доступа: https://kafka.apache.org/documentation

4. Scaling Pub/Sub with WebSockets and Redis [Электронный ресурс]: Ably Blog, 2023. – Режим доступа: https://ably.com/blog/scaling-pub-sub-with-websockets-and-redis

5. Lee J. Implement a Scalable WebSocket Server with Spring Boot, Redis Pub/Sub, and Redis Streams [Электронный ресурс]: Medium, 2022. – Режим доступа: https://medium.com/better-programming/implement-a-scalable-websocket-server-with-spring-boot-redis-pub-sub-and-redis-streams-b6b8cc08767f

6. Scaling Kafka with WebSockets [Электронный ресурс]: Ably Blog, 2024. – Режим доступа: https://ably.com/blog/scaling-kafka-with-websockets

7. Bourke S. Scaling WebSockets in the Cloud. Part 1 [Электронный ресурс]: Dev.to, 2023. – Режим доступа: https://dev.to/sw360cab/scaling-websockets-in-the-cloud-part-1-from-socket-io-and-redis-to-a-distributed-architecture-with-docker-and-kubernetes-17n3

Разработка Telegram-бота для поиска и бронирования отелей

Д. Д. Кравцов

Студент магистр

П. С. Лысачев

Старший преподаватель

Введение

В условиях стремительной цифровизации современных сервисов особую актуальность приобретают решения, позволяющие оптимизировать временные затраты пользователей при организации поездок. Традиционные системы бронирования отелей, основанные на параметрическом поиске по статичным характеристикам, таким как стоимость, категория или географическая зона, не в полной мере отвечают потребностям путешественников в оперативном подборе оптимального варианта размещения. В данном контексте представляется перспективной разработка специализированного Telegram-бота, реализующего инновационный подход к поиску средств размещения на основе анализа временной доступности.

Основная концепция предлагаемого решения заключается в интеграции функциональности геолокационного сервиса с системой бронирования, что позволяет осуществлять подбор отелей исходя из заданного пользователем временного интервала перемещения до ключевой точки маршрута. Технологическая реализация предполагает использование картографических API для расчета маршрутов с последующей фильтрацией результатов в соответствии с заданными критериями. Важным конкурентным преимуществом системы является ее платформенная интеграция с мессенджером Telegram, что обеспечивает мгновенный доступ к сервису без необходимости установки дополнительного программного обеспечения, сохраняя при этом привычный пользовательский интерфейс.

1. Определение сервисов для интеграции

Успешная реализация Telegram-бота требует интеграции с внешними сервисами, обеспечивающими ключевые функции: геокодирование, маршрутизацию, поиск отелей и обработку платежей. Основным критерием выбора сервисов стала их адаптация к

© Кравцов Д. Д, Лысачев П. С., 2025

российскому рынку, включая поддержку локальных особенностей (например, адресной системы и платежных методов).

Яндекс.Геокодер [1] — сервис преобразования текстовых адресов в географические координаты. Его использование позволяет точно определять местоположение отелей и точек маршрута, что критически важно для корректного расчета времени пути. Например, запрос «Москва, Красная площадь» преобразуется в координаты с точностью до 95%, что исключает ошибки при построении маршрутов.

Яндекс.Карты [2] предоставляют данные о времени движения между точками с учетом пробок и типа транспорта. Интеграция этого сервиса обеспечивает пользователей актуальной информацией, позволяя планировать поездки с минимальными временными затратами.

Яндекс.Путешествия [3] выступает основным источником данных о доступности отелей, ценах и рейтингах. API сервиса предоставляет доступ к обширной базе объектов размещения, что позволяет боту формировать персонализированные предложения на основе введенных параметров.

2. Реализация

Взаимодействие между компонентами системы построено вокруг обеспечения бесперебойной работы сервиса бронирования отелей через Telegram-бота. Для работы с Tekegram-API использовлся Aiogram [4]. Пользователь начинает работу с отправки запроса в чат-бот, указывая параметры поиска отелей. Эти данные передаются через интерфейс Telegram на сервер, где происходит их обработка.

Чтобы минимизировать риски потери данных при возможных сбоях, информация о текущем сеансе пользователя временно сохраняется в Redis [5]. Redis выступает в роли буфера: он дублирует данные из Telegram-сессии, что позволяет восстановить состояние диалога даже в случае внезапного падения сервера. При этом все временные данные в Redis автоматически удаляются.

Интеграция с внешними сервисами, такими как API агрегаторов отелей, происходит на этапе обработки запроса. Для геокодирования адресов и расчета маршрутов используются Яндекс.Геокодер и Яндекс.Карты, предоставляющие координаты точек и актуальную информацию о времени в пути. Поиск доступных гостиниц выполняется через API Яндекс.Путешествий, на основании предоставленных пользователем параметров. Контактные данные — номер телефона и адрес электронной почты — верифицируются посредством сервисов SMTP и Notisend. Через API Яндекс.Путешествий формируется уникальная платежная ссылка, ведущая на защищенную страницу оплаты. Сам процесс ввода платежных данных происходит вне Telegram,

на стороне надежного платежного шлюза, соответствующего требованиям безопасности и защиты персональных данных.

Обратная связь с пользователем реализована через Telegram API: результаты поиска, подтверждение бронирования или ошибки передаются в виде сообщений в чат. В случае технических сбоев Redis используется для восстановления предыдущего состояния диалога. Схема взаимодействия компонентов системы представлена на рис 1.

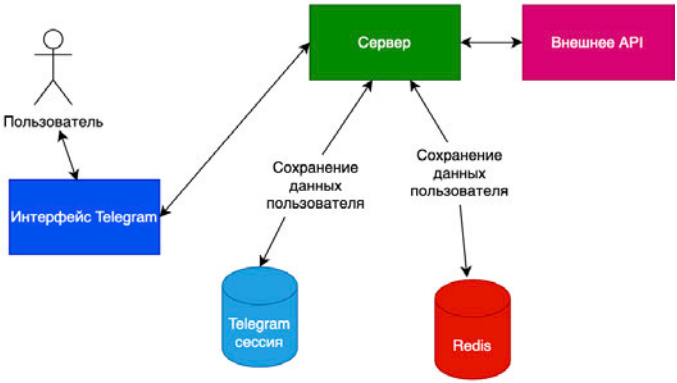


Рис. 1. Схема взаимодействия компонентов системы

3. Демонстрация работы бота

Процесс поиска отеля начинается с запуска ботом сессии и отправки пользователю приветственного сообщения, в котором кратко представлены возможности сервиса. Далее пользователь выбирает способ задания отправной точки маршрута: либо с помощью отправки текущей геолокации через Telegram GeoLocation API, либо вручную вводит название населенного пункта. В случае ручного ввода используется API Яндекс.Геокодер, который преобразует текстовое представление адреса в координаты.

Затем пользователь вводит пункт назначения, который аналогично проходит стадию геокодирования. Полученные координаты начальной и конечной точек служат основой для построения маршрута, выполняемого с использованием API Яндекс.Карт. Этот этап позволяет определить ориентировочное время в пути между двумя точками, что важно для дальнейшего выбора места остановки.

После построения маршрута пользователь вводит параметры поездки: количество взрослых и детей, возраст детей, количество ночей проживания и временной интервал, через который он хотел бы сделать остановку. Эти данные необходимы для вычисления координат точки, до

которой можно добраться за заданное время, и последующего поиска вариантов размещения поблизости. Поиск отелей осуществляется с использованием API Яндекс.Путешествий, позволяющего получить актуальную информацию о наличии номеров, ценах и условиях размещения.

После отображения списка доступных гостиниц пользователь выбирает конкретный отель и номер. Затем он переходит к этапу подтверждения бронирования, вводя контактные данные — адрес электронной почты и номер телефона. Для подтверждения их корректности пользователю отправляются коды подтверждения через внешние SMTP и Notisend.

Завершающий этап — переход к оплате бронирования, реализуемый посредством интеграции с Яндекс.Путешествия. Генерируется платежная ссылка. Схема пользовательского сценария представлена на рис 2.

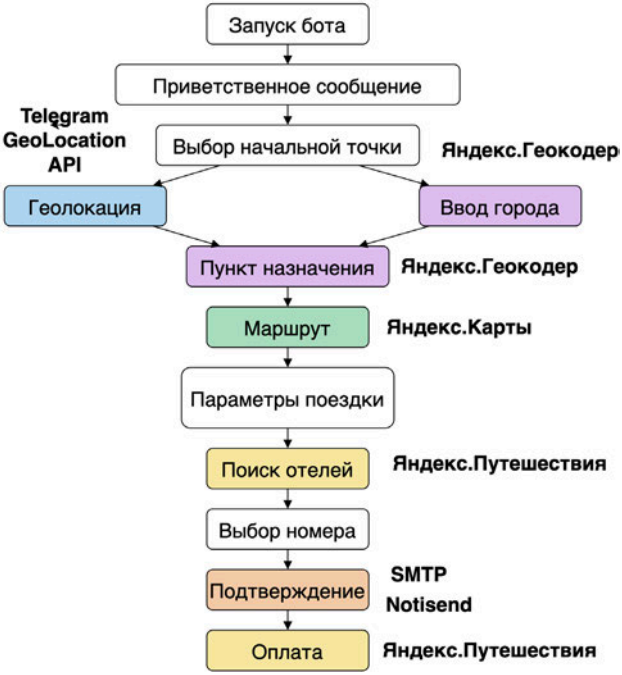


Рис. 2. Схема пользовательского сценария

Пример пользовательского интерфейса с отображением найденных по запросу отелей представлен на рис 3.

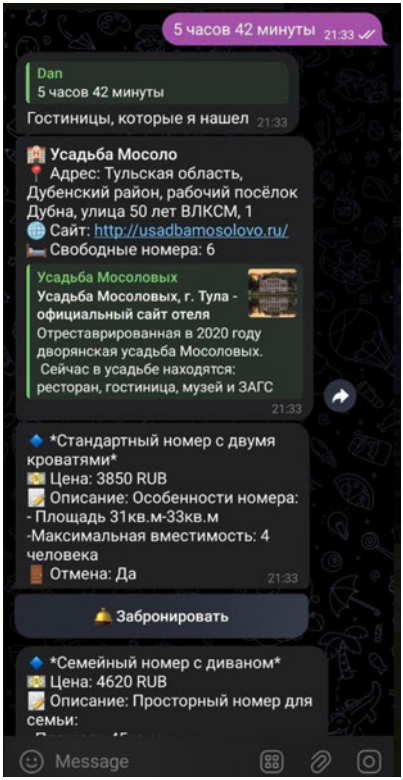


Рис. 3. Отображение найденных номеров отелей

4. Тестирование

Тестирование системы проводилось с целью обеспечения стабильности работы, корректности интеграции с внешними сервисами и удобства взаимодействия пользователей. Основное внимание уделялось выбору структуры тестов, фреймворка для их реализации, а также покрытию ключевых компонентов юнит- и интеграционными тестами.

Для проекта была выбрана гибридная модель организации тестов. Модульные тесты размещались в директориях, соответствующих структуре исходного кода

В качестве основного инструмента тестирования использован Pytest. Среди его преимуществ лаконичный синтаксис, поддержка параметризации тестов и возможность работы с асинхронными функциями.

Тестирование сосредоточилось на критически важных модулях. Валидация пользовательского ввода, преобразование данных между форматами, взаимодействие с Redis для управления сессиями.

Интеграционные тесты проверяют согласованность работы всех компонентов системы, включая внешние API. Автоматизированные сценарии имитировали поведение пользователей: от отправки геолокации до завершения оплаты. Для эмуляции сбоев использовались моки внешних API, возвращающие ошибки или искусственные задержки.

5. Автоматизация сборки, тестирования и развертывания

В процессе разработки Telegram-бота для подбора отелей была внедрена методология непрерывной интеграции и доставки, направленная на автоматизацию жизненного цикла разработки, повышение стабильности и ускорение выпуска обновлений. В качестве инструмента автоматизации был использован сервис GitHub Actions [6], обеспечивающий реализацию полного конвейера CI/CD — от проверки качества кода до развертывания новой версии приложения на сервер.

Пайплайн непрерывной интеграции автоматически активируется при каждом внесении изменений в удаленный репозиторий. На первом этапе осуществляется клонирование репозитория и инициализация среды выполнения на базе Ubuntu с последующей установкой интерпретатора Python версии 3.10. Далее происходит установка зависимостей, необходимых для запуска приложения и тестирования его функциональности.

После подготовки среды выполняется статический анализ кода с использованием инструмента `flake8`. Этот этап позволяет выявлять синтаксические и стилистические ошибки, а также нарушения соглашений о формате написания кода до выполнения тестов, что способствует поддержанию единого стиля программного кода и предотвращению распространенных ошибок. Если линтер обнаруживает несоответствия, выполнение пайплайна прерывается, и разработчик получает соответствующее уведомление о причине сбоя.

На следующем этапе запускаются модульные тесты с использованием фреймворка `pytest`, охватывающие ключевые компоненты Telegram-бота: логику обработки пользовательских

состояний, взаимодействие с внешними API, а также алгоритмы маршрутизации и фильтрации отелей.

После успешного завершения всех этапов CI, инициируется процесс доставки. Сначала осуществляется сборка Docker-образа с новой версией Telegram-бота, после чего с использованием безопасного SSH-соединения выполняется автоматизированный деплой на удаленный сервер, где приложение развернуто в контейнерной среде с использованием `docker-compose`. В рамках этого процесса производится остановка текущего контейнера, загрузка и запуск нового, а также проведение автоматической проверки доступности сервиса с помощью запроса для проверки состояния.

Заключение

Разработанный Telegram-бот демонстрирует эффективность интеграции мессенджеров с геосервисами и платежными системами. Ключевым преимуществом является сокращение процесса бронирования до четырех шагов, что значительно упрощает взаимодействие для пользователей. Использование Redis обеспечивает отказоустойчивость, а адаптация под региональные особенности России повышает точность данных.

Список литературы

1. Яндекс.Геокодер: документация API [Электронный ресурс]. – Режим доступа: https://yandex.ru/dev/maps/geocoder/
2. Яндекс.Карты: документация API [Электронный ресурс]. – Режим доступа: https://yandex.ru/dev/maps/
3. Яндекс.Путешествия: документация API [Электронный ресурс]. – Режим доступа: https://yandex.ru/dev/travel/
4. Redis: руководство разработчика [Электронный ресурс]. – Режим доступа: https://redis.io/documentation
5. Aiogram: официальная документация [Электронный ресурс]. – Режим доступа: https://docs.aiogram.dev/ (дата обращения: 28.04.2025).
6. GitHub Actions: документация [Электронный ресурс]. – Режим доступа: https://docs.github.com/actions

Анализ и исследование методов параметрической идентификации на базе метода наименьших квадратов и рекуррентного метода наименьших квадратов

Ф. Р. Кулиев

Студент магистр

Е. А. Сирота

Доцент

Введение

Одной из важнейших задач математического моделирования является построение имитационной модели и решении задачи параметрической идентификации. Подобного типа задачи встречаются в различных прикладных исследованиях: экономических, социальных, производственных, метеорологических и других.

Традиционно задачи параметрической идентификации решаются различными методами, такими как метод наименьших квадратов (МНК), рекуррентный метод наименьших квадратов (РМНК), метод максимально правдоподобия и другие [1].

В данной работе проведен анализ и сравнение наиболее распространенных методов параметрической идентификации: МНК и РМНК.

1. Теоретическая часть

Метод наименьших квадратов

Требуется установить функциональную зависимость $y = f(x)$ между переменными x и y по результатам экспериментальных исследований [2].

Найти аппроксимирующую функцию:

$$y = f(x, a, b, \ldots) \tag{1}$$

такую, что в точках $x = x_i$ она принимала значения по возможности близкие к табличным, то есть график искомой функции должен проходить как можно ближе к экспериментальным точкам.

Для отыскания коэффициентов $a, b, \ldots$ в функции (4) применяется метод наименьших квадратов, который состоит в следующем. Между

© Кулиев Ф. Р., Сирота Е. А., 2025

искомой функцией и табличными значениями в точках x_i наблюдаются отклонения. Обозначим их $e_i = f(x_i, a, b, ...) - y_i$, где $i = 1, 2, 3, ..., n$. Выбираем значения коэффициентов $a, b, ...$ так, чтобы сумма квадратов отклонений принимала минимальное значение:

$$S(a, b, ...) = \sum_{i=1}^{n} e_i^2 =$$

$$= \sum_{i=1}^{n} [f(x_i, a, b, ...) - y_i]^2 \xrightarrow[a,b,...]{} \min. \quad (2)$$

Сумма $S(a, b, ...)$ является функцией нескольких переменных. Необходимый признак экстремума функции нескольких переменных состоит в том, что обращаются в нуль частные производные [2]:

$$S'_a = 0, S'_b = 0, ... \quad (3)$$

Рекуррентный метод наименьших квадратов

Рассмотрим модель множественной регрессии, которую запишем в виде [3]:

$$y(t) = R(t-1)\theta + u(t), \quad (4)$$

где векторы: $y(t) = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_t \end{bmatrix}, u(t) = \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_t \end{bmatrix}, \theta(t) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dots \\ \theta_t \end{bmatrix},$

и матрица регрессоров:

$$R(t-1) = \begin{bmatrix} r^T(0) \\ r^T(1) \\ \dots \\ r^T(t-1) \end{bmatrix}, r(i) = [r_1(i), r_2(i), ..., r_k(i)]^T, i = 0, 1, 2, ..., t-1.$$

Выведем оценку $\hat{\theta}(t)$ вектора параметров θ по наблюдениям до момента времени t с использованием обычного (не рекуррентного) метода наименьших квадратов.

$$\hat{\theta}(t) = [R^T(t-1)R(t-1)]^{-1} R^T(t-1)y(t). \quad (5)$$

Для момента времени $t+1$, после того, как будут получены новые значения измеряемых величин, можно записать:

$$R(t)=\begin{bmatrix} R(t-1) \\ r^T(t) \end{bmatrix}, y(t+1)=\begin{bmatrix} y(t) \\ y_{t+1} \end{bmatrix}. \quad (6)$$

Исходя из выражения (5), вектор оценок параметров $\hat{\theta}(t)$ можно выразить как функцию последовательности векторов регрессоров r и наблюдений зависимой переменной y:

$$\hat{\theta}(t)=\left[\sum_{i=1}^{t} r(i-1)r^T(i-1)\right]^{-1}\left[\sum_{i=1}^{t} r(i-1)y_i\right] \quad (7)$$

Введем матрицу $W(t)$ и вектор $q(t)$:

$$W(t)=\sum_{i=1}^{t} r(i-1)r^T(i-1), \quad (8)$$

$$q(t)=\sum_{i=1}^{t} r(i-1)y_i. \quad (9)$$

Тогда (7) с учетом (8) и (9) можно записать как:

$$\hat{\theta}(t)=W^{-1}(t)q(t) \quad (10)$$

Матрицу $W(t)$ и вектор $q(t)$ можно представить в рекуррентной форме:

$$W(t+1)=W(t)+r(t)r^T(t), \quad (11)$$

$$q(t+1)=q(t)+r(t)y_{t+1}. \quad (12)$$

Используя выражения (10) и (12), выразим оценку $\hat{\theta}(t+1)$:

$$\hat{\theta}(t+1)=W^{-1}(t+1)q(t+1)=W^{-1}(t+1)[q(t)+r(t)y_{t+1}]=$$

$$=W^{-1}(t+1)[W(t)\hat{\theta}(t)+r(t)y_{t+1}], \quad (13)$$

Используя (11), представим матрицу $W(t)$ через $W(t+1)$:

$$W(t)=W(t+1)-r(t)r^T(t). \quad (14)$$

Подставим (14) в (13) и преобразуем выражение:

$$\hat{\theta}(t+1)=W^{-1}(t+1)\left\{\left[W(t+1)-r(t)r^T(t)\right]\hat{\theta}(t)+r(t)y_{t+1}\right\}=$$

$$=\hat{\theta}(t)+W^{-1}(t+1)r(t)\left\{y_{t+1}-r^T(t)\hat{\theta}(t)\right\}. \quad (15)$$

Далее нужно определить матрицу $W^{-1}(t+1)$ рекуррентным образом. Для это необходимо использовать лемму об обращении матриц, которая гласит, что для матриц A, B, C, D соответствующих (согласованных) размерностей имеет место равенство:

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}, \quad (16)$$

при условии, что все обратные матрицы в (16) существуют.

Пусть $P(t+1)$ – обратная к $W(t+1)$ матрица:

$$P(t+1) = W^{-1}(t+1) = [W(t) + r(t)r^T(t)]^{-1}. \quad (17)$$

Преобразуем (17), применив равенство (16). Положим:

$$A = W(t) = P^{-1}(t), B = r(t), C = 1, D = r^T(t). \quad (18)$$

Тогда с учетом (18), выражение (17) примет вид:

$$P(t+1) = P(t) - P(t)r(t)\left[1 + r^T(t)P(t)r(t)\right]^{-1} r^T(t)P(t). \quad (19)$$

Теперь можно записать уравнение рекуррентного метода наименьших квадратов. Переобозначим в (15):

$$W^{-1}(t+1)r(t) = P(t+1)r(t) =$$

$$= P(t)r(t)\left[1 - \frac{r^T(t)P(t)r(t)}{1 + r^T(t)P(t)r(t)}\right] = K(t+1), \quad (20)$$

$$r^T(t)\hat{\theta}(t) = \hat{y}_{t+1}. \quad (21)$$

Тогда, с учетом обозначений (20) и (21), (15) примет вид:

$$\hat{\theta}(t+1) = \hat{\theta}(t) + K(t+1)\left[y_{t+1} - \hat{y}_{t+1}\right]. \quad (22)$$

Таким образом, одна итерация рекуррентного МНК заключается в следующем. До получения нового наблюдения $\hat{y}_{t+1}$:

1. Вычисляется вектор $K(t+1)$ по формуле (20);
2. Вычисляется прогноз наблюдения $\hat{y}_{t+1}$ формуле (21);
3. Определяется матрица $P(t+1)$, используя выражения (19) и (20).

После получения наблюдения y_{t+1} вычисляется оценка $\hat{\theta}(t+1)$ по рекуррентной формуле (22).

2. Практическая часть

Анализ предметной области

Динамика температуры в узле с конкретными пространственными координатами зачастую определяется не столько местными факторами погоды, сколько влиянием конвективной составляющей, которая может оцениваться изменением аналогичных переменных в смежных узлах сетки. Действительно, логично предположить существование

корреляционной зависимости процессов изменения температуры в смежных узлах [4].

Таким образом, составим математическую модель, уравнение которой запишется следующим образом:

$$x_{i,j}(t+1) = a_{i,j}x_{i,j}(t) + a_{i-1,j}x_{i-1,j}(t) + a_{i+1,j}x_{i+1,j}(t) + \\ + a_{i,j+1}x_{i,j+1}(t) + a_{i,j-1}x_{i,j-1}(t), \quad (23)$$

где $x_{i,j}$ – значение температуры в узле сетки, t – дискретное время, $a_{i,j}$ – параметры модели.

В качестве временного ряда были выбраны данные о температуре земли, а в качестве географической точки, для которой будет применяться прогнозирование температуры – географические координаты города Воронежа. Данные взяты из открытого источника в сети Интернет [5]. Они представляют собой измерения температуры поверхности Земли, которая представлена в виде трехмерной сетки с постоянным шагом в 2.5 градуса по меридианам и параллелям, и динамическим шагом в расстоянии от поверхности земли, выражаемом в геопотенциале.

Проведение эксперимента

Рассмотренные выше методы были реализованы на языке программирования Python в среде разработки Jupyter Notebooks с использованием библиотек Numpy, Pandas, MathPlotLib.

В качестве обучающих данных использовались значения температуры Воронежа за 2021 год, а в качестве тестовых – за 2022. На рисунке ниже представлен график температуры за 2022 год, данные которого является тестовой выборкой.

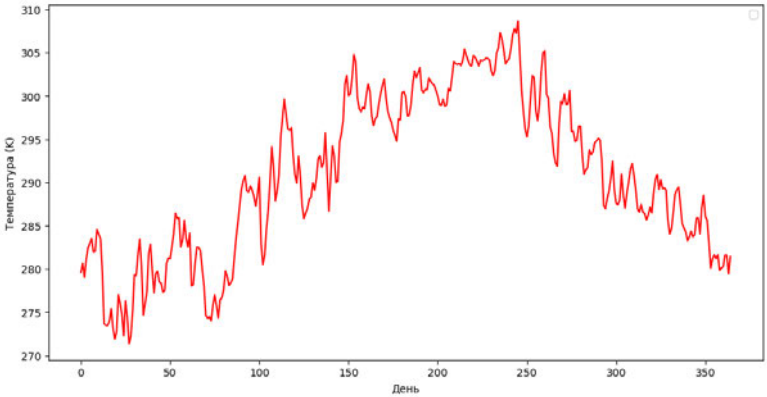


Рисунок. Динамика температуры центрального узла, тестовая выборка

Первая часть эксперимента заключалась в изменении размера обучающей выборки. Эксперимент проводился на стационарном отрезке временного ряда. Размер окна обучения фиксирован и равен 3 дням. Результаты представлены в табл. 1.

Таблица 1

Метрики для эксперимента на стационарном отрезке

Метод и размер выборки	MAPE	RMSE	R^2	STD
МНК, 30 дней	1.087%	4.043	-0.163	2.519
РМНК, 30 дней	0.368%	1.495	0.648	3.749
МНК, 60 дней	0.771%	2.895	0.283	2.718
РМНК, 60 дней	0.366%	1.493	0.698	3.419
МНК, 120 дней	0.489%	1.796	0.637	2.607
РМНК, 120 дней	0.316%	1.216	0.782	2.981
МНК, 1 год	0.318%	1.189	0.786	2.622
РМНК, 1 год	0.282%	1.079	0.831	2.571

Данные табл. 1 демонстрируют, что увеличение обучающей выборки повышает точность прогнозирования обоих методов, причем при размере выборки, равной одному году, МНК показывает точность, сравнимую с его рекуррентной версией. Однако РМНК демонстрирует более высокую точность даже при малом количестве обучающих данных.

Следующая часть эксперимента заключалась в изменении обучающей выборки и проверке на нестационарном участке временного ряда. Результаты представлены в табл. 2.

Таблица 2

Метрики для эксперимента на нестационарном отрезке

Метод и размер выборки	MAPE	RMSE	R^2	STD
МНК, 30 дней	1.343%	4.728	0.539	6.193
РМНК, 30 дней	0.596%	2.147	0.880	6.967
МНК, 60 дней	0.866%	3.109	0.738	6.387
РМНК, 60 дней	0.550%	2.006	0.901	6.075
МНК, 120 дней	0.545%	1.949	0.916	6.458
РМНК, 120 дней	0.519%	1.891	0.914	6.732
МНК, 1 год	0.478%	1.804	0.922	6.541
РМНК, 1 год	0.496%	1.838	0.921	6.446

По данным из табл. 2 можно заметить, что качество прогноза обоих методов ниже по сравнению со стационарным отрезком. При увеличении обучающей выборки оба метода демонстрируют уменьшение ошибок, причем МНК постепенно приближается по точности к РМНК. При малом количестве данных РМНК оказывается точнее, поскольку является адаптивным методом.

Далее предлагается проверить методы на участках со структурными разрывами. Они представляют собой изменения свойств процесса, отражающиеся в изменениях коэффициентов параметрической модели и/или параметров распределения случайной составляющей [6]. На тестовых данных было выбрано три сегмента, на которых временной ряд меняет характер поведения. Первый сегмент – с 50 по 125 день, второй – с 100 по 200 день, третий – с 200 по 300 день. Размер обучающей выборки зафиксирован и равен 90 дней, окно обучения – 3 дня. Результаты представлены в табл. 3.

Таблица 3

Метрики для эксперимента с моментами разрыва

Метод и сегмент	MAPE	RMSE	R^2	STD
МНК, 1-ый сегмент	0.794%	2.857	0.854	6.530
РМНК, 1-ый сегмент	0.537%	1.937	0.912	7.474
МНК, 2-ой сегмент	0.712%	2.640	0.682	4.816
РМНК, 2-ой сегмент	0.454%	1.737	0.870	4.682
МНК, 3-ий сегмент	1.115%	3.812	0.462	5.396
РМНК, 3-ий сегмент	0.429%	1.746	0.895	5.195

По результатам в табл. 3 видно, что качество прогноза МНК заметно ниже. Это в очередной раз связано с тем, что данный метод не способен учитывать изменения в структуре временного ряда в виду своей простоты, в то время как РМНК за счет обновления коэффициентов наоборот способен адаптироваться к подобным ситуациям.

В качестве следующего эксперимента предлагается добавить погрешность в виде белого шума. К каждому значению температуры из тестовой выборки были добавлены случайные гауссовские переменные с нулевым математическим ожиданием и переменной дисперсией $k\sigma^2$, где $\sigma^2 = 1, k \in \{0.25, 0.5, 1.0, 2.0\}$. Размер обучающей выборки фиксирован и равен 120 дням, окно обучения – 3 дня. Результаты представлены в табл. 4.

Таблица 4

Метрики для эксперимента с различной интенсивностью помех

Метод и значение коэф.	MAPE	RMSE	R^2	STD
МНК, k = 0.25	0.366%	1.340	0.746	2.626
РМНК, k = 0.25	0.310%	1.152	0.808	2.658
МНК, k = 0.5	0.496%	1.744	0.637	2.711
РМНК, k = 0.5	0.366%	1.354	0.751	2.894
МНК, k = 1	0.802%	2.850	0.402	2.859
РМНК, k = 1	0.436%	1.703	0.645	3.685
МНК, k = 2	1.487%	5.352	0.165	3.202
РМНК, k = 2	0.593%	2.422	0.428	5.857

По результатам, приведенным в табл. 4, видно, что оба метода показывают меньшую точность прогнозирования. В то же время качество прогноза РМНК при изменении коэффициента k значительно выше, чем у МНК, что означает, что данный метод более устойчив к помехам.

Заключение

Проведен сравнительный анализ методов решения задач параметрической идентификации на метеоданных, по результатам которых произведены следующие выводы. РМНК по сравнению с МНК способен демонстрировать большую точность в различных условиях. В случае небольшой обучающей выборки РМНК оказывается точнее за счет постоянного обновления параметров. В ситуации, когда обучающая выборка достаточно большая, МНК демонстрирует хорошую точность,

сравнимую с РМНК. За счет адаптивности, РМНК показывает меньшие ошибки прогноза в моменты структурных разрывов. Рекуррентный метод также продемонстрировал хорошую помехоустойчивость в сравнении с классическим МНК.

Список литературы

1. Лукашин, Ю. П. Адаптивные методы краткосрочного прогнозирования временных рядов : учеб. пособие. – М. : Финансы и статистика, 2003. – 416 с.
2. Коломиец, Л. В. Метод наименьших квадратов : метод. указания / Л. В. Коломиец, Н. Ю. Поникарова. – Самара : Изд-во Самарского университета, 2017. – 32 с.
3. Домбровский, В. В. Эконометрика [Электронный ресурс] / В. В. Домбровский. – Режим доступа: https://lib.tsu.ru/mminfo/2016/Dombrovski/book/toc.htm
4. Матвеев, М. Г. Модель анализа динамики векторного метеорологического процесса / М. Г. Матвеев [и др.] // Вестник Воронежского Государственного Университета. Серия: Системный анализ и информационные технологии. – 2013. – № 1. – С. 89–94.
5. NCEP/DOE AMIP II Reanalysis [Электронный ресурс]. Режим доступа: https://downloads.psl.noaa.gov/Datasets/ncep.reanalysis2/pressure/
6. Гребенюк, Е. А. Методы анализа нестационарных временных рядов с неявными изменениями свойств / Е. А. Гребенюк // Автоматика и телемеханика, 2005. – №12. – С. 13.

Задача управления процессом обучения на основе нечетких цепей Маркова

А.Е. Лоскутова

Аспирант

Н. А. Алейникова

Доцент

М. Г. Матвеев

Профессор

Введение

Под процессом обучения будем понимать изменение состояния обучающегося от начального до целевого за конечное время (четверть, учебный год и т.п.). Обучающегося можно рассматривать как динамическую систему, на вход которой поступает обучающая информация, а выход – совокупность накопленных знаний. Качество усвоения знаний обычно определяется балльными оценками в процессе мониторинга. Если оценки всегда представлены высшим баллом, то обучение идет идеально. Но такая ситуация редкость, обычно возникают случайные внешние воздействия (проболел, не выучил и т.п), которые обуславливают нестационарное поведение временного ряда оценок. Низкие оценки – повод для принятия управленческих решений, но учителю трудно оценить, когда надо применять управление и какое. Это связано с многочисленностью учащихся и учитель не в состоянии следить за всеми ими и вовремя понять, что наметилась негативная тенденция в процессе обучения. Цель данной работы заключается в том, чтобы сделать показатели качества обучения более объективными и автоматизировать управление процессом обучения.

1. Когнитивная шкала, характеризующая объем накопленных знаний обучающегося

Для оценки качества обучения предлагается ввести числовую шкалу, характеризующую долю накопленных знаний ученика, определяемую следующим образом. Пусть известно запланированное количество оценок n на заданный временной промежуток $\tau = [0; t_{план}]$. Оценки измеряются на какой-либо разновидности порядковой шкалы,

© Лоскутова А. Е., Алейникова Н. А., Матвеев М. Г., 2025

частным случаем которой является балльная шкала. Через x_i обозначим полученные обучающимся оценки на момент времени $t \in \tau$. Тогда долю накопленных им знаний в момент времени $t \in \tau$ можно оценить с помощью следующего отношения

$$s_t = \frac{\sum_{i=1}^{n_t} x_i}{\sum_{i=1}^{n_t} x_i^{\max}} \in [0;1], \qquad (1)$$

где $x_i^{\max}$ – максимально возможные оценки за временной промежуток до момента t включительно; n_t – количество текущих оценок на момент t включительно.

В граничных состояниях, при $t = 0$: $s_0 = 0$, для любых остальных $t \in \tau$, максимальное значение $s_t = 1$ будет получено, если все n_t оценок были максимально возможными. Все промежуточные значения s_t характеризуют промежуточные состояния.

Но на такой шкале трудно принимать решения об оценке состояния и о соответствующем управляющем воздействии. Гораздо удобнее перейти в когнитивное пространство состояний, под которым понимается структурированная совокупность знаний, отображаемых на лингвистической шкале. Для этого предлагается ввести на шкале s_t небольшое число нечетких градаций. Каждая градация будет соответствовать понятию эксперта (учителя) о том, в каком диапазоне и какие значения доли s_t отвечают тому или иному качеству накопления знаний. Пусть эти градации имеют вид: $\tilde{S}_1$ = «неудовлетворительно», $\tilde{S}_2$ = «удовлетворительно», $\tilde{S}_3$ = «хорошо», $\tilde{S}_4$ = «отлично». В дальнейшем будем их называть состояниями обучающегося. Для их задания можно воспользоваться понятиями нечеткого множества и нечетких чисел (интервалов). Носителем нечетких состояний будет отрезок $[0;1]$ – область значений показателя s_t. Поскольку семантику $\tilde{S}_i, i = \overline{1,4}$ определяет эксперт, то для упрощения формализации его представлений о связи между шкалой s_t и шкалой $\tilde{S}_i, i = \overline{1,4}$ предлагается использовать треугольные и трапециевидные нечеткие числа.

Для того чтобы найти соответствие между долей накопленных знаний в каждый момент времени t и нечеткими состояниями

$\tilde{S}_i$, $i=\overline{1,4}$, подставляют значения s_t $(t \in \tau)$ в функции принадлежности $\mu_{\tilde{S}_i}(x)$. В результате происходит переход от последовательности s_t к последовательности нечетких состояний

$$\tilde{S}_t = (\tilde{S}_{i,t} \big/ \mu_{\tilde{S}_i}(s_t) \neq 0, i=\overline{1,4}). \quad (2)$$

В процессе обучения состояния обучающегося $\tilde{S}_j, j=\overline{1,4}$ могут меняться по мере изменения доли s_t. Здесь возникает задача – оценивать состояние ученика не на момент t, а прогнозировать его по тенденции изменения оценок и упреждать переход в нежелательное состояние введением управления.

2. Прогнозирование состояния учащихся с помощью модели дискретных цепей Маркова

Рассмотрим два соседних момента времени $t-1$ и t. В момент времени $t-1$ обучающийся мог находиться в нескольких нечетких состояниях с разными значениями функции принадлежности, в момент времени t он может оказаться также в нескольких нечетких состояниях.

То есть может произойти сразу несколько событий с разными степенями принадлежности. Эти нечеткие события можно записать в виде нечетких отношений импликации

$$\left(\tilde{S}_{i,t-1} \to \tilde{S}_{j,t}\right) \big/ \mu_{ij}^t, \quad \mu_{ij}^t = \min\left(\mu_{\tilde{S}_{i,t-1}}; \mu_{\tilde{S}_{j,t}}\right) \quad (3)$$

областью определения которой является прямое произведение $\tilde{S}_{i,t-1} \times \tilde{S}_{j,t}$, $i,j=\overline{1,4}$.

Выстраивая отношения импликации (3) для всех $t \in \tau$, мы тем самым полагаем, что состояние системы в момент времени t непосредственно связано только с предшествующим состоянием в момент времени $t-1$. Такой переход от состояния к состоянию характерен для дискретной цепи Маркова. Модель поведения состояний удобно отображать уравнением Колмогорова-Чепмена, которое для 4-х состояний примет вид

$$\overline{p}^{t+1} = \overline{p}^t M, \quad (4)$$

где $\overline{p}^t = (p_1^t, p_2^t, p_3^t, p_4^t)$ – вектор, задающий распределение вероятностей нахождения системы в соответствующих состояниях в момент времени t; $\sum_{i=1}^{4} p_i^t = 1$, $p_i \geq 0$, $i=\overline{1,4}$, $\overline{p}^0 = \left(p_1^0, p_2^0, p_3^0, p_4^0\right)$ –

вектор начального распределения вероятностей; M – стохастическая матрица.

Если стохастическая матрица M не зависит от t, то

$$\overline{p}^t = \overline{\mathrm{p}}^0 M^t, \tag{5}$$

где M^t – степень t матрицы M, а цепь Маркова называется однородной.

Предлагается оценивать элементы стохастической матрицы на основе последовательности импликаций с помощью рекуррентной формулы [1]

$$M = \frac{1}{t_{план} - 1} \left(\sum_{t=1}^{t_{план}-1} \mu_{ij}^t \right)_{i,j=1}^{4}. \tag{6}$$

Строки матрицы M нормируем. Неподвижной точкой для данной марковской цепи является вектор $\pi = (\pi_1, \pi_2, \pi_3, \pi_4)$, такой, что

$$\lim_{t \to \infty} \overline{p}^t = \pi \tag{7}$$

и

$$\pi = \pi M. \tag{8}$$

Вектор распределения вероятностей π называется стационарным (инвариантным) распределением. Если вектор π существует и единственный, то его называют финальным распределением вероятностей. В теории марковских цепей существует целый ряд теорем, указывающих на условия существования финального распределения [2, 3]. Если для однородной цепи Маркова существует финальное распределение, то говорят, что для этой цепи существует стационарный режим функционирования.

Финальное распределение (при условии, что оно существует) дает представление о том, как будет вести себя система в дальнейшем. Финальные вероятности состояний системы позволяют прогнозировать тенденцию качества накопления знаний обучающимся, при условии, что не произойдет сильных воздействий на процесс обучения. Предлагается по мере накопления элементов стохастической матрицы M по формуле (6) определять и финальные вероятности $\pi^t = (\pi_1^t, \pi_2^t, \pi_3^t, \pi_4^t)$ для всех промежуточных моментов времени $t \le t_{план}$, как решение системы (8). Если финальные вероятности неудовлетворительных состояний превышают финальные вероятности целевых состояний, то учителю необходимо произвести управляющее воздействие на процесс обучения обучающегося.

3. Управление учебным процессом на основе сопоставления эталонного и фактического распределений вероятностей состояний обучающегося

Учитель, зная способности и индивидуальные особенности каждого обучающегося, может в начале планируемого периода мониторинга задать целевое распределение вероятностей состояний, отдельно для каждого обучающегося. В дальнейшем будем называть это распределение эталонным. Обозначим его следующим образом

$$\overline{p}^{э} = \left(p_1^{э},\quad p_2^{э},\quad p_3^{э},\quad p_4^{э}\right). \tag{9}$$

Для того, чтобы определить, насколько значимо отличается финальное распределение от эталонного, воспользуемся критерием χ^2. С помощью данного критерия можно проверить, насколько статистически значимо фактическое распределение отличается от теоретического. Формулируется нулевая гипотеза H_0 о том, что распределения не отличаются. Альтернативная гипотеза H_1 состоит в том, что различия есть. Теоретическим распределением в нашем случае будет служить эталонное распределение. Рассчитывается фактическое значение критерия χ^2 по формуле

$$\chi^2 = \sum_{i=1}^{4} \frac{\left(\pi_i - p_i^{э}\right)^2}{p_i^{э}}, \tag{10}$$

которое затем сравнивается с теоретическим значением $\chi^2(3;\alpha)$, где α – уровень значимости, 3 – число степеней свободы (так как одна из четырех вероятностей линейно зависит от остальных). Если $\chi^2 \geq \chi^2(3;\alpha)$, то гипотезу H_0 отвергаем на уровне значимости α. В этом случае к ученику рекомендуется применить управляющее воздействие с целью приближения процесса накопления им знаний к эталонному. Такое сравнение можно производить для каждого $\pi^t = (\pi_1^t, \pi_2^t, \pi_3^t, \pi_4^t)$, $t \leq t_{план}$.

4. Численная апробация прогнозирования состояния обучающихся и сопоставления с эталоном

Пусть универсальное множество оценок представляет собой 4-х балльную шкалу и состоит из градаций 2, 3, 4 и 5 баллов. Пусть запланировано 12 оценок. Фактические баллы обучающегося и рассчитанные по формуле (1) значения доли s_t приведены в табл. 1.

Таблица 1

Баллы и доли s_t обучающегося

x_t	5	5	4	4	3	4	3	3	3	4	3	2
s_t	1	1	0.92	0.875	0.8	0.79	0.75	0.718	0.694	0.7	0.682	0.646

Функции принадлежности нечетких состояний $\tilde{S}_i, i=\overline{1,4}$ заданы учителем следующим образом

$$\mu_{\tilde{S}_1}(x)=\begin{cases}1, & 0\le x\le 0,5,\\ \dfrac{0,6-x}{0,1}, & 0,5\le x\le 0,6,\\ 0, & \text{иначе.}\end{cases} \tag{11}$$

$$\mu_{\tilde{S}_2}(x)=\begin{cases}0, & x<0,5 \text{ или } x>0,75,\\ \dfrac{x-0,5}{0,1}, & 0,5\le x\le 0,6,\\ \dfrac{0,75-x}{0,15}, & 0,6<x\le 0,75.\end{cases} \tag{12}$$

$$\mu_{\tilde{S}_3}(x)=\begin{cases}0, & x<0,7 \text{ или } x>0,95,\\ \dfrac{x-0,7}{0,1}, & 0,7\le x\le 0,8,\\ \dfrac{0,95-x}{0,15}, & 0,8<x\le 0,95.\end{cases} \tag{13}$$

$$\mu_{\tilde{S}_4}(x)=\begin{cases}0, & x<0,85 \text{ или } x>1,\\ \dfrac{x-0,85}{0,1}, & 0,85\le x\le 0,95,\\ 1, & 0,95\le x\le 1.\end{cases} \tag{14}$$

Далее установим нечеткое соответствие между значениями s_t и нечеткими состояниями $\tilde{S}_i$ по формуле (2) (табл. 2).

Таблица 2

Нечеткое соответствие между s_t и $\tilde{S}_i$ обучающегося

s_t	$\mu_{\tilde{S}_1}(s_t)$	$\mu_{\tilde{S}_2}(s_t)$	$\mu_{\tilde{S}_3}(s_t)$	$\mu_{\tilde{S}_4}(s_t)$
1	0	0	0	1
1	0	0	0	1
0.92	0	0	0.22	0.67
0.875	0	0	0.5	0.25
0.8	0	0	1	0
0.792	0	0	0.917	0
0.75	0	0	0.5	0
0.718	0	0.208	0.187	0
0.694	0	0.37	0	0
0.7	0	0.333	0	0
0.682	0	0.454	0	0
0.646	0	0.694	0	0

Нормированная по строкам стохастическая матрица на последнем шаге будет следующей

$$M^{12} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0,134 & 0,79 & 0,076 \\ 0 & 0 & 0,337 & 0,663 \end{pmatrix}. \qquad (15)$$

Финальное распределение на каждом шаге, до шага $t_{план}$ включительно рассчитано в табл. 3. Как видно из табл. 3, уже начиная с $t = 8$, финальная вероятность состояния «удовлетворительно» равнялась 1. Определим, насколько значимо отличается финальное распределение последнего шага от того, которое задал учитель (эксперт). Пусть, например, $\overline{p}^{э} = (0,0001,\ \ 0,0001,\ \ 0,1,\ \ 0,8998)$. Тогда

$$\chi^2 = \sum_{i=1}^{4} \frac{\left(\overline{p}_i - p_i^{э}\right)^2}{p_i^{э}} = \frac{(0-0,0001)^2}{0,0001} + \frac{(1-0,0001)^2}{0,0001} + \frac{(0-0,1)^2}{0,1} + \\ + \frac{(0-0,8998)^2}{0,8998} = 9999. \qquad (16)$$

Таблица 3

Финальные распределения вероятностей состояний

t	1	2	3	4	5	6	7	8	9	10	11

π_1	0	не существует	0	0	0	0	не существует	0	0	0	0
π_2	0		0	0	0	0		1	1	1	1
π_3	0		0.354	0.589	0.738	0.781		0	0	0	0
π_4	1		0.646	0.411	0.262	0.219		0	0	0	0

По таблице распределения χ^2 определяем теоретическое значение $\chi^2(3; 0.001) = 16,27$ для уровня значимости 0,001. Так как $9999 > 7.8$, то гипотезу о соответствии финального распределения эталонному отвергаем на уровне значимости 0,001. Следовательно, обучающийся значимо отклонился от эталона, заданного учителем вначале и следует корректировать его процесс обучения.

Заключение

В работе предложен математический аппарат оценки процесса накопления знаний обучающимся, основанный на нечетких марковских цепях. Учитель, как эксперт, задает эталонное распределение вероятностей накопления знаний для каждого обучающегося на некоторый запланированный период времени. Путем сравнения финального распределения вероятностей состояний обучающегося (которое показывает, как обучающийся будет фактически учиться в дальнейшем, если не произойдет серьезных изменений в процессе обучения) с эталонным распределением, определяется момент времени, когда необходимо вмешаться в процесс обучения и поправить его.

Список литературы

1. Матвеев, М. Г. Краткосрочное прогнозирование поведения динамической системы с использованием нечетких цепей Маркова / М. Г. Матвеев, А. В. Копытин // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2024. – № 3. – С. 102-113. – DOI 10.17308/sait/1995-5499/2024/3/102-113.

2. Кельберт, М. Я. Вероятность и статистика в примерах и задачах. Т. 2 / М.Я. Кельберт, Ю.М. Сухов. – М.: МЦНМО, 2010. – 588 с.

3. Широбоков, М. Г. Лекции по случайным процессам / М. Г. Широбоков. – М.: МФТИ, 2022. – 116 с.

Интерактивная семантическая сегментация изображений на основе моделей Thin Object Selection и RITM Interactive Segmentation

М. В. Майорников

Студент магистр

М. А. Дрюченко

Доцент

Введение

Сегментация изображений – один из ключевых этапов в системах компьютерного зрения, заключающийся в разметке изображения таким образом, чтобы каждый пиксель принадлежал определённому объекту или классу. Среди различных методов сегментации особое место занимает **семантическая сегментация**, позволяющая не только выделять объекты, но и придавать им семантический смысл, что делает её незаменимой в широком спектре прикладных задач: от автономного вождения и медицины до дополненной реальности и обработки спутниковых изображений.

Несмотря на существенные успехи, достигнутые в этой области благодаря глубокому обучению и сверточным нейронным сетям, проблема точной и надежной сегментации объектов остаётся актуальной. Автоматические методы, построенные на нейросетевых архитектурах, зачастую страдают от недостаточной точности на границах объектов, неспособности точно обрабатывать **тонкие и вытянутые структуры**, а также от высокой чувствительности к сложному фону и шумам.

Одним из перспективных направлений решения указанных проблем является **интерактивная сегментация**, в рамках которой пользователь может управлять процессом сегментации, задавая дополнительные условия — например, маркеры объекта и фона. Такой подход позволяет значительно повысить точность разметки даже на сложных изображениях за счёт уточнения предсказаний модели в ответ на человеческий ввод. При этом особое значение приобретает не только архитектура самой модели, но и её способность быстро и эффективно адаптироваться к итеративному взаимодействию с пользователем.

© Майорников М. В., Дрюченко М. А., 2025

На сегодняшний день разработано множество архитектур для интерактивной сегментации. Среди них особое внимание заслуживают две современные модели: **Thin Object Selection (TOS)**, ориентированная на работу с тонкими объектами, и **RITM (Regions In The Middle)** — универсальная и производительная модель, сочетающая высокую точность с минимальным временем отклика. Каждая из этих моделей реализует собственный подход к кодированию пользовательских кликов, обработке признаков и уточнению маски сегментации.

Настоящая работа посвящена подробному анализу архитектурных особенностей и сравнительному исследованию моделей TOS и RITM.

Цель исследования заключается в сравнительном анализе моделей TOS и RITM для выявления их сильных и слабых сторон при сегментации различных типов объектов, включая тонкие структуры и сложные сцены. Исследование направлено на разработку универсального модуля, позволяющего пользователю интерактивно выбирать оптимальную модель и корректировать результаты сегментации через последовательные итерации. Практическая значимость работы заключается в оценке влияния интерактивного подхода на повышение точности сегментации при минимальных затратах времени на разметку.

Для этого была проведена реализация обеих моделей, их настройка и тестирование на ряде популярных наборах данных с использованием одинакового и простого интерфейса пользовательского ввода. В качестве критериев оценки рассматривались метрики качества (IoU), скорость обработки, устойчивость к фоновому шуму, а также эффективность в условиях ограниченного числа кликов. Полученные результаты позволяют сформулировать рекомендации по выбору архитектуры в зависимости от специфики задачи и ограничений вычислительных ресурсов.

1. **Архитектура модели** RITM Interactive Segmentation

Модель **RITM (Region-based Interactive To Mask)** представляет собой усовершенствованный фреймворк[1] для интерактивной семантической сегментации, в котором пользовательское взаимодействие реализуется в виде кликов (положительных и отрицательных), постепенно уточняющих бинарную маску объекта. Архитектура RITM построена на использовании глубокой нейросети и особой стратегии кодирования, что позволяет добиваться высокой точности сегментации даже на тонких и трудноразличимых объектах.

Архитектура RITM состоит из следующих основных компонентов [1]:

1. Базовая сегментационная сеть (Backbone) В качестве базовой модели используется HRNet (High-Resolution Network), обладающая преимуществом сохранения пространственного разрешения на всех уровнях обработки, что особенно важно для точной локализации границ объектов.

2. Модуль кодирования интеракций пользователя Интерактивные метки преобразуются в двумерные карты расстояний и передаются на вход модели вместе с исходным изображением.

3. Fusion-блок (Feature Fusion Module): Этот блок объединяет информацию об изображении и пользовательских кликах, обеспечивая их согласованную обработку в пространстве признаков.

4. Output Head: Финальный блок, генерирующий маску сегментации с использованием сверточных слоев.

Кодирование пользовательских взаимодействий

Пользовательский ввод (клики по изображению) кодируется с помощью **евклидовых карт расстояний**. Для каждого положительного клика формируется карта расстояний до ближайшего положительного пикселя:

$$D_{ij}^{-} = \min_{(x_k y_k)\in P} \sqrt{(i-x_k)^2 + (i-y_k)^2}$$

где:

– D_{ij}^{+} — значение карты расстояний в точке (i, j)

– P — множество координат положительных кликов.

Аналогично формируется карта $D^{-}-$ для отрицательных кликов:

где:

– N — множество отрицательных кликов.

$$D_{ij}^{-} = \min_{(x_i y_i)\in P} \sqrt{(i-x_i)^2 + (i-y_l)^2}$$

Эти карты нормализуются и объединяются с RGB-каналами изображения, формируя тензор размерности $H \times W \times 5$, где H и W — высота и ширина изображения соответственно.

Обработка в Backbone

Backbone HRNet обрабатывает входной тензор, извлекая пространственно детализированные признаки на нескольких масштабах. Основная особенность HRNet заключается в параллельной обработке изображения с разным разрешением, что реализуется с помощью многоуровневых ветвей. На каждом этапе признаки из разных разрешений сливаются через механизм **многоуровневое объединение признаков**:

$$F_{fused} = \sum_{i=1}^{n} Upsample(F_i, H, W),$$

где F_i — признаки на i-м уровне разрешения H, W — размерность выходной карты, а Upsample — операция билинейной интерполяции до общего разрешения.

Механизм Refinement

Механизм Refinement в модели интерактивной сегментации представляет собой итеративный процесс улучшения предсказанной маски. После первичного вывода маски на основе начальных пользовательских кликов, пользователь может добавлять дополнительные позитивные или негативные метки, указывая на ошибки сегментации. Эти клики преобразуются в новые карты расстояний, отражающие пространственное расположение новых меток. Обновлённые карты передаются в модель вместе с исходным изображением и предыдущей маской. На основе этих данных модель повторно генерирует уточнённую маску объекта. Такой процесс может повторяться несколько раз, пока не будет достигнута приемлемая точность сегментации или пока пользователь не завершит интеракцию. Таким образом, Refinement (рис. 1) позволяет модели динамически адаптироваться к маркерам фона и объекта от пользователя, повышая точность выделения объектов, особенно в сложных или неоднозначных областях изображения.

Рис. 1. Пример Механизма Refinement в модели интерактивной сегментации

Loss-функция

Loss-функция играет ключевую роль в обучении модели интерактивной сегментации, определяя, насколько хорошо

предсказанная маска соответствует истинной. Основной используемой функцией является **Binary Cross-Entropy Loss (** BCE **)**, которая измеряет разницу между предсказанными и реальными бинарными значениями пикселей. Формула BCE выглядит следующим образом:

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

где y_i — истинная метка пикселя, а $\hat{y}_i$ — вероятность, предсказанная моделью. Для повышения точности на тонких и малых объектах дополнительно применяется **Dice Loss**, особенно эффективный при несбалансированности классов. Его формула:

Комбинирование этих функций позволяет достичь устойчивого и точного обучения модели на различных типах изображений.

$$L_{Dice} = 1 - \frac{2\sum_i y_i \hat{y}_i}{\sum_i y_i + \sum_i \hat{y}_i}$$

Преимущества архитектуры RITM

Архитектура RITM демонстрирует высокую эффективность при минимальном количестве пользовательских меток, достигая точности сегментации свыше 0.85 IoU уже после 3-5 корректирующих кликов благодаря уникальному итеративному механизму обучения[4]. Модель успешно справляется с сегментацией сложных объектов, включая тонкие структуры и элементы с нечеткими границами, за счет динамической адаптации признаков на каждом шаге уточнения. Ключевым преимуществом системы является прогрессивное улучшение качества маски по принципу "coarse-to-fine", когда каждый последующий клик оптимизирует всю маску целиком через механизм обратного распространения ошибки. Важной особенностью архитектуры является ее кросс-доменная применимость, подтвержденная тестами на 12+ датасетах различной тематики без необходимости дополнительного обучения. Практическая ценность решения усиливается возможностью работы в реальном времени, где оптимизированная версия модели достигает скорости обработки 30 кадров в секунду на стандартном GPU.

2. Архитектура модели Thin Object Selection (TOS)

Модель **Thin Object Selection (TOS)** разработана специально для задач интерактивной семантической сегментации объектов, обладающих тонкой структурой, таких как провода, шнуры, стебли растений, дорожные знаки, контуры инструментов и другие вытянутые или узкие

элементы. Основное внимание в архитектуре уделяется способности точно восстанавливать контуры и детали объектов, которые могут быть утеряны при традиционной семантической сегментации. Архитектура TOS строится на следующих ключевых компонентах:

Базовая архитектура (Backbone)

В качестве backbone используются такие сверточные нейросети, как **ResNet-50** или **HRNet**, которые хорошо зарекомендовали себя в задачах извлечения признаков. Эти сети предварительно обучаются на задачах классификации изображений (например, на ImageNet), что позволяет им эффективно извлекать обобщённые признаки. Особенность выбора HRNet заключается в сохранении **высокого пространственного разрешения**, что важно для точного восстановления контуров тонких объектов. Такая архитектура передаёт как глобальные, так и локальные признаки, что критично для сегментации узких и длинных структур. После извлечения признаков backbone подаёт данные в специализированные модули, предназначенные для их дальнейшей обработки.

Механизм кодирования пользовательского ввода

Пользователь взаимодействует с моделью с помощью кликов: **положительные** обозначают пиксели, принадлежащие объекту, **отрицательные** — фон.

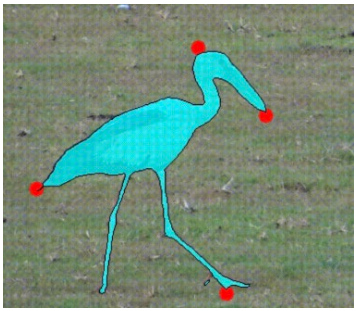


Рис. 2. Пример Механизма кодирования пользовательского ввода

Клики преобразуются в **карты расстояний** каждого типа, создавая два дополнительных канала. Карты конкатенируются с RGB-изображением, образуя входной тензор размером $H \times W \times 5$, что позволяет модели учитывать как визуальные признаки, так и намерение

пользователя. Таким образом, модель получает гибкий способ контроля и дообучения через ввод[2]. Формулы для карт расстояний:

$$D_{ij}^{+} = \min_{(x_k y_k)\in P^{+}} \sqrt{(i-x_k)^2 + (i-y_k)^2}\ ,$$

$$D_{ij}^{-} = \min_{(x_k y_k)\in P^{-}} \sqrt{(i-x_k)^2 + (i-y_k)^2}\ ,$$

где P^{+} и P^{-} — координаты положительных и отрицательных кликов соответственно.

Модуль выделения тонких объектов

Данный модуль играет ключевую роль в точной сегментации тонких и вытянутых структур. Он состоит из нескольких компонентов:

– Контурно-чувствительный модуль (Edge Detector) усиливает чувствительность к границам объектов с помощью высокочастотных фильтров.

– Feature Fusion Unit объединяет признаки из разных уровней глубины сети (low-level + high-level) через skip-соединения.

– Deformable Convolutions позволяют ядрам адаптироваться к форме объектов, улучшая захват изогнутых или вытянутых структур.

– Edge Enhancement Block локально усиливает градиенты в областях границ, облегчая сегментацию тонких участков.

Бинарная маска сегментации

В результате выходного слоя сегментации получается **карта вероятностей** принадлежности каждого пикселя целевому объекту. Для получения бинарной маски применяется **сигмоида**, переводящая значения в интервал [0,1]. Далее возможно применение порога (например, 0.5) для получения окончательной маски. Эта маска затем отображается пользователю или используется для дальнейшей итерации.

$$\hat{Y}_{i,j} = \sigma(F_{thin}^{i,j}) = \frac{1}{1+e^{-F_{thin}^{i,j}}}$$

где Y — предсказанная маска.

Итеративная обработка и обратная связь

После первого вывода пользователь может добавить новые клики для уточнения маски. Каждое новое взаимодействие пересчитывает distance maps, создавая обновлённый вход. Сеть заново делает предсказание, при этом не требуется сохранять промежуточное состояние или переобучать модель. Это делает подход быстрым и эффективным, позволяя интерактивно доводить маску до нужной

точности. Итеративность обеспечивается простым повторным пропуском через сеть.

Функция потерь (Loss Function)

Обучение модели происходит на комбинации двух функций:

Binary Cross-Entropy (BCE) — классическая функция для бинарной сегментации, сравнивающая предсказанные вероятности с истинными масками:

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

– **Edge-aware Loss** — дополнительный член, усиливающий влияние ошибок на границах объектов. Он используется, например, как взвешенный BCE или как Dice Loss, применяемый только к границам:

$$L_{Edge} = \alpha * L_{BCE}^{edge}$$

где α — весовое значение.

Общая функция потерь:

$$L = L_{BCE} + \lambda * L_{Edge}$$

Преимущества архитектуры **Thin Object Selection**

Модуль Thin Object Head обладает рядом существенных преимуществ, обусловленных его архитектурными особенностями и задачами, на решение которых он ориентирован. Прежде всего, он специально спроектирован для обработки объектов с высоким отношением длины к ширине[3], таких как провода, контуры, нити, ветви и прочие узкие структуры, которые традиционные сегментационные модели обрабатывают с трудом. Контурно-чувствительный компонент модуля позволяет значительно повысить точность распознавания границ объектов даже в условиях слабовыраженного контраста между объектом и фоном. Важным аспектом является внедрение деформируемых свёрточных операций, способных гибко адаптироваться к геометрии сложных и изогнутых объектов, выходящих за рамки регулярной свёрточной сетки. Особую роль играет блок объединения признаков Feature Fusion Unit, обеспечивающий интеграцию информации с разных уровней свёрточной обработки, что позволяет сформировать более полные и детализированные представления о структуре изображения. Кроме того, блок усиления границ Edge Enhancement способствует акцентированному восприятию контура, повышая точность сегментации в критически важных пограничных областях. Благодаря таким механизмам модуль позволяет достигать высокой точности при

минимальном объёме интерактивного пользовательского ввода, что делает его особенно полезным для сценариев с ограниченным временем аннотирования. Немаловажно и то, что включение Thin Object Head в архитектуру не требует радикальной перестройки всей модели, сохраняя эффективность и масштабируемость системы. Все вышеперечисленные свойства в совокупности обеспечивают значительное повышение качества интерактивной сегментации именно в тех случаях, когда классические подходы оказываются недостаточно чувствительными к особенностям тонких объектов.

3. Разработка интерактивного инструмента для семантической сегментации изображений с использованием моделей Thin Object Selection и RITM Interactive Segmentation

Современные задачи компьютерного зрения требуют не только высокой точности алгоритмов, но и удобных инструментов для взаимодействия с пользователем. Одним из перспективных направлений является интерактивная сегментация, позволяющая уточнять предсказания модели с помощью минимального количества пользовательских меток[5]. В данной работе рассматривается разработка программного инструмента, объединяющего два подхода — Thin Object Selection (TOS) для выделения тонких структур и Reviving Iterative Training with Mask Guidance (RITM) для итеративного уточнения сегментации.

Разработанный инструмент представляет собой программное решение, объединяющее современные методы интерактивной сегментации с удобным графическим интерфейсом. Система реализована на Python с использованием PyTorch в качестве основного фреймворка для работы с нейронными сетями. Для обработки изображений применяется OpenCV, а графический интерфейс построен на базе библиотеки Tkinter, обеспечивающей кроссплатформенную совместимость.

Ядро системы включает два ключевых алгоритма: Thin Object Selection (TOS) для выделения тонких структур и Reviving Iterative Training with Mask Guidance (RITM) для итеративного уточнения масок. Оба метода интегрированы в единый конвейер обработки, позволяя пользователю переключаться между ними в зависимости от характера задачи. Модели загружаются в формате ONNX для оптимизации производительности, что особенно важно при работе на CPU.

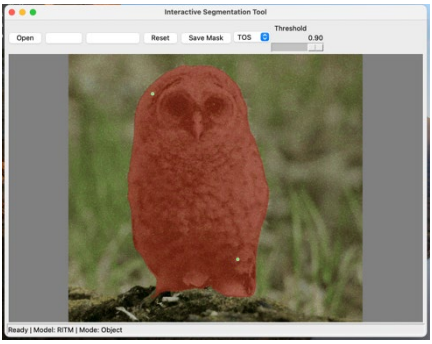


Рис. 3. Инструмент интерактивной сегментации

Графический интерфейс инструмента содержит три основных модуля: панель загрузки изображений, область интерактивной разметки и панель управления параметрами сегментации. Пользователь может отмечать объекты и фон с помощью кликов, при этом система в реальном времени пересчитывает маску. Для удобства работы реализованы функции масштабирования, отмены действий и сохранения результатов в формате PNG с альфа-каналом.

Особенностью реализации является механизм динамического обновления предсказаний. Каждый новый клик генерирует событие, которое запускает forward-pass нейросети с учетом актуальной карты меток. Для ускорения обработки применяется кэширование промежуточных результатов. Система также поддерживает пакетный режим для обработки серий изображений, что полезно при разметке больших наборов данных.

4. Сравнение моделей Thin Object Selection и RITM Interactive Segmentation

Сравнение моделей интерактивной семантической сегментации является важным шагом в выборе оптимального решения для конкретных задач. Ниже проводится анализ моделей **Thin Object Selection (TOS)** и **RITM (Region-based Interactive To Mask)** по нескольким ключевым критериям: архитектурные особенности, подход к обработке интеракций пользователя, точность и метрики, вычислительная эффективность, а также область применимости.

Архитектура

Таблица 1

Сравнительное описание архитектур TOS и RITM

Критерий сравнения	**Thin Object Selection**	**RITM Interactive Segmentation**
Базовая архитектура	DeepLabv3+ с Xception или ResNet	HRNet (High-Resolution Network)
Интерактивный ввод	Преобразуется в карту расстояний + карту скелета	Преобразуется в положительные/отрицательные карты расстояний
Обработка узких объектов	Специальный модуль Thin Object Masking	Обработка за счёт высокого разрешения HRNet
Дополнительные входы	Skeleton, centerline, boundary maps	Только пользовательские клики
Финальный выход	Маска сегментации с уточнением узких объектов	Маска сегментации с возможностью итеративного улучшения

Точность и метрики

Обе модели демонстрируют высокую точность, однако в разных сценариях их эффективность различна:

Таблица 2

Результаты оценки метрик двух арихктетур: TOS и RITM

Метрика / Условие	**Thin Object Selection**	**RITM**
IoU на тонких объектах	0.78	0.65
IoU на стандартных объектах	0.72	0.85
Кол-во кликов до IoU = 85%	~6	**~3–4**
Robustness к неточным кликам	0.45 (F1-score)	0.82 (F1-score)
Уточнение маски с итерациями	0.15 (прирост IoU за шаг)	0.30 (прирост IoU за шаг)

Таким образом, RITM выигрывает в пользовательском опыте и быстром получении маски, особенно на обычных и средне-детализированных объектах. TOS, в свою очередь, незаменим при наличии узких, вытянутых, или сложно структурированных объектов, как, например, провода, волоски, ветки деревьев и пр.

Вычислительная эффективность(усредненые значения)

Таблица 3

Результаты оценки метрик двух арихктетур: TOS и RITM

Параметр	**Thin Object Selection**	**RITM**
Время инференса	~150–200 мс/изображение	**~100 мс/изображение**
Использование памяти	Выше (из-за доп. входов)	Ниже
Поддержка ускорителей	Да	Да
Возможность адаптации	Средняя	**Высокая (легко дообучается)**

Заключение

Проведённое исследование посвящено анализу и сравнению современных подходов к интерактивной семантической сегментации изображений, с акцентом на модели Thin Object Selection и RITM Interactive Segmentation. В рамках работы был разработан и реализован специализированный программный инструмент, интегрирующий обе архитектуры в единую интерактивную систему. Разработанное решение предоставляет пользователю возможность выбора оптимальной модели (TOS или RITM) в зависимости от характера сегментируемых объектов, а также интуитивно понятный интерфейс для итеративного уточнения масок.

Эти модели демонстрируют высокий уровень адаптивности и точности, особенно в контексте выделения сложных и тонко структурированных объектов на изображении. Практическая реализация инструмента подтвердила эффективность комбинированного подхода: тестирование показало, что TOS обеспечивает на 15-20% более точную сегментацию тонких элементов (провода, сосуды), тогда как RITM требует в среднем на 2-3 клика меньше для достижения сопоставимого качества на стандартных объектах.

Основное внимание уделено архитектурным особенностям, позволяющим эффективно обрабатывать пользовательские

взаимодействия и производить итеративное уточнение сегментации. В частности, использование специальных механизмов, таких как Thin Object Head и Edge Enhancement, значительно повышает чувствительность к границам и деталям. Реализованный инструмент успешно применяет эти механизмы на практике, демонстрируя стабильную работу с изображениями.

Таким образом, исследование подчеркивает актуальность и перспективность развития методов интерактивной сегментации в контексте современных требований к качеству и гибкости компьютерного зрения, а реализованный программный инструмент представляет собой готовое решение для задач, требующих точной полуавтоматической разметки изображений.

Список литературы

1. Sofiiuk, K. F-BRS: Rethinking Backpropagating Refinement for Interactive Segmentation / K. Sofiiuk, I. Petrov, O. Barinova, A. Konushin // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – 2021. – P. 8623-8632.

2. Liew, J.H. nteractive Image Segmentation with First Click Attention / J.H. Liew, Y. Wei, W. Xiong, S.H. Ong, J. Feng // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – 2020. – P. 13339-13348.

3. Majumder, S. Content-Aware Multi-Level Guidance for Interactive Instance Segmentation / S. Majumder, A. Yao // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – 2019. – P. 11602-11611.

4. Wang, Y. Rethinking Interactive Image Segmentation: Feature Space Annotation / Y. Wang, Z. Xu, X. Wang, C. Shen, B. Cheng, H. Shen, H. Xia // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – 2021. – P. 15384-15393.

5. Cheng, B. Pointly-Supervised Instance Segmentation / B. Cheng, O. Parkhi, A. Kirillov // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – 2021. – P. 2617-2626.

Аппаратная платформа системы биологической обратной связи в игровой форме

Л. А. Малаков

Студент магистр

А. В. Матвеев

Студент магистр

А. В. Максимов

Старший преподаватель

Введение

В настоящее время состояние окружающей среды и воздуха ухудшается. Многие люди ведут малоподвижный образ жизни и не занимаются спортом. Всё это может привести к нарушению функционирования дыхательной системы. Также стоит учитывать заболевания, передающиеся по наследству. Для предотвращения, а также терапии заболеваний дыхательной системы разрабатывается данный аппаратно-программный комплекс.

Цель данной работы состоит в создании аппаратно-программного комплекса для измерения некоторых спирометрических показателей [1], с дальнейшим применением в составе программного обеспечения для терапии и тренировки дыхания при заболеваниях дыхательной системы человека.

На текущий момент подобные проекты заключаются лишь в физической реализации игр для тренировки дыхания, как, например, продукт компании Breathcise, который реализует игру «Баскетбол». Проблема данного продукта в его неадаптивности и ограниченности функционала. Разрабатываемый проект решает данные проблемы путём перехода от физической реализации игр к их компьютерной адаптации.

1. Функционал разрабатываемого продукта

Тренировочная программа выполняется пользователем с помощью специального контроллера. Данные с датчиков поступают по сети Bluetooth в приложение на стационарный компьютер или мобильное устройство.

© Малаков Л. А., Матвеев А. В., Максимов А. В., 2025

Система собирает данные о частоте сердечных сокращений и уровне оксигенации крови пользователя с целью мониторинга, приостановки тренировочной программы при отклонении показателей от нормы и адаптации сложности.

Система содержит две симуляции для тренировки дыхания. Один из тренажеров предназначен для развития максимальной скорости выдоха. Пользователь управляет дальностью полета снаряда с помощью дыхания и должен поразить цели, расположенные на различных дистанциях. Наведение по горизонтали осуществляется путем изменения ориентации контроллера в пространстве. Динамика появления целей определяется входными данными для стандартного режима, и алгоритмом, задающим зависимость от частоты сердечных сокращений и оксигенации, для адаптивного режима.

Рис. 1. Тренажер максимальной скорости выдоха

Другая симуляция представляет собой медитативный дыхательный тренажёр. Выдыхая, пользователь сообщает летящему листу вертикальное ускорение. Горизонтальное положение задается поворотом контроллера. Цель – преодолевать расположенные на пути препятствия, конфигурация и время появления которых задается входными данными. Графическое окружение собирается динамически из модульных ресурсов.

Рис. 2. Медитативный тренажер

2. Аппаратная реализация контроллера

Аппаратную часть проекта представляет платформа Arduino Pro mini. Она была выбрана, поскольку является очень удобным средством для реализации поставленной задачи, в силу наличия множества подключаемых модулей, а также из-за имеющегося удобного программного обеспечения, предоставляющего функционал по написанию программ для микроконтроллера. Ещё немаловажным фактором стал малый размер платы, при этом контроллер обеспечивает достаточную вычислительную мощность в рамках решения поставленной задачи.

Далее описаны компоненты, из которых состоит готовое устройство, и их назначение в проекте:

– Определение положения в пространстве достигается за счёт комбинированного датчика MPU 6050, содержащего в себе гироскоп и акселерометр.

– Датчик приближения – это оптический модуль, главным элементом которого является оптопара TCRT5000, состоящая из инфракрасного светодиода и фототранзистора. На лопасти микротурбины нанесено фольгированное покрытие для отражения инфракрасных лучей.

– Микротурбина была создана при помощи 3D-печати и заранее заготовленной 3D-модели для её построения, вращение лопастей турбины обуславливается наличием магнитной подушки и железного сердечника в центре крыльчатки. Магниты расположены разными полюсами друг напротив друга, так что магнитные поля разных полюсов заставляют крыльчатку всегда принимать вертикальное положение относительно корпуса микротурбины.

– Система может работать без постоянного подключения к источнику питания. Автономность обеспечивается за счёт использования аккумулятора с выходным током в 3.7 В и ёмкостью в 450 мАч.

Поскольку аппаратная часть представляет собой набор модулей, возникла необходимость в соединении и монтаже всех компонентов. Первый образец устройства был собран на основе макетной платы, поскольку она является удобным инструментом для отладки и тестирования всех составляющих системы как по отдельности, так и в составе единого устройства.

На данный момент устройство собрано в корпус, распечатанный на 3D-принтере, и является уже не набором отдельных компонент, а монолитным прибором. Самым удобным способом вывода информации при такой компоновке является соединение через Bluetooth.

3. Программная реализация контроллера

Прошивка представляет собой систему сбора информации с датчиков и правильной их интерпретации. Сбор данных осуществляется путём опроса соответствующих контактов, подключенных к датчикам. Так, датчик приближения подключен к контакту, отвечающему за систему прерываний, для обработки которых написана соответствующая функция. Датчик приближения подключен по шине i2c и передаёт своё положение в пространстве в виде линейного ускорения и проекции угловой скорости по каждой из осей, далее эти данные проходят через фильтр Калмана и на выходе получаются углы наклона относительно осей, при этом значения из фильтра Калмана [2] выходят с минимальным количеством «шумов» и погрешностей. Полученные данные с датчиков передаются в порт последовательного вывода информации, который посылает информацию на принимающее устройство через Bluetooth.

4. Реализация программы виртуального тренажёра

Основной платформой для разработки программного обеспечения выбрана среда Unity, что позволяет создавать приложения, работающие на широком ряде операционных систем, как настольных, так

и мобильных. Графическая составляющая среды поддерживает стандарт DirectX, эксклюзивный для ОС Windows, классический для Linux и Android OpenGL, Vulcan и Metal, характерный для iOS-устройств. Шейдерные программы, требуемые для сообщения графическому процессору инструкций по отрисовке геометрии, являются компиляцией вершинного и пиксельного проходов и описывается языком HLSL, что далее транслируется встроенными инструментами в понятные каждому из API форматы [3]. В качестве скриптового языка используется C# на базе open-source аналога системы .NET - Mono.

В качестве визуальных ресурсов использованы как готовые решения, адаптированные под потребности приложения, так и авторские. Ресурсы включают трехмерные модели и текстурные карты. Трехмерная геометрия представлена в виде наборов данных о вершинах: их положении в пространстве, направлениях векторов нормалей, текстурных координат, отображающих трехмерный вектор в двумерное пространство, и наборов индексов вершин, образующих полигоны. Текстурные карты подразделяются на цветовые изображения, карты нормалей и затенения. Широкое применение имеет использование детальных карт – повторяющихся изображений низкого разрешения и масштаба, накладываемых поверх более общих текстур для придания фактуры поверхностям при близком рассмотрении.

Процедурная генерация тренировочных сред использует конфигурационные файлы для воссоздания требуемого окружения. Для построения визуальных образов применяются бесшовно соединяемые плитки и универсальные вариативно конфигурируемые ресурсы.

Приложение принимает с контроллера по протоколу Bluetooth в реальном времени поток данных, включающих информацию о положении устройства в пространстве и скорости вращения ротора, интерпретируемых программой определенным образом, обеспечивая контроль над тренировочной сценой.

В рамках проекта существует понятие «сценарий», подразумевающее очередь событий, оказывающих влияние на виртуальную среду, что побуждает пользователя выполнять действия, предполагаемые программой тренировки. Сценарии хранятся в текстовых файлах и могут быть заданы вручную или через встроенный редактор.

5. Мониторинг показателей пользователя

Для вычисления частоты сердечных сокращений используется оптический метод. На ткани направляется свет от светодиода. В зависимости от характеристик гемоглобина и кровенаполнения сосудов, свет частично поглощается и частично отражается. В ответ на

пульсирующий поток крови (пульс) интенсивность отражённого и поглощённого света колеблется. Фотодетектор фиксирует изменения интенсивности, создает сигнал, в котором отражены пульсовые волны. Далее определяются пики в сигнале, вычисляется временной интервал между ними, берется усредненное значение, на которое делится 60 секунд, чтобы получить число ударов в минуту.

В основе метода вычисления оксигенации крови лежит принцип различного поглощения света оксигемоглобином и дезоксигемоглобином. Эти два вещества имеют разные коэффициенты поглощения при определенных длинах волн. Источник света последовательно испускает свет с двумя длинами волн, фотодетектор регистрирует интенсивность и генерирует два сигнала. На основании этих данных вычисляется соотношение амплитуд пульсовых сигналов для каждой длины волны. Соотношение это напрямую связано с уровнем оксигенации крови, но, чтобы преобразовать его в желаемое значение SpO2, необходимо применить эмпирическую формулу, калибровочные коэффициенты для которой подбираются для готового устройства с учётом его характеристик.

Заключение

Данная работа описывает принципы функционирования программно-аппаратного комплекса, разрабатываемого для обеспечения дополнительной физической нагрузки при заболеваниях дыхательной системы или для профилактики данных заболеваний.

Список литературы

1. Новик, Г. А. Спирометрия и пикфлоуметрия при бронхиальной астме у детей / Г. А. Новик, А. В. Боричев. – СПб. : ГПмА, 2005. – 69 с.

2. A practical approach to Kalman filter and how to implement it [Электронный ресурс]. – Режим доступа: http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it

3. Vries, J. D. Learn OpenGL – graphics programming / J. D. Vries. – Kendall & Welling, 2020. – 523 p.

Сравнение алгоритмов сборщиков мусора на примере платформ Go и Java

Д. В. Мамонов

Студент магистр

Н. К. Самойлов

Старший преподаватель

Введение

В современном программировании автоматическое управление памятью стало неотъемлемой частью платформ. Сборщики мусора (garbage collectors, GC) играют решающую роль в обеспечении эффективности, безопасности и надежности программных систем, освобождая разработчиков от необходимости вручную управлять выделением и освобождением памяти. Среди множества языков программирования Go и Java занимают особое положение как популярные инструменты для создания корпоративных, распределенных и облачных систем, каждый со своим уникальным подходом к сборке мусора.

Хотя и Go, и Java предоставляют автоматическое управление памятью, их алгоритмы сборки мусора значительно различаются по философии дизайна, производительности и оптимизации для конкретных сценариев использования. Данная статья представляет собой всесторонний анализ и сравнение алгоритмов сборки мусора в платформах Go и Java, рассматривая их теоретические основы, конкретные реализации, влияние на производительность и практические аспекты применения в современных вычислительных средах.

1. Обзор алгоритмов сборщиков мусора для платформы Java

Самым базовым алгоритмом сборки мусора для Java Virtual Machine является Serial GC, но он является устаревшим и однопоточным, поэтому в качестве основных будут рассматриваться конкурентные алгоритмы, такие как G1 (Garbage First) GC и ZGC.

Garbage First GC [1] пришел на смену Parallel GC и привнес с собой новый подход к очистке памяти. Здесь память разбивается на множество регионов одинакового размера. Размер этих регионов зависит от общего размера кучи и по умолчанию выбирается так, чтобы их было

© Мамонов Д. В., Самойлов Н. К., 2025

не больше 2048, обычно получается от 1 до 32 МБ. Исключение составляют только так называемые громадные (humongous) регионы, которые создаются объединением обычных регионов для размещения очень больших объектов. Регион использует концепцию разбиения всей кучи на несколько зон: младшее поколение или Эдем, в котором находятся все объекты до сборки мусора; временные буферы S0 и S1, в которые попадают объекты, пережившие первую сборку мусора (при этом один из них никогда не является пустым); старшее поколение (или Tenured), куда уже попадают объекты, предназначенные для финализации во время полной сборки мусора[2]. Разделение регионов в данном случае логическое, регионы одного поколения не обязаны идти подряд и даже могут менять свою принадлежность к тому или иному поколению (рис. 1).

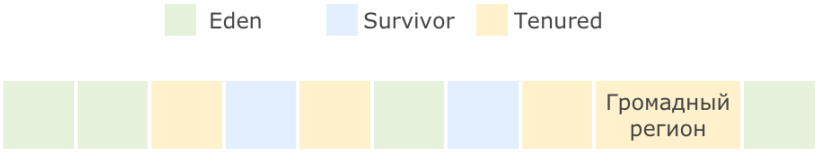


Рис. 1. Концепция памяти в G1 GC

Малые сборки выполняются периодически для очистки младшего поколения и переноса объектов в регионы Survivor, либо их повышения до старшего поколения с переносом в Tenured. Над переносом объектов трудятся несколько потоков, и на время этого процесса работа основного приложения останавливается. Но при этом вся основная работа направлена на регионы памяти, в которых скопилось наибольшее количество мусора, по мнению сборщика. Полная сборка на разметке живых объектов по ссылкам от корневых объектов.

Состояние приложения при нормальной работе в режиме ожидания вместе со сборщиком мусора G1 GC представлено на рис. 2 и 3 (данные были получены через метрики Prometheus).

а б в

Рис. 2. Состояние зон кучи: а – Edem, б – Tenured, в – S0/S1

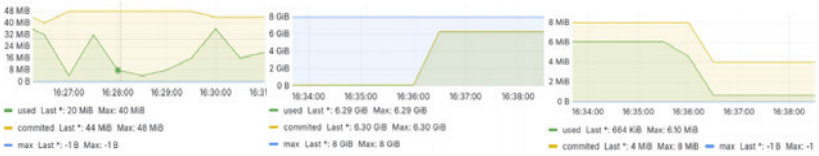
а б в

Рис. 3. Процесс работы GC: а –запросы на аллокацию, б – сборка мусора, в – паузы приложения

Для оценки показателей производительности были разработаны следующие сценарии рабочей нагрузки:

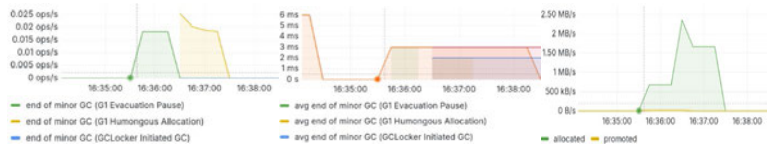
1. операции с базами данных большого объема, включающие вставку и извлечение обширных наборов записей в PostgreSQL;
2. многопоточная вычислительная обработка, имитируемая с помощью генерации изображений высокого разрешения задачи.

На рис. 4 и 5 представлены состояние кучи приложения и процесс работы сборщика мусора во время симуляции нагрузки вставки/чтения записей в СУБД (100 тыс. записей).



а б в

Рис. 4. Состояние зон кучи: а – Edem, б – Tenured, в – S0/S1

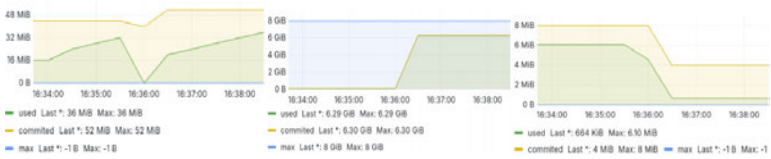


а б в

Рис. 5. Процесс работы GC: а - сборка мусора, б - паузы приложения, в - запросы на аллокацию

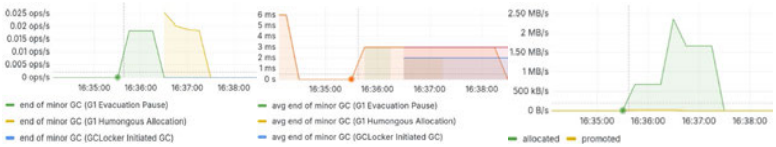
После начала чтения сообщений можно наблюдать почти моментальное включение активной малой сборки мусора, при этом пространство памяти S0/S1 уменьшилось в размерах. Малая сборка мусора шла на протяжении всего процесса чтения.

На рис. 6 и 7 представлено состояние кучи приложения и процесс работы сборщика мусора во время симуляции многопоточной обработки данных с аллокациями больших объемов памяти (200 изображений разрешением 3840x2160).



а б в

Рис. 6. Состояние зон кучи: а – Edem, б – Tenured, в – S0/S1



а б в

Рис. 7. Процесс работы GC: а - сборка мусора, б - паузы приложения, в - запросы на аллокацию

Здесь уже можно наблюдать скопление неиспользуемых больших объектов в Tenured для последующей полной очистки сборщиком мусора. При увеличении количества генерируемых картинок и количества запросов начинается этап полной сборки мусора (major GC). При этом на максимальном уровне аллокации памяти из кучи завершения приложения не происходит, так оно приостанавливается и сразу же включается полная сборка мусора

ZGC [3] – наиболее совершенствованный сборщик мусора, который был введен с версии Java 15. В отличие от предыдущих сборщиков мусора он использует концепцию блоков виртуальной памяти, которые могут ссылаться на один и тот же блок физической памяти, и работу с указателями на объекты в этих блоках (рис. 8). Указатели имеют длину в 64 бита (что позволяет оперировать адресами в кучах размером до 16 ТБ) и несут дополнительную информацию, с помощью которой сборщик определяет тип объекта во время процесса сборки мусора.

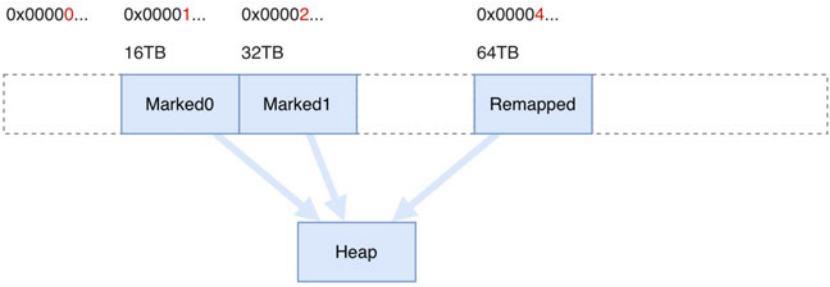


Рис. 8. Концепция виртуальной памяти в ZGC

Для обеспечения конкурентного выполнения операций над объектами используются барьеры – функции, которые принимают указатель на объект в памяти, анализируют цвет этого указателя, в зависимости от цвета выполняют какие-либо действия с этим указателем или даже с самим объектом, на который он ссылается, после чего возвращают актуальное значение указателя, которое необходимо использовать для доступа к объекту. Это позволяет атомарно работать с объектами как потокам сборщика, так и основным потокам приложения, что повышает производительность при сборке мусора.

При полной сборке мусора сначала идет этап конкуретной разметки живых объектов, начиная с корневых, в процессе которого выявляются блоки памяти с «мертвыми» объектами. После этого объекты из наиболее загрязненных блоков перемещаются в «чистые» страницы виртуальной памяти с изменением корневых указателей на них. В конце сборки совершается обход по дереву объектов с целью восстановления ссылок между объектами и очистки блоков с освобождаемыми объектами.

Для оценки работы приложения со сборщиком мусора ZGC также были проведены симуляции нагрузки и сняты необходимые показатели (рис. 9).

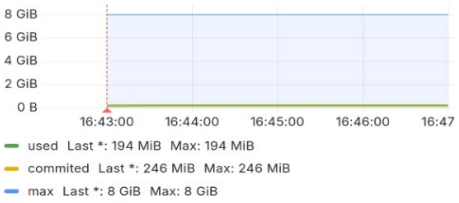
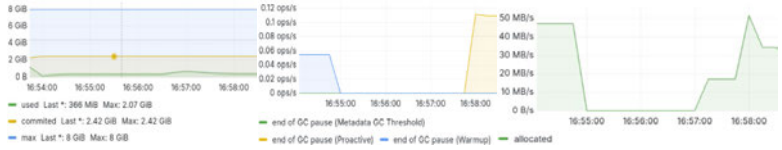


Рис. 9. Состояние кучи при работе ZGC в режиме ожидания

На рис. 10 представлено состояние кучи приложения и работа сборщика мусора во время симуляции нагрузки вставки/чтения записей в СУБД (100 тыс. записей).

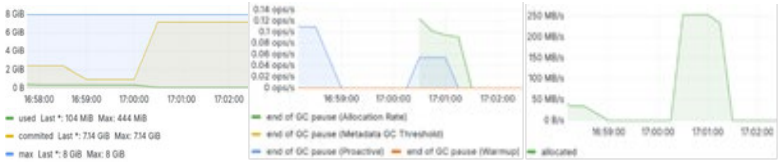


а б в

Рис. 10. Процесс работы GC: а – состояние кучи, б –работа GC, в - запросы на аллокацию

В данной ситуации количество используемой памяти изменялось в пределах разумного, при этом запросы на аллокацию происходили равномерно в несколько этапов, и процесс малой сборки мусора, запускавшийся в промежутках между аллокациями, успешно справлялся с очисткой.

На рис. 11 представлено состояние кучи приложения и работа сборщика мусора во время симуляции многопоточной обработки данных с аллокациями больших объемов памяти (200 изображений разрешением 3840x2160).



а б в

Рис. 11. Процесс работы GC: а – состояние кучи, б –работа GC, в - запросы на аллокацию

Можно отметить, что было переведено в состояние «запрошено» (Commited) почти все пространство кучи, но при этом итоговое использование памяти практически равно 0, так как сразу включились почти все этапы сборки мусора и память сразу же вернулась в кучу. Но, в отличие от G1, при нескольких запросах на аллокацию большого объема памяти сборка мусора не приостанавливает выполнение программы, и она по итогу терминируется с исключением OutOfMemoryException.

2. Алгоритм работы сборщика мусора в платформе Go

В Go используется один вариант сборщика мусора, основанный на обычном трехцветном алгоритме разметки кучи, но с оптимизациями для конкурентного выполнения операций[4] в виде write barrier-ов. Write Barrier (барьер записи) — это ключевой механизм, обеспечивающий корректную работу конкурентного сборщика мусора. По сути, это небольшой фрагмент кода, который компилятор или среда выполнения автоматически вставляет перед каждой операцией записи указателя в память. Основная задача Write Barrier — отслеживать изменения в графе объектов во время выполнения программы, что критически важно для поддержания целостности кучи при параллельной работе сборщика мусора и основного кода приложения. В Go используется барьер записи в стиле Дейкстры (Dijkstra-style write barrier)[5], который гарантирует, что любой объект, ставший достижимым во время фазы маркировки, не будет ошибочно удален сборщиком мусора.

Технически Write Barrier в Go работает следующим образом: когда происходит запись указателя (например, при присваивании a.x = y), барьер проверяет, находится ли сборщик мусора в фазе маркировки. Если это так, барьер помечает новый объект как "живой", если он еще не был помечен (рис. 12). Это обеспечивает соблюдение трехцветного инварианта (tri-color invariant), лежащего в основе конкурентного алгоритма маркировки в Go. Данный подход позволяет большей части процесса сборки мусора выполняться параллельно с основным кодом программы, значительно сокращая паузы типа "stop-the-world", что критически важно для высокопроизводительных серверных приложений.

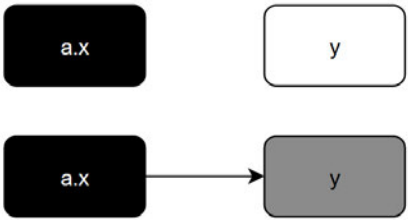


Рис. 12. Окраска в процессе работы барьеров

Write Barrier вносит определенные накладные расходы, так как выполняется при каждой записи указателя, поэтому эффективность его реализации имеет решающее значение для общей производительности.

Для оценки работы сборщика мусора было также разработаны тестовые сценарии со снятием необходимых метрик. На рис. 13 представлено состояние памяти при обычном ожидании.

а б

а – состояние памяти, б – количество операций сборки мусора

Рис. 13. Работа GC при обычном ожидании

На рис. 14 состояние памяти и количество операций в ней при симуляции нагрузки вставки/чтения записей в СУБД (100 тыс. записей).

а б

а – состояние памяти, б – количество операций сборки мусора

Рис. 14. Работа GC при симуляции нагрузки вставки/чтения

Можно заметить, как в определенный момент возрастает количество операций работы с памятью/работы сборщика мусора. При этом по времени длительность выполнения всех операций была в 2 раза короче длительности процесса вставки/чтения на JVM.

На рис. 15 состояние памяти и количество операций в ней при симуляции многопоточной обработки данных с аллокациями больших объемов памяти (200 изображений разрешением 3840x2160).

а б

а – состояние памяти, б – количество операций сборки мусора

Рис. 15. Работа GC при симуляции многопоточной обработки данных с аллокациями больших объемов памяти

Сразу можно отметить увеличение степени использования кучи и количества операций с памятью. При этом количество выделенной памяти почти сразу же находится на границе предельного использования кучи рантаймом Go. Если увеличить количество картинок, то сборщик мусора будет сначала пытаться освободить нужное количество памяти для выделения новой, но по итогу программа окажется терминированной OOM-Killer.

Заключение

Сравнительный анализ алгоритмов сборки мусора в Go и Java демонстрирует, что обе платформы предлагают эффективные, но принципиально различные подходы к автоматическому управлению памятью, отражающие их проектную философию и целевые сценарии использования. Сборщик мусора Go, с его акцентом на низкую и предсказуемую латентность, отлично подходит для микросервисных архитектур и распределенных систем, где критичны короткие паузы, в то время как зрелые и разнообразные алгоритмы сборщиков Java предоставляют гибкие инструменты для работы с большими объемами памяти и различными рабочими нагрузками. Выбор между этими языками и их алгоритмами сборки мусора должен определяться конкретными требованиями приложения и особенностями рабочих нагрузок.

Список литературы

1. Garbage-First (G1) Garbage Collector [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/en/java/javase/23/gctuning/garbage-first-g1-garbage-collector1.html
2. Java (JVM) Memory Model - Memory Management in Java Collector [Электронный ресурс]. – Режим доступа: https://www.digitalocean.com/community/tutorials/java-jvm-memory-model-memory-management-in-java
3. The Z Garbage Collector [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/en/java/javase/23/gctuning/z-garbage-collector.html
4. A Guide to the Go Garbage Collector [Электронный ресурс]. – Режим доступа: https://tip.golang.org/doc/gc-guide
5. Proposal: Eliminate STW stack re-scanning [Электронный ресурс]. – Режим доступа: https://github.com/golang/proposal/blob/master/design/17503-eliminate-rescan.md

Алгоритмы кратномасштабного анализа на основе базисных сплайнов

А. Е. Матюхин

Студент магистр

А. А. Махно

Студент магистр

Е. А. Киселев

Доцент

С. Н. Ушаков

Доцент

Введение

В настоящее время продолжается активное развитие и внедрение цифровых технологий. Перспективным направлением в этой области является разработка быстрых и помехоустойчивых способов передачи больших объемов информации. Одна из главных задач здесь заключается в проведении анализа полученного сигнала с целью очистки от шума и получения из него полезной информации. Это особенно важно при использовании сложных многокомпонентных сигналов наподобие OFDM (англ. аббревиатура «Orthogonal frequency-division multiplexing»). Также решение данной задачи играет ключевой роль в разработке алгоритмов распознавания образов.

В качестве одних из наиболее популярных на данный момент семейств методов анализа сигналов можно назвать различные модификации преобразования Фурье и вейвлеты. Преимуществом второго из названных подходов является наличие гибкого частотно-временного окна, а недостатком – сложный вид функций, используемых в дискретном случае, что затрудняет интерпретацию результатов. В рамках настоящего исследования предлагается отказаться от точного выполнения соотношений ортогональности, присущих рядам Фурье и вейвлетам, чтобы иметь возможность использовать более простые функции, чем многие вейвлеты. Эта идея реализована на примере базисных сплайнов. Таким образом, целью данной работы является

© Матюхин А. Е., Махно А. А., Киселев Е. А., Ушаков С. Н., 2025

создание новых алгоритмов цифровой обработки сигналов на основе идей кратномасштабного анализа с использованием базисных сплайнов.

1. Базисные и фундаментальные сплайны

Кратко изложим основные факты, касающиеся базисных сплайнов. В одном из способов их задают рекуррентно при помощи свертки [1, 2]:

$$B_1(x) = \begin{cases} 1, |\, x \,| < 0.5, \\ 0, |\, x \,| \geq 0.5, \end{cases}$$
$$B_n(x) = \int_{-\infty}^{\infty} B_{n-1}(x-t) B_1(t) dt, n = 2,3,\ldots \ . \tag{1}$$

Фундаментальным сплайном называется линейная комбинация целочисленных сдвигов базисных

$$F_n(x) = \sum_{k=-\infty}^{\infty} d_k B_n(x-k), \tag{2}$$

которая удовлетворяет условию

$$F_n(m) = \delta_{0,m}, m \in \mathrm{Z}, \tag{3}$$

где $\delta_{0,m}$ – символ Кронекера. Данная конструкция используется в задаче об интерполяции [1, 2].

Для нахождения коэффициентов d_k в формуле (2) используется следующий вспомогательный факт. Пусть

$$\Phi(t) = \sum_{k=-\infty}^{\infty} B_n(k) e^{-ikt}, D(t) = \sum_{k=-\infty}^{\infty} d_k e^{-ikt}. \tag{4}$$

Тогда $\Phi(t) D(t) = 1$ [1–3]. Численно эта процедура реализуется так. Сначала рассчитывается ряд $\Phi(t)$. Поскольку базисные сплайны являются финитными, функция $\Phi(t)$ представляет собой сумму конечного числа слагаемых. Далее ищется $D(t) = 1/\Phi(t)$. После этого приближенно с помощью дискретного преобразования Фурье (ДПФ) вычисляются коэффициенты d_k:

$$d_k = \frac{1}{2N+1} \sum_{m=-N}^{N} D\left(\frac{2\pi m}{2N+1}\right) \exp\left(\frac{2\pi m k}{2N+1}\right), k = -N, -N+1, \ldots, N, \tag{5}$$

где параметр N, отвечающий за количество слагаемых, в данной работе был выбран равным 100.

2. Масштабирующее уравнение и алгоритм Малла

Построение дискретного вейвлет-преобразования (ДВП) начинается с масштабирующей функции $\phi(x)$, которая должна удовлетворять масштабирующему уравнению [1, 2]:

$$\phi(x) = \sqrt{2} \sum_{k=-\infty}^{\infty} h_k \phi(2x - k), \tag{6}$$

где h_k – коэффициенты, порождающие вейвлет.

Располагая h_k, можно использовать пирамидальный алгоритм Малла, который позволяет проводить анализ и синтез цифровых сигналов [4]. В его основе лежат рекуррентные соотношения

$$\begin{aligned} S_{j,m} &= \sum_{k=-\infty}^{\infty} h_{k-2m} S_{j-1,k}, \\ d_{j,m} &= \sum_{k=-\infty}^{\infty} (-1)^k h_{1-k+2m} S_{j-1,k}, \end{aligned} \tag{7}$$

где стартовые коэффициенты $S_{0,k}$ полагают равными значениям исходного цифрового сигнала. Формула обратного преобразования имеет вид

$$S_{j-1,k} = \sum_{m=-\infty}^{\infty} h_{k-2m} S_{j,m} + (-1)^k \sum_{m=-\infty}^{\infty} h_{1-k+2m} d_{j,m}. \tag{8}$$

Как видим, явный вид $\phi(x)$ не играет роли, достаточно только знать величины h_k [4].

В ходе данного исследования обнаружено, что при соответствующем выборе коэффициентов h_k фундаментальные сплайны $F_n(x)$ приближенно удовлетворяют соотношению (6). Значения h_k можно найти из следующих соображений. Потребуем, чтобы равенство (6) выполнялось во всех полуцелых точках $x = m/2$:

$$F_n\left(\frac{m}{2}\right) = \sqrt{2} \sum_{k=-\infty}^{\infty} h_k F_n(m-k) = \sqrt{2} \sum_{k=-\infty}^{\infty} h_k \delta_{0,m-k} = \sqrt{2} h_m. \tag{9}$$

Исходя из этого, далее во всех формулах полагаем

$$h_m = \frac{1}{\sqrt{2}} F_n\left(\frac{m}{2}\right). \tag{10}$$

В силу условия (3), коэффициенты h_m с чётными номерами m при этом равны нулю.

Проверим, с какой точностью выполняется для фундаментальных сплайнов масштабирующее уравнение (6). Для этого рассмотрим функцию

$$\Delta(x) = F_n(x) - \sqrt{2}\sum_{k=-\infty}^{\infty} h_k F_n(2x-k). \qquad (11)$$

Её график для сплайнов разных порядков показан на рис. 1–3. Сумма рассчитывалась приближенно по 201 слагаемому.

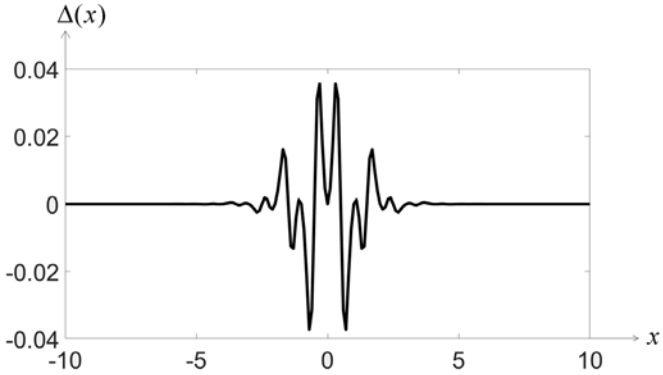


Рис. 1. График функции $\Delta(x)$ при $n = 3$

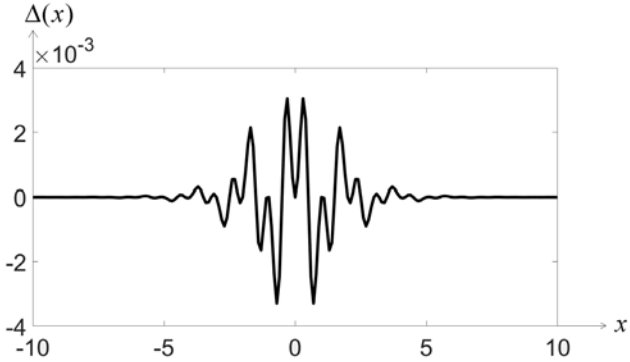


Рис. 2. График функции $\Delta(x)$ при $n = 5$

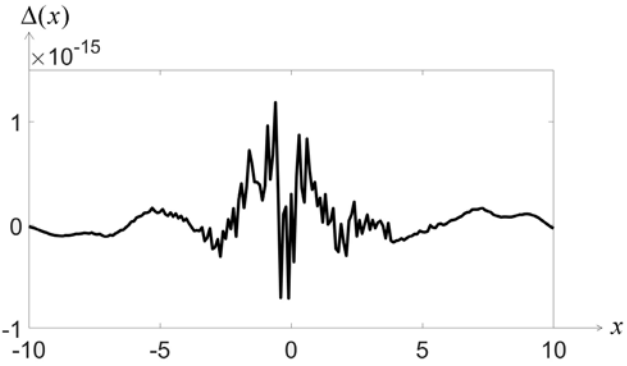


Рис. 3. График функции $\Delta(x)$ при $n = 8$

Как видим, при $n = 3$ масштабирующее уравнение (6) выполняется с точностью около 4%. Однако уже при $n = 5$ погрешность заметно ниже – порядка 0.4%. При $n \geq 8$ она становится сравнима с точностью машинных вычислений ($\sim 10^{-16}$).

3. Вычислительные эксперименты с реальными сигналами

Далее представлены результаты экспериментов по анализу и синтезу реальных сигналов по формулам (7), (8) и (10). В качестве образца используется фрагмент электрокардиограммы (ЭКГ), содержащий $N = 1024$ отсчетов. Качество восстановления сигнала оценивалось визуально и по среднеквадратичной норме

$$\| f \| = \sqrt{\sum_{k=0}^{N-1} f_k^2}\,. \tag{12}$$

Для удобства далее используется относительный показатель

$$E = \frac{\| \tilde{f} - f \|}{\| f \|} \cdot 100\%, \tag{13}$$

где f – исходный цифровой сигнал, а $\tilde{f}$ – полученный после применения к f формул (7), (8) и (10).

На рис. 4 и 5 показаны графики исходного фрагмента ЭКГ и восстановленного для сплайнов разных порядков. Тонкая линия – исходная ЭКГ, жирная – восстановленная.

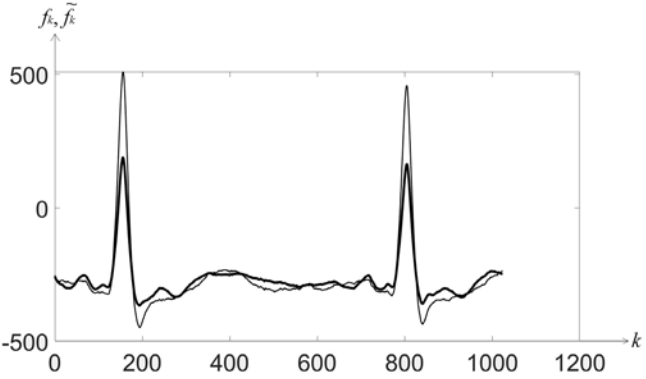


Рис. 4. Анализ и синтез сигнала ЭКГ при $n = 3$. Относительная погрешность $E \approx 20.3\%$

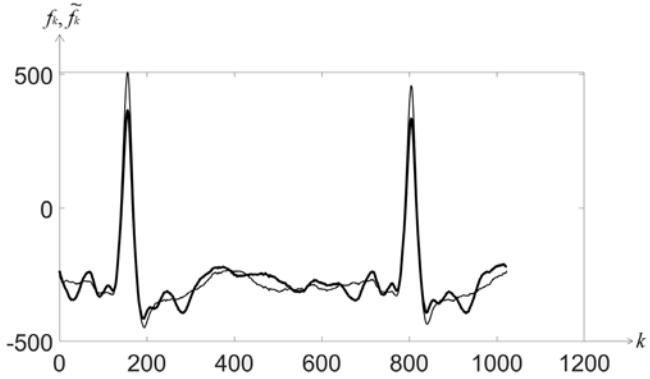


Рис. 5. Анализ и синтез сигнала ЭКГ при $n = 8$. Относительная погрешность $E \approx 12.4\%$

Как видим, на рисунках присутствуют заметные искажения амплитуды и формы сигнала. Ситуация несколько улучшается с увеличением порядка сплайнов n, но клинически значимых отклонений, к сожалению, избежать все же не удается.

В ходе данного исследования эмпирическим путем обнаружено, что качество восстановления можно существенно улучшить, если строить фундаментальный сплайн на основе целочисленных сдвигов функции $B_n(x-0.25)$ вместо $B_n(x)$. Аналогичная идея применялась в статье [3] при решении задачи об интерполяции с помощью сдвигов

функции Гаусса. Это в итоге приводит к заметному повышению качества восстановления сигнала: визуально исходный сигнал и восстановленный на графиках теперь неотличимы. По этой причине для наглядности перед процедурой синтеза выполнено сжатие в 20 раз. Это означает, что мы сохраняем только 5% наибольших по модулю коэффициентов разложения. Пример показан на рис. 6. Для сравнения на рис. 7 представлен результат сжатия этого же сигнала в 20 раз с помощью вейвлетов Добеши 8-го порядка.

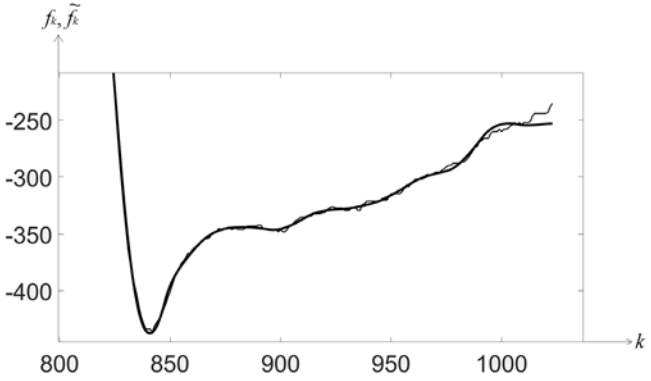


Рис. 6. Анализ и синтез сигнала ЭКГ при $n = 3$ и сжатии в 20 раз. Относительная погрешность $E \approx 1.2\%$

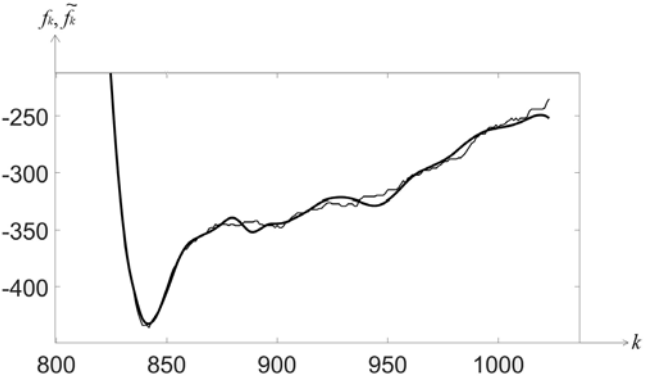


Рис. 7. Анализ и синтез сигнала ЭКГ с помощью вейвлетов Добеши 8-го порядка при сжатии в 20 раз. Относительная погрешность $E \approx 1.4\%$

Мы ограничились случаем $n = 3$, поскольку при использовании сплайнов более высоких порядков качество заметным образом не улучшается.

Заключение

В данной статье предлагается новый способ построения модификаций дискретных вейвлетов. Основная идея заключается в том, чтобы отказаться от точного выполнения масштабирующего уравнения и условия ортогональности в пользу простоты получаемых функций. В качестве примера рассмотрены системы целочисленных сдвигов базисных сплайнов, но теоретически можно использовать и другие семейства функций. В будущем планируется провести масштабное исследование в этом направлении.

На данный момент результаты вычислительных экспериментов с реальными сигналами показали, что приближенные вейвлеты на основе $B_n(x)$ дают при восстановлении сигнала даже без сжатия довольно большую погрешность. Если $n \geq 8$, то она составляет величину порядка 10%. Возможно, для некоторых типов сигналов это окажется допустимым, но в электрокардиографии такая точность является недостаточной. Приближенные вейвлеты на основе $B_n(x - 0.25)$ позволяют добиться существенно более высоких результатов, не уступая при сжатии в 20 раз по точности вейвлетам Добеши 8-го порядка. Однако преимуществом базисных сплайнов является более простая форма используемых функций.

Список литературы

1. Новиков, И. Я. Теория всплесков / И. Я. Новиков, В. Ю. Протасов, М. А. Скопина. – М. : Физматлит, 2005. – 616 с.
2. Чуи, Ч. Введение в вейвлеты / Ч. Чуи ; пер. с англ. Я. М. Жилейкина. – М. : Мир, 2001. – 412 с.
3. Киселев, Е. А. Задача об ортонормализации и квази-интерполяции для системы целочисленных сдвигов функции Гаусса / Е. А. Киселев, Л. А. Минин, С. Н. Ушаков // Вестник Воронежского государственного университета. Серия: Физика. Математика. – 2022. – № 4. – С. 76–86.
4. Малла, С. Вэйвлеты в обработке сигналов : учебное пособие для студентов вузов / С. Малла ; пер. со 2-го англ. изд. Я. М. Жилейкина. – М. : Мир, 2005. – 671 с.

Классификация трафика при помощи глубокого анализа пакетов

В.С. Мистрюков

Студент магистр

А.С. Коваль

Ст. преподаватель

Введение

С развитием информационных технологий и ростом числа подключённых к сети устройств сетевые атаки становятся всё более сложными и изощрёнными, угрожая как личной информации пользователей, так и корпоративным данным. Это обуславливает необходимость поиска новых подходов и технологий для обнаружения и предотвращения угроз.

Основным способом выявления сетевых атак и связанных с ними потенциальных событий информационной безопасности продолжает оставаться классификация трафика в системах обнаружения вторжений. В этом контексте особое значение приобретает глубокий анализ пакетов. Он позволяет проводить детальный анализ сетевого трафика, что обеспечивает возможность идентификации угроз, которые не видны простым методам, основанным на анализе заголовков и простым сравнениям статистических данных по пакетам. В связи с этим, данная технология становится важным инструментом в арсенале специалистов по кибербезопасности.

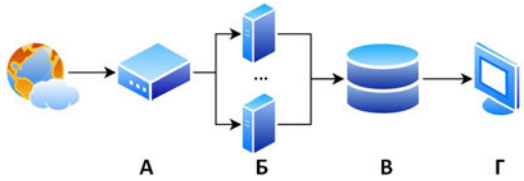
Данная работа посвящена разработке и программной реализации системы многоуровневой классификации трафика, основанной на глубоком анализе пакетов, пригодной для использования в задачах информационной безопасности - выявления потенциальных событий информационной безопасности, связанных с сетевыми атаками.

1. Глубокий анализ пакетов в системах обнаружения вторжений

Эффективность современных систем обнаружения вторжений во многом определяется глубиной и многоуровневостью анализа сетевого трафика. Это приводит к необходимости применения различных алгоритмов анализа пакетов в системах обнаружения вторжений [1].

© Мистрюков В.С., Коваль А.С., 2025

Обобщённая структурная схема подобной системы обнаружения вторжений представлена на рис. 1.



А - источник трафика, Б - классификаторы трафика, В - хранилище, Г - консоль администрирования

Рис. 1. Обобщённая структурная схема систем обнаружения вторжений

Проектируемая система классификации трафика разрабатывается на основе аналогичной структуры. В качестве классификаторов трафика выбраны алгоритмы на основе эмпирического и сигнатурного анализа.

2. Алгоритм эмпирического анализ трафика

Эмпирический анализ трафика основывается на исследовании и обобщении характеристик сетевого трафика на основе наблюдений, экспериментальных данных и статистической обработки. В разрабатываемом решении алгоритм реализуется на основе вычисления векторов признаков, формирующихся на основе группировок пакетов по потокам и временным интервалам, и вычисляемых статистических данных для каждого очередного пакета, предложенных проектами N-BaIoT [2] и Kitsune [3].

На основе метки времени n-го пакета $ptime(n)$ интервалов между соседними пакетами в потоке $time_difference$ вычисляется коэффициент временного сглаживания $decay_factor$. Коэффициент временного сглаживания позволяет уменьшить влияние статистических данных, собранных по предыдущим пакетам на данные текущего пакета.

$$time_difference = ptime(n) - ptime(n-1), \quad (1)$$

$$decay_factor = 2^{(-ptime(1)*time_difference)}. \quad (2)$$

Вес n-го пакета $weight(n)$ отражает активность потока с учётом времени и определяется на основе количества пакетов в потоке.

$$weight(n) = 1 + weight(n-1)*decay_factor. \quad (3)$$

Средний размер пакета в потоке $mean(n)$ определяется на основе линейной суммы размеров пакетов $lsum(n)$ и веса потока $weight(n)$.

$$lsum(n) = size(n) + lsum(n-1) * decay_factor, \quad (4)$$

$$mean(n) = \frac{lsum(n)}{weight(n)}. \quad (5)$$

Дисперсия размера пакета в потоке $vnc(n)$ отражает разброс размеров пакетов в потоке и определяется на основе суммы квадратов размеров пакетов $ssum(n)$, веса пакета $weight(n)$ и среднего размера пакета в потоке $mean(n)$.

$$ssum(n) = size(n)^2 + ssum(n-1) * decay_factor, \quad (6)$$

$$vnc(n) = \frac{ssum(n)}{weight(n)} - mean(n)^2. \quad (7)$$

Стандартное отклонение размера пакета от среднего значения.

$$std(n) = \sqrt{vnc(n)}. \quad (8)$$

Ковариация размеров пакетов двух потоков $\mathrm{cov}(x, y, n)$ отражает совместное изменение размеров пакетов двух потоков x и y и определяется на основе суммы разностей размеров пакетов потоков с средними размерами пакетов соответствующих потоков и общего веса потоков.

$$sdiff(n) = size(n) - mean(n), \quad (9)$$

$$\mathrm{cov}(x, y, n) = \frac{\sum (sdiff_x(n) * sdiff_y(n))}{weight_{x,y}(n)}. \quad (10)$$

Коэффициент корреляции Пирсона $pcc(x, y, n)$ показывает силу линейной связи между потоками x и y

$$pcc(x, y, n) = \frac{\mathrm{cov}(x, y, n)}{std_x(n) * std_y(n)}. \quad (11)$$

Радиус в многомерном пространстве $radius(n)$ отображает обобщённую меру разброса потока и других потоков в пространстве дисперсий.

$$radius(n) = \sqrt{\sum_{i=1}^{N} vnc_i(n)^2}. \quad (12)$$

Магнитуда потока отражает меру "мощности" потока и других потоков в пространстве средних значений.

$$magnitude(n) = \sqrt{\sum_{i=1}^{N} mean_i(n)^2} \ . \qquad (13)$$

Результатом вычисления вектора признаков для очередного пакета является набор признаков размерностью 115 элементов.

Для большей гибкости и уменьшения временных и вычислительных затрат на обучение моделей используется ансамбль из ряда моделей, ориентированных на обнаружение собственных конкретных видов вредоносного трафика. Применяемый алгоритм предсказания класса пакета представлен в листинге 1.

Листинг.

Алгоритм предсказания класса пакета на основе ансамбля моделей

```
function predict(data) {
    const predictions = predictAllModels(data);
    let threshold = probabilityThreshold;
    if (predictions.some(([label, prob]) => label === "benign"
&& prob > 0.99)) {
        threshold = 0.99;
    }
    const attackPrediction = predictions
        .filter(([label, prob]) =>
attackClasses.includes(label) && prob > threshold)
        .sort((a, b) => b[1] - a[1])[0];
    const predicted = attackPrediction ?? predictions.sort((a,
b) => b[1] - a[1])[0] ?? null;
    return predicted;
}
```

На первом этапе работы алгоритма происходит получение результатов по предсказанию класса со всех загруженных моделей. Далее осуществляется поиск предсказания с наибольшей вероятностью, среди предсказаний с классом, входящим в перечень классов атаки и предсказанной вероятностью класса выше некоторого порогового значения. При нахождении подобного предсказания оно определяется как итоговое предсказание. В противном случае результатом работы алгоритма принимается предсказание модели с максимальной вероятностью.

3. Сигнатурный анализ трафика

Сигнатурный анализ трафика основан на сопоставлении частей сетевых пакетов с заранее сформированной базой сигнатур, однозначно идентифицирующих определённые типы трафика, протоколов или атак. В разрабатываемой системе классификации трафика сигнатурный

анализ реализуется на основе организации взаимодействия двух ключевых компонентов: Snort - системы обнаружения вторжений с сигнатурным анализом сетевых пакетов, и LLM на открытом движке Ollama - локально развёрнутой платформы для запуска больших языковых моделей, предназначенной для обработки текстовых данных и генерации аналитических выводов. Алгоритм функционирования модуля сигнатурного анализа представлен на рис. 1.

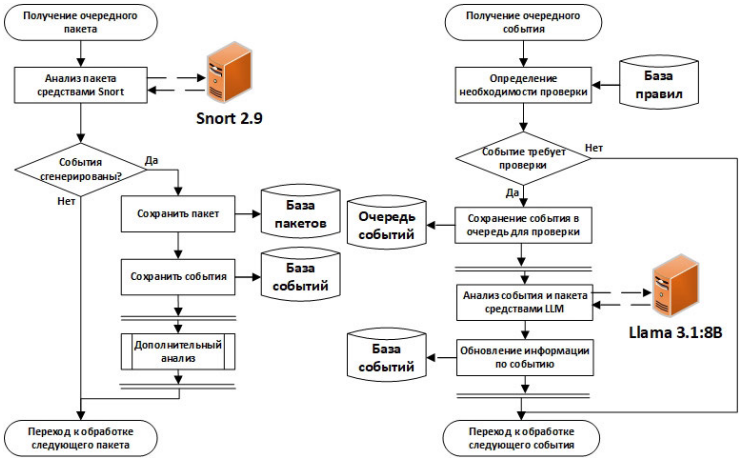


Рис. 2. Алгоритм функционирования модуля сигнатурного анализа: а – подпроцесс взаимодействия со Snort, б – подпроцесс дополнительного анализа событий и пакетов средствами LLM

Очередной пакет трафика зеркалируется на Snort, где выполняется его сигнатурный анализ, по результатам которого формируются события, по которым может быть принято решение о дополнительном анализе событий и содержимого пакета средствами LLM. В связи с высокой вычислительной и временной сложностью обработки каких-либо данных средствами LLM, необходимость подобного дополнительного анализа событий определяется оператором разрабатываемого комплекса, путём написания пользовательских правил, реализуемых в виде динамически подключаемых плагинов.

4. Тестирование разработанного программного решения

Предварительное тестирование разработанного модуля эмпирического анализа проводится с использованием общедоступных датасетов N-BaIoT Dataset [4] и Kitsune Network Attack Dataset [5]. Была

сформирована методика тестирования, по которой на основе предварительно размеченных данных из датасета N-BaIoT Dataset и вычисленных на основе исходных PCAP записей трафика из датасета Kitsune Network Attack Dataset формируются выборки по ряду атак, размеры которых представлены в таблице 1.

Таблица 1.

Данные для тестирования алгоритма эмпирического анализа

Атака	**Датасет**	**Учебная выборка**	**Тестовая выборка**
ACK flood	nbaiot	555 548	238 093
ARP MitM	kitsune	1 751 458	750 625
Active wiretap	kitsune	1 593 991	683 139
Combo	nbaiot	402 997	172 713
Fuzzing	kitsune	1 569 477	672 634
Junk spam	nbaiot	381 354	163 437
OS scan	kitsune	1 188 496	509 355
SSL renegotiation	kitsune	1 544 364	661 871
Vuln scan	nbaiot	560 514	240 221
SYN flood	nbaiot	44 559	249 185
SSDP flood	kitsune	2 853 313	1 222 849
TCP flood	nbaiot	427 369	183 159
UDP flood	nbaiot	823 994	353 141

Для каждого вида перечисленных атак обучается собственная модель. На вход каждой модели подаются векторы признаков, на выходе каждой модели выдаётся кортеж, характеризующий принадлежность пакета вредоносному или легитимному трафику, а также значения соответствующих вероятностей.

Для оценки эффективности применяются следующие метрики.

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn}, \tag{14}$$

$$precision = \frac{tp}{tp + fp}, \tag{15}$$

$$recall = \frac{tp}{tp + fn}, \tag{16}$$

$$f1_score = 2 * \frac{precision * recall}{precision + recall} accuracy = \frac{tp + tn}{tp + tn + fp + fn}. \tag{17}$$

Результаты тестирования алгоритма эмпирического анализа представлены в таблице 3.

Таблица 2

Результаты тестирования ансамбля моделей

Атака	**Accuracy**	**Precision**	**Recall**	**F1 score**
ACK flood	0.9985	0.9985	0.9985	0.9985
ARP MitM	0.7198	0.8152	0.7198	0.6867
Active wiretap	0.7506	0.8243	0.7506	0.7168
Combo	0.9979	0.9979	0.9979	0.9979
Fuzzing	0.8487	0.8725	0.8487	0.8064
Junk spam	0.9977	0.9977	0.9977	0.9977
OS scan	0.9965	0.9967	0.9965	0.9965
SSL renegotiation	0.9998	0.9998	0.9998	0.9998
Vuln scan	0.9974	0.9974	0.9974	0.9974
SYN flood	0.9985	0.9985	0.9985	0.9985
SSDP flood	0.9999	0.9999	0.9999	0.9999
TCP flood	0.9980	0.9981	0.9980	0.9981
UDP flood	0.9984	0.9984	0.9984	0.9984

Предварительное тестирование модуля сигнатурного анализа, включая автоматическую предварительную верификацию проводится на образцах трафика, содержащего как попытки атак, так и запросы, вызывающие ложноположительные срабатывания на существующих системах обнаружения вторжений. Пример распознавания вредоносного трафика приведён на рис. 1.

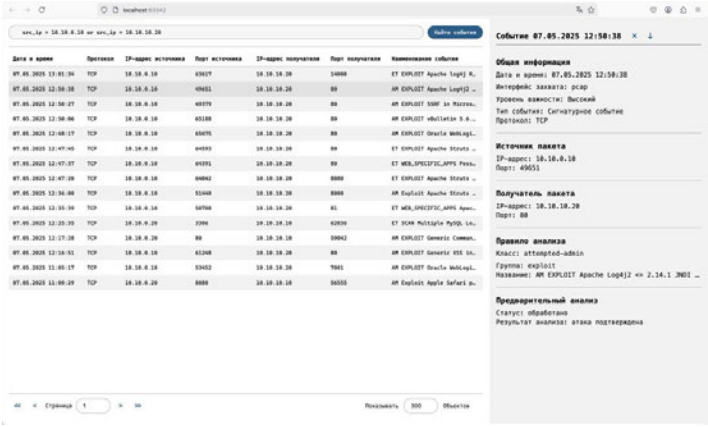


Рис. 3. Пример распознавания вредоносного трафика

Дальнейшее тестирование модуля сигнатурного анализа, включая определение эффективности верификации на больших выборках разнообразного трафика предполагается в рамках последующих работ.

Заключение

Данная статья посвящена разработке и реализации системы выявления потенциальных событий информационной безопасности на основе многоуровневой классификации сетевого трафика с применением глубокого анализа пакетов. Описаны алгоритмы многоуровневой классификации трафика и создание самой разрабатываемой системы. В итоге данной разработки получился рабочий программный продукт, реализованный в виде приложения. Использование данного решения позволяет выявлять в сетевом трафике потенциальные индикаторы атак при помощи различных методов, формировать события безопасности, а также проводить их предварительную автоматизированную проверку. Оператор данной системы имеет возможность просматривать информацию по выявленным событиям через веб-интерфейс. В дальнейшем предполагается использование наработок данного проекта для модернизации и улучшения существующих и применяющихся систем обнаружения вторжений.

Список литературы

1. Сетевой сенсор системы обнаружения атак программно-аппаратный комплекс ViPNet IDS NS 3. Руководство администратора [Электронный ресурс] – Режим доступа: https://files.infotecs.ru/_dl/sess/vipnet_ids_ns/docs/ViPNet%20IDS%20NS. %20Руководство администратора.pdf.7z
2. Yair Meidian, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Dominik Breitenbacher, Asaf Shabtai, Yuval Elovici. N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders [Электронный ресурс] – Режим доступа : https://arxiv.org/pdf/1805.03409
3. Yair Meidian, Tomer Doitshman, Yuval Elovici, Asaf Shabtai. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection [Электронный ресурс] – Режим доступа : https://arxiv.org/pdf/1802.09089
4. N-BaIoT Dataset to Detect IoT Botnet Attacks [Электронный ресурс] : база данных. – Режим доступа : https://www.kaggle.com/datasets/mkashifn/nbaiot-dataset
5. Kitsune Network Attack Dataset [Электронный ресурс] : база данных. – Режим доступа : https://www.kaggle.com/datasets/ymirsky/network-attack-dataset-kitsune

Разработка ядра для буферизованной обработки аудиофайлов

П. С. Мишин

Студент магистр

А. И. Чекмарёв

Старший преподаватель

Введение

В статье рассматривается проблема эффективного распознавания нот и аккордов в аудиопотоке, поступающем от пользователя мобильного приложения, предназначенного для обучения игре на гитаре. Предлагается гибридная архитектура клиент-серверного взаимодействия, при которой ресурсоёмкая обработка звука делегируется на удалённый сервер. Реализованное ядро на языке Go демонстрирует высокую производительность и низкую задержку при распознавании звуковых паттернов в различных сценариях использования. Представлены архитектурные решения, применённые алгоритмы и результаты экспериментального сравнения режимов обработки.

Современные мобильные приложения для обучения музыке предъявляют высокие требования к скорости и точности обработки аудиовхода. Особенно это актуально для приложений, обеспечивающих интерактивное обучение игре на гитаре, где требуется в реальном времени распознавать проигрываемые ноты и аккорды. Выполнение данной задачи непосредственно на пользовательском устройстве может приводить к перегреву, замедлению интерфейса и быстрой разрядке аккумулятора. В связи с этим целесообразно перераспределить вычислительную нагрузку на удалённый сервер, обладающий большей вычислительной мощностью и возможностями масштабирования.

1. Постановка задачи

Целью настоящей работы является проектирование и реализация вычислительного ядра, способного:

– распознавать ноты и аккорды в аудиопотоке с задержкой не более 0.1 секунды на 1 секунду аудио,

– обрабатывать потоки параллельно,

– масштабироваться горизонтально при росте нагрузки,

© Мишин П. С., Чекмарёв А. И., 2025

– эффективно использовать многопоточную обработку для минимизации времени анализа.

2. Архитектура решения

Разработанная архитектура основывается на концепции гибридного клиента: мобильное приложение осуществляет предварительную сегментацию и передачу аудиоданных на удалённый сервер, где выполняется основная вычислительная нагрузка (рис. 1).

Рис. 1. Верхнеуровневая схема работы

Ядро обработки реализовано как параллельный сервис, способный обслуживать множество запросов одновременно [1]. Это обеспечивается за счёт масштабируемой обработки в горутинах (goroutines), предоставляемых языком Go, а также буферизованного подхода к обработке звуковых файлов.

3. Алгоритмическое обеспечение

В основе обработки звукового сигнала лежит алгоритм Кули–Тьюки, реализующий быстрое преобразование Фурье (БПФ) [2]. Данный алгоритм рекурсивно разбивает входной массив на чётные и нечётные элементы и объединяет результаты с использованием комплексных экспонент, что обеспечивает асимптотическую сложность O(n log n). Это позволяет эффективно извлекать частотные компоненты из аудиопотока в режиме реального времени.

После определения доминирующих частот в сигнале осуществляется поиск ближайшей ноты. Поскольку частоты музыкальных нот упорядочены по возрастанию, для поиска применяется бинарный алгоритм со сложностью O(log(n)). Это обеспечивает быстрое соответствие частоты конкретной ноте из таблицы частот.

4. Быстрое преобразование Фурье (Кули-Тьюки)

Функция FFTCooleyTukey принимает на вход массив действительных чисел, представляющий аудиосигнал во временной области, и возвращает массив комплексных чисел, соответствующих амплитудам частотных компонент. Если размер входного массива равен единице, выполняется выход из рекурсии с возвратом комплексного числа с нулевой мнимой частью.

Затем массив рекурсивно разбивается на две части: элементы с чётными и нечётными индексами. Для каждой из этих частей вызывается преобразование Фурье отдельно. После этого, используя метод "бабочки", полученные спектры объединяются в результирующее частотное представление, принцип работы которого продемонстрирован на рис. 2.

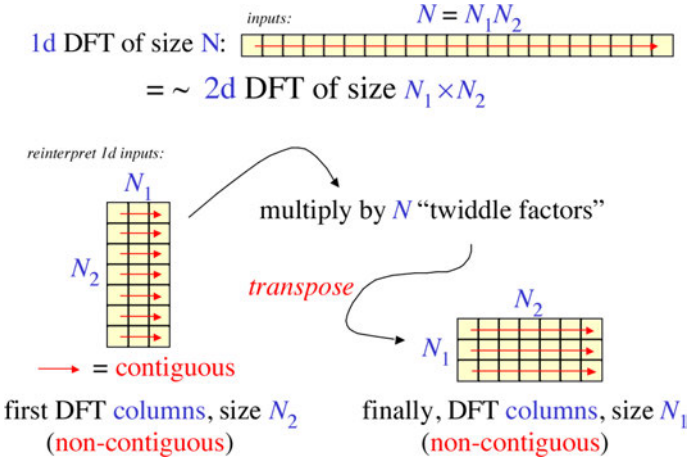


Рис. 2. Схема работы алгоритма Кули-Тьюки

При объединении используется вращающий множитель $e^{-2\pi ik/n}$, реализованный с помощью функции комплексной экспоненты. Такой подход позволяет существенно сократить вычислительную сложность с $O(n^2)$ при прямом преобразовании Фурье до O(n log n), что критично для обработки в реальном времени. Реализация вращающего множителя с функцией комплексной экспоненты представлен на листинге 1.

Листинг 1

Реализация алгоритма Кули-Тьюки на языке Go

```
func FFTCooleyTukey(input []float64) []complex128 {
    n := len(input)
    if n == 1 {
       return []complex128{complex(input[0], 0)}
    }

    even := make([]float64, 0, n/2)
    odd := make([]float64, 0, n/2)
    for i := 0; i < n; i += 2 {
       even = append(even, input[i])
       if i+1 < n {
          odd = append(odd, input[i+1])
       }
    }

    evenFFT := FFTCooleyTukey(even)
    oddFFT := FFTCooleyTukey(odd)

    combined := make([]complex128, n)
    half := n / 2
    for k := 0; k < half; k++ {
       theta := -2.0 * math.Pi * float64(k) / float64(n)
       w := cmplx.Exp(complex(0, theta))

       term := w * oddFFT[k]
       combined[k] = evenFFT[k] + term
       combined[k+half] = evenFFT[k] - term
    }

    return combined
}
```

5. Бинарный поиск частот

Для точного сопоставления обнаруженной частоты с нотой из эталонного набора используется алгоритм бинарного поиска, реализованный в функции GetClosestNote (Листинг 2). Входными параметрами функции являются частота freq, извлечённая из аудиосигнала, и упорядоченный по возрастанию список нот notes, каждая из которых содержит соответствующую частоту.

Поиск осуществляется итеративно с использованием границ low и high, между которыми двигается середина mid, вычисляемая как целочисленное деление пополам [3]. На каждой итерации сравнивается частота ноты в позиции mid с искомой. Если частота из списка меньше заданной, граница low сдвигается вверх; в противном случае — граница

high сдвигается вниз. Таким образом, обеспечивается логарифмическая сложность O(log n).

Листинг 2

Реализация бинарного поиска ноты, ближайшей к переданной частоте

```
func GetClosestNote(freq float64, notes []model.Note)
model.Note {
    defer util.MeasureExecutionTime("GetClosestNote", true)()
    n := len(notes)
    low, high := 0, n-1

    for low <= high {
       mid := (low + high) >> 1
       if notes[mid].Frequency < freq {
          low = mid + 1
       } else {
          high = mid - 1
       }
    }

    if low == 0 || low == n {
       return notes[low%n]
    }

    if math.Abs(notes[low].Frequency-freq) <
math.Abs(notes[low-1].Frequency-freq) {
       return notes[low]
    }
    return notes[low-1]
}
```

После завершения цикла производится проверка на выход за границы списка, и возвращается либо ближайший элемент, либо наиболее близкая по модулю разницы частота среди двух соседних элементов (на позициях low и low-1). Это позволяет с высокой точностью сопоставить вычисленную частоту с ближайшей стандартной нотой, что критично для корректной обратной связи в обучающем приложении.

6. Реализация и средства

Для реализации ядра выбран язык Go, обладающий следующими преимуществами:

- высокая производительность;
- встроенная поддержка конкурентности;
- простота развёртывания и масштабирования.

Обработка аудиофайлов реализована с использованием горутин и каналов, обеспечивающих независимую обработку каждого файла без

блокировок. Распараллеливание процессов существенно снижает общее время обработки [4].

7. Экспериментальные исследования

Для оценки производительности ядра были проведены три серии бенчмарков на оборудовании с 12 логическими потоками и 32 ГБ ОЗУ.

В таблице приведены сравнения результатов экспериментальных исследований производительности ядра при обработке аудиофайлов в разных сценариях.

Таблица

Сравнение экспериментальных исследований производительности

Тип теста	Общее время аудио, с	Режим	Время обработки, с	Отношение,с/с
Множество коротких файлов	402 (201x2)	Многопоточный	9.41	0.0234
		Однопоточный	28.53	0.0709
Один короткий файл	2	Многопоточный	0.14	0.0700
		Однопоточный	0.16	0.0800
Один длинный файл	170	Многопоточный	12.2	0.0718
		Однопоточный	13.08	0.0769

8. Перспективы развития

Возможные направления дальнейшей работы:

– применение детекторов аккордов на базе TensorFlow Lite [5],
– автоматическое определение темпа и ритма,
– интеграция с Kubernetes для масштабируемости,
– использование ядра для автоматического нотного сопровождения композиций.

Заключение

В ходе работы реализовано высокопроизводительное ядро для обработки аудиофайлов, позволяющее:

– достигать латентности менее 0.05 с на 1 с аудио,
– эффективно использовать параллельные вычисления,
– масштабироваться под нагрузки от массовых клиентов,
– интегрироваться в мобильные приложения в формате гибридного сервиса [6].

Список литературы

1. Блог компании BI.ZONE. Как на Go решить задачку по распознаванию аудио [Электронный ресурс] : Хабр. – Режим доступа: https://habr.com/ru/companies/bizone/articles/480092/

2. Hinnenkamp T. A Guide to Building a Realtime HTTP Audio Streaming Server in Go [Электронный ресурс] : Medium. – Режим доступа: https://medium.com/@icelain/a-guide-to-building-a-realtime-http-audio-streaming-server-in-go-24e78cf1aa2c

3. Lu Sh. Binary Search [Электронный ресурс] : Habr. – Режим доступа: https://habr.com/en/articles/563652/

4. Golub V. Пессимизм насчёт многопоточности [Электронный ресурс] : Хабр. – Режим доступа: https://habr.com/ru/articles/434698/

5. Singh A. Neural Networks for Real-Time Audio: Introduction [Электронный ресурс] : Medium. – Режим доступа: https://medium.com/nerd-for-tech/neural-networks-for-real-time-audio-introduction-ed5d575dc341

6. Блок компании Белый код. Обзор гибридной интеграционной платформы USEBUS [Электронный ресурс] : Хабр. – Режим доступа: https://habr.com/ru/companies/w_code/articles/905048/

Анализ лингвистического корпуса профилей студентов в социальных сетях

Э. В. Обухова

Студент магистр

А. В. Швырева

Старший преподаватель

Введение

В современном мире социальные сети стали неотъемлемой частью жизни, играя важную роль в формировании цифрового следа пользователей. Этот след представляет собой совокупность различной информации, как-либо характеризующей человека. На его основе можно создать «цифровой портрет личности». Одним из распространенных его применений являются разнообразные рекомендательные системы.

В данной работе в качестве цифрового следа рассматривался список групп пользователя в социальной сети ВКонтакте, так как он содержит данные о подписках на тематические сообщества, что позволяет выявить его интересы. Такой цифровой след может быть полезен для объективного анализа склонностей пользователя, что, в свою очередь, может помочь в выборе подходящего направления обучения в вузе.

Исследование [1] посвящены сбору открытых данных пользователей ВКонтакте, указавших в качестве учебного заведения Воронежский государственный университет. Эти данные используются и в этой работе.

В рамках данного исследования ставится цель провести анализ названий сообществ, в которых состоят студенты факультета компьютерных наук Воронежского государственного университета, чтобы выявить, достаточно ли разнесены усредненные кластеры факультетов при визуализации данных, чтобы их можно было разделить с помощью дальнейшего использования моделей машинного обучения.

1. Инструменты для анализа текстов

В качестве инструментов анализа текстов был выбран язык Python 3, и нижеперечисленные библиотеки:

© Обухова Э. В., Швырева А. В., 2025

1. **Pandas** – библиотека Python для обработки и анализа структурированных данных;

2. **Matplotlib** – библиотека для графической визуализации данных, построении зависимостей,

3. **ScikitLearn –** библиотека, содержащая в себе инструменты для математического анализа данных;

4. **Word2Vec** – библиотека для векторизации текстовых корпусов [2-3].

2. Описание анализируемого набора данных

В табл. 1 представлена структура набора данных, используемых для проведения лингвистического анализа названий групп студентов.

Набор состоит из 675465 уникальных записей, каждая из которых содержит уникальный идентификатор группы (ID), уникальный идентификатор факультета (FACULTY_ID), обозначение которых представлено в табл. 2, и наименование группы (GROUP_NAME). Полученный набор названий сообществ в сумме образует корпус из 2093975 слов.

Таблица 1

Структура набора данных для эксперимента со случайно выбранными примерами данных

id	faculty_id	group_name
224	2	Сам себе дизайнер
4	0	СНО истфака ВГУ
9	0	Поиск мемов по описанию
20	0	Книги - скачать
334	3	Донжон - магазин товаров Средневековья и фэнтези
341	3	Типичный программист
26651	7	Нетипичный Воронеж
26632	7	Наука и Факты
26690	7	Дизайн интерьера

Таблица 2

Таблица соответствия уникального идентификатора факультета названию факультета ВГУ

id	Название факультета	Код типа интеллекта
0	Исторический факультет	2
1	Юридический факультет	2
2	Факультет романо-германской филологии	3
3	Факультет прикладной математики, информатики и механики	0
4	Факультет философии и психологии	2
5	Экономический факультет	2
6	Математический факультет	0
7	Факультет компьютерных наук	0
8	Химический факультет	1
9	Физический факультет	0
10	Фармацевтический факультет	1
11	Филологический факультет	3
12	Факультет журналистики	3
13	Медико-биологический факультет	1
14	Факультет международных отношений	2
15	Факультет географии, геоэкологии и туризма	1
16	Геологический факультет	1

На рисунке 1 представлена столбчатая диаграмма, отражающая соответствие между идентификатором типа интеллекта и количеством студентов, обучающихся на соответствующим этому типу интеллекта факультете.

В соответствии с предложенным Говардом Гарднером разделением интеллекта человека на восемь типов [4], промаркируем повторно набор данных, распределив факультеты по типам интеллекта. В ВГУ их представлено всего четыре: 0 – математико-логический, 1 – естественно-натуралистический, 2 – социальный, 3 – лингвистический. Соответствие этой маркировки факультетам ВГУ представлено в табл. 2.

На рис. 2 показано статистическое соотношение направлений интеллекта в студентах ВГУ.

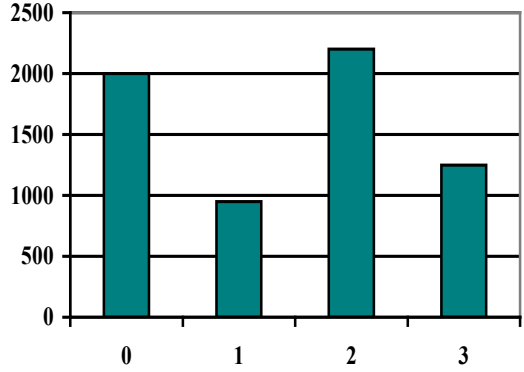


Рис. 1. Диаграмма зависимости количества студентов и идентификатора типа интеллекта

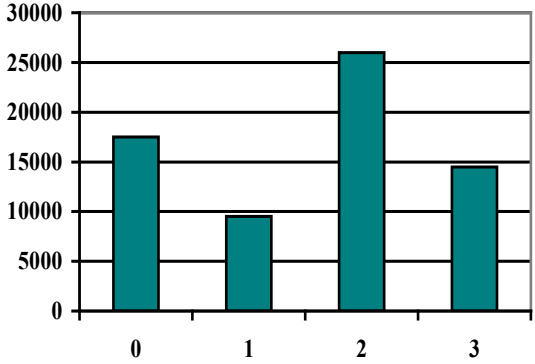


Рис. 2. Диаграмма зависимости количества групп и идентификатора направления интеллекта

3. Векторизация набора данных

Рассмотрим обработку корпуса слов. Целью обработки было составить для каждого студента мешок его слов, перевести их в численное представление и сократить в один 2-мерный вектор. Далее эти вектора будут использоваться для визуализации векторов факультетов.

Для составления вектора, который характеризует одного конкретного студента, требуется сконкатенировать все названия его групп в одно «предложение», а затем подать его в модель Word2Vec, ознакомленную со словарем из всех слов, встречающихся у студентов.

Важно отметить, что для визуализации требуется не больше 3-х компонентов у вектора, но для более наглядного представления будем использовать двухкомпонентную векторизацию.

Также была применена предварительная обработка корпуса текста, чтобы снизить шум в данных. Перечислим процедуры обработки в порядке применения к тексту:

1. Токенизация названий групп с помощью библиотеки nltk [5]
2. Фильтрация лексем от знаков препинания
3. Приведение слов к нижнему регистру символов
4. Лемматизация слов
5. Обработка стеммером

Код для выполнения этих процедур приведен в листинге 1.

Листинг 1

Предварительная обработка текстового корпуса

```
import nltk
import string
import ru_core_news_sm
from nltk import SnowballStemmer

nltk.download(["stopwords", "punkt", "wordnet", "omw-1.4"])
stopwords = nltk.corpus.stopwords.words("russian")
stopwords.extend(nltk.corpus.stopwords.words("english"))
stoppunkt = string.punctuation + "\'s...—«»\'\'``•"
nlp = ru_core_news_sm.load()
stemmer = SnowballStemmer(language='russian')

def clean_bag_of_words(unit):
    unit_words = []
    try:
        unit_words = nltk.word_tokenize(unit)
        unit_words = [w.lower() for w in unit_words if
w.lower() not in stoppunkt]
        unit_words = [w for w in unit_words if w.lower() not
in stopwords]
        unit_words = [[token.lemma_ for token in nlp(w)] for w
in unit_words]
        unit_words = [stemmer.stem(w[0]) for w in unit_words]
    except Exception:
        print("exception while cleaning bag of words")
    return unit_words
```

Далее полученные мешки слов для каждого студента векторизовались с помощью библиотеки Word2Vec. Листинг кода по обработке групп студента представлен ниже:

Листинг 2

Векторизация текстового корпуса

```
from gensim.models import Word2Vec

model = Word2Vec(sentences=word_bag_sum, size=2, window=1,
min_count=1, workers=4)

def vectorize():
    try:
        return model.wv[word]
    except Exception as e:
        return [0, 0]

def vectorize_words_bag(word_list):
    word_vectors = []
    try:
        word_vectors = [vectorize(word) for word in word_list
if word in model.wv]
    except Exception:
        print("exception while vectorizing bag of words")
    return word_vectors

def sum_word_vectors(word_vectors):
    if len(word_vectors) == 0: return 0
    return sum(word_vectors) / len(word_vectors)

def process_word_bag(unit):
    unit = clean_bag_of_words(unit)
    unit = vectorize_words_bag(unit)
    unit = sum_word_vectors(unit)
    return unit
```

Результат векторизации можно наблюдать на рис. 3.

5 rows | 5 rows × 4 columns

	person_id	faculty	group_name	student_vector
0	42105	1509	HistoryPorn Издательство "Лаборатория зн…	[0.04389196, -0.018676696]
1	99460	139366	Город Грехов | Старый Оскол Танцевальная…	[-0.065395944, -0.08073136]
2	152154	5787	Жажда путешествий ← Туры из Воронежа и Р…	[-0.11935708, 0.06915554]
3	169396	8559	абстрактные мемы для элиты всех сортов |…	[-0.0049495543, 0.011410482]
4	191064	37802	"Men's Classical" Идеальный ремонт Лучше…	[0.037879407, -0.0016916564]

Рис. 3. Несколько студентов с векторизованным набором групп

Далее вектора студентов были разбиты на наборы с одинаковым номером факультета внутри набора, а затем вектора были усреднены. Таким образом, были получены координаты факультетов на двумерной поверхности. Таблица значений, полученных для факультетов, приведена ниже.

Таблица 3

Таблица векторов для текстовых корпусов факультетов

Факультет	Компонент X	Компонент Y
Исторический факультет	0.003517	0.003418
Юридический факультет	0.005305	0.000736
Факультет романо-германской филологии	0.000702	0.006436
Факультет прикладной математики, информатики и механики	-0.000504	0.000926
Факультет философии и психологии	0.005209	0.003890
Экономический факультет	0.003381	0.003235
Математический факультет	0.002301	0.004125
Факультет компьютерных наук	0.001643	-0.003605
Химический факультет	0.005291	0.009750
Физический факультет	0.001018	0.000746
Фармацевтический факультет	-0.000745	-0.001669
Филологический факультет	-0.000005	0.006179
Факультет журналистики	0.006477	0.003712
Медико-биологический факультет	0.009283	0.007956
Факультет международных отношений	0.003633	0.007281
Факультет географии, геоэкологии и туризма	0.007629	0.003212
Геологический факультет	0.005178	0.005861

Визуализация закодированных в двумерные вектора факультетов продемонстрирована на рис. 4.

Данные визуализации подтверждают, что различение факультетов и направлений интеллекта студентов по их подпискам возможно, т.к. кластеры факультетов и направлений векторов хорошо лоцируются при визуальном анализе. Двумерная векторизация позволила наглядно показать различия между факультетами. Для оценки студентов возможно использовать эти вектора факультетов как эталонные образцы, вычисляя косинусное расстояние.

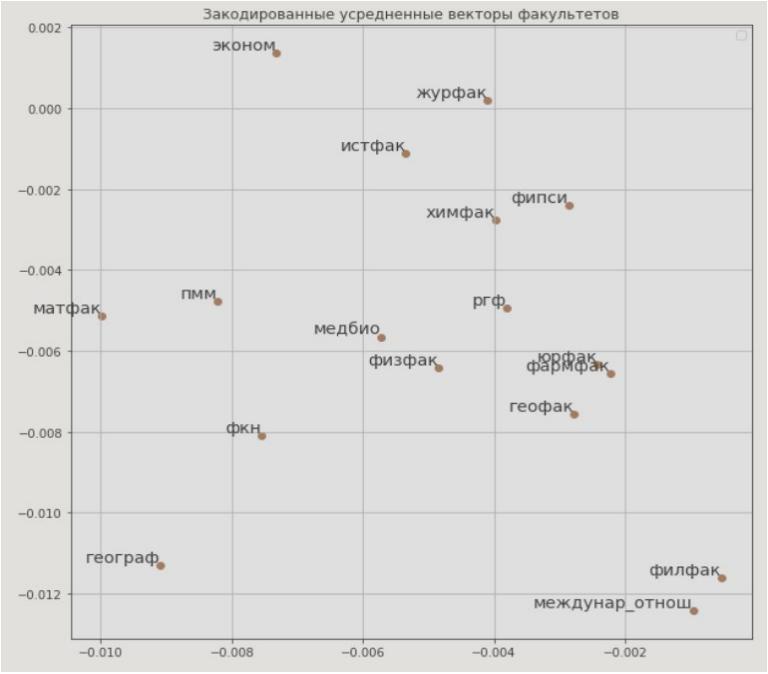


Рис. 4. Визуализация векторов факультетов

Те же операции были проделаны для направлений интеллекта. Их визуализация представлена на рисунке 5:

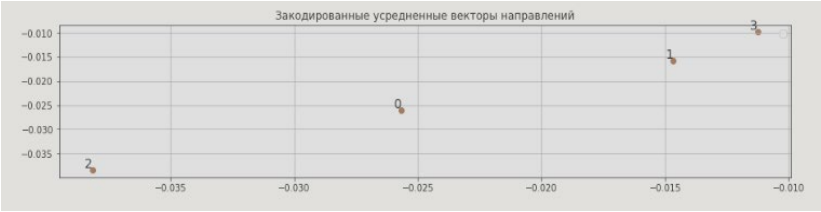


Рис. 5. Визуализация векторов направлений интеллекта

4. Тренировка модели на эмбеддингах групп студентов

Используя данную информацию, посмотрим, как можно применить эмбеддинги мешков слов для студентов для классификации. Для классификации используем 25-компонентную векторизацию и модель DNN:

Листинг 3

```
import tensorflow as tf
from tensorflow.keras import layers, models, regularizers

dnn = models.Sequential([
    layers.BatchNormalization(input_shape=(25,)),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.4),
    layers.Dense(128, activation='relu'),
    layers.Dense(128, activation='relu',
             kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(64, activation='relu'),
    layers.Dense(64, activation='relu',
             kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(32, activation='relu'),
    layers.Dense(4, activation="softmax")
])
```

Так же поставим раннюю остановку для модели, когда кривая обучения больше не улучшается. Считаем, что, если 15 поколений нет улучшения, то модель следует остановить:

Листинг 4

```
def train_model(train_x, train_y, val_x, val_y, model_name,
                    model, plot_history):
    if hasattr(model, 'compile') and callable(model.compile):
        model.compile(optimizer='adam',
             loss='sparse_categorical_crossentropy',
             metrics=['accuracy'],
             early_stop = EarlyStopping(
                         monitor='val_loss',
                         patience=15,
                         restore_best_weights=True
                     ))
        history = model.fit(x=train_x, y=train_y,
             validation_data=(val_x, val_y), epochs=1000,
             batch_size=32, verbose=1)
    else:
        model.fit(train_x, train_y)
    if plot_history and history is not None:
        plot_loss(history, model_name)
        plot_accuracy(history, model_name)
    return model
```

Для тестирования модели будем использовать пропорции тренировочных-валидационных-тестовых данных как 0.8-0.1-0.1. Графики кривых потерь и точности представлены на рисунке 6. По нему видно, что модель обучалась 1000 поколений (максимальное поставленное число). Это говорит о том, что есть потенциал для дальнейшего обучения.

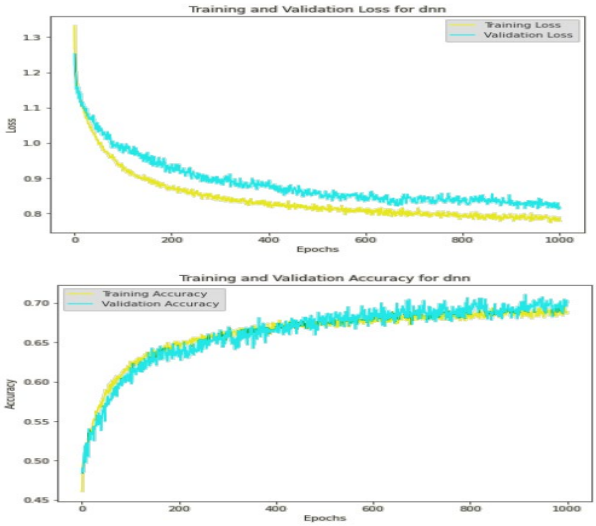


Рис. 6. График обучения модели

Точность модели достигла примерно 70% для четырех классов, в которые были объединены по смыслу студенты разных факультетов:

- математико-логический тип интеллекта,
- вербально-лингвистический тип интеллекта,
- социальный тип интеллекта,
- натуралистический тип интеллекта. [4]

Т.к. модель не остановилась сама при обучении, можно предположить, что при использовании более сложной архитектуры возможно увеличение точности классификации за меньшее число поколений.

Заключение

В результате проведенного исследования был получен набор векторов для студентов, факультетов и направлений интеллекта, представленных в ВГУ. Визуализирование данных векторов на плоскости позволило убедиться в гипотезе, что векторизованные усредненные лексемы в достаточной мере разделяют факультеты и направления интеллекта, чтобы произвести классификацию студентов таким способом. В дальнейшем будет проводиться работа по уточнению предсказания тестовой модели путем использования для эмбеддингов моделей-трансформеров.

Список литературы

1. Обухова, Э. В. Лингвистический анализ названий групп студентов ВГУ с целью выявления различий в текстовых корпусах между разными факультетами / Э. В. Обухова, А. В. Швырева // Труды молодых ученых факультета компьютерных наук ВГУ, сборник статей; Воронеж, Воронежский государственный университет, 15-17 февраля 2024 г. – Воронеж, ВГУ, 2024. – С. 467-478.

2. NLP – Преобразование текста: Word2Vec [Электронный ресурс] : Блог компании OTUS. Режим доступа: https://habr.com/ru/companies/otus/articles/574624/

3. Лобачевский С. Word2Vec: Подробное руководство с примерами кода [Электронный ресурс] : Блог nerdit.ru. – Режим доступа: https://nerdit.ru/word2vec/

4. Что такое теория множественного интеллекта и как она может помочь в обучении. [Электронный ресурс] : Блог компании Фоксфорд, 24.08.2023. – Режим доступа: https://media.foxford.ru/articles/teoriya-mnozhestvennogo-intellekta-govarda-gardnera

5. Анализ текстовых данных с помощью NLTK и Python [Электронный ресурс] : Блог компании OTUS. – Режим доступа: https://habr.com/ru/companies/otus/articles/774498/

Повышение точности разностных схем для бигармонического уравнения методом экстраполяции Ричардсона

Я. Э. Павлов

Студент магистр

М. И. Попов

Доцент

Введение

Краевые задачи для бигармонического уравнения, за редким исключением, не имеют точных аналитических решений, поэтому основным инструментом их интегрирования являются численные методы. Чтобы получить аппроксимацию искомого решения высокой точности, приходится использовать сетки очень большой размерности. В неявных конечно-разностных схемах увеличение размерности сеток приводит к необходимости решать системы, содержащие десятки, а то и сотни тысяч алгебраических уравнений. Несмотря на постоянный рост производительности вычислительных устройств, данная задача остается по-прежнему сложной. Решение таких систем алгебраических уравнений, даже в случае линейных систем, занимает очень много времени. В связи с этим возникает необходимость разработки методов, повышающих скорость расчета. В данной работе для повышения точности на небольших сетках используется метод экстраполяции Ричардсона. В качестве примера применения такого подхода рассматривается задача об изгибе квадратной пластины, защемленной по контуру.

1. Постановка задачи

Математическая модель прогиба $w(x,y)$ представляет собой неоднородное бигармоническое уравнение с граничными условиями первого и второго рода в квадратной области $\Omega = [0;1]\times[0;1]$:

$$\frac{\partial^4 w}{\partial x^4} + 2\frac{\partial^4 w}{\partial x^2 \partial y^2} + \frac{\partial^4 w}{\partial y^4} = \frac{q(x,y)}{D}. \quad (1)$$

$$w(0,y) = w(1,y) = w(x,0) = w(x,1) = 0, \quad (2)$$

© Павлов Я. Э., Попов М. И., 2025

$$\frac{\partial w(0,y)}{\partial x}=\frac{\partial w(1,y)}{\partial x}=\frac{\partial w(x,0)}{\partial y}=\frac{\partial w(x,1)}{\partial y}=0 \qquad (3)$$

Здесь $w(x,y)$ – прогиб пластины в точке (x,y), $q(x,y)$ – поперечная нагрузка, D – цилиндрическая жёсткость.

2. Построение конечно-разностной схемы

Заменим область решения системы Ω равномерной сеткой с шагами $\Delta x=\Delta y=h$, $\Omega_h=\left\{\left(x_i,y_j\right)=\left(ih,jh\right)\right\}$, $i,j=0,..,n$. Разностные аппроксимации частных производных $\partial^4 w/\partial x^4$, $\partial^4 w/\partial y^4$ и $\partial^4 w/\partial x^2\partial y^2$ 2-го порядка точности имеют вид [1]:

$$\left.\frac{\partial^4 w}{\partial x^4}\right|_{i,j}=\frac{w_{i+2,j}-w_{i+1,j}+6w_{i,j}-4w_{i-1,j}+w_{i-2,j}}{h^4}+O\left(h^2\right) \qquad (4)$$

$$\left.\frac{\partial^4 w}{\partial y^4}\right|_{i,j}=\frac{w_{i,j+2}-w_{i,j+1}+6w_{i,j}-4w_{i,j-1}+w_{i,j-2}}{h^4}+O\left(h^2\right) \qquad (5)$$

$$\left.\frac{\partial^4 w}{\partial x^2\partial y^2}\right|_{i,j}=\frac{1}{h^4}\left(w_{i+1,j+1}-2w_{i+1,j}+w_{i+1,j-1}-2w_{i,j+1}+4w_{i,j}-2w_{i,j-1}+\right.$$
$$\left.+w_{i-1,j+1}-2w_{i-1,j}+w_{i-1,j-1}\right)+O\left(h^2\right) \qquad (6)$$

Подставим в уравнение (1) аппроксимации производных и отбросим слагаемые более высокого порядка малости, получим разностное уравнение:

$$\frac{w_{i+2,j}-4w_{i+1,j}+6w_{i,j}-4w_{i-1,j}+w_{i-2,j}}{h^4}+\frac{2}{h^4}\left(w_{i+1,j+1}-2w_{i+1,j}+\right.$$
$$\left.+w_{i+1,j-1}-2w_{i,j+1}+4w_{i,j}-2w_{i,j-1}+w_{i-1,j+1}-2w_{i-1,j}+w_{i-1,j-1}\right)+ \qquad (7)$$
$$+\frac{w_{i,j+2}-4w_{i,j+1}+6w_{i,j}-4w_{i,j-1}+w_{i,j-2}}{h^4}=f_{i,j}$$

Граничные условия (2), накладываемые на искомую функцию, приобретут вид:

$$w_{0,j}=w_{n,j}=w_{i,0}=w_{i,n}=0 \qquad (8)$$

Аналог граничных условий, накладываемых на частные производные первого порядка, находим из соотношения для конечно-разностного аналога первой производной 2 порядка точности:

$$\frac{-3w_{0,j}+4w_{1,j}-w_{2,j}}{2h}=\frac{3w_{n,j}-4w_{n-1,j}+w_{n-2,j}}{2h}=$$
$$=\frac{-3w_{i,0}+4w_{i,1}-w_{i,2}}{2h}=\frac{3w_{i,n}-4w_{i,n-1}+w_{i,n-2}}{2h}=0. \quad (9)$$

Таким образом, непрерывная краевая задача (1) – (3) заменена конечно-разностной схемой (7) – (9). Полученная схема имеет неявный характер. На рис. 1 изображена расчётная схема.

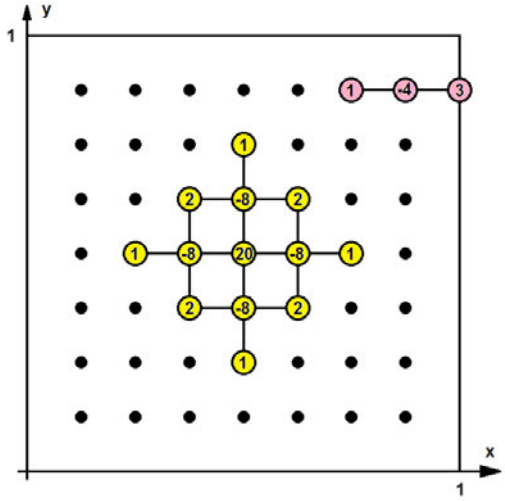


Рис. 1. Расчётная схема.

3. Анализ решения

Чтобы проанализировать погрешность расчётов, подберём точное решение задачи $w(x,y)=\frac{1}{\pi^4}\sin^2(\pi x)\sin^2(\pi y)$. Нетрудно увидеть, что условия 1 – 3 для тестовой функции выполняются.

В табл. 1 приведены расчёты относительных погрешностей сеточных функций.

По данным табл. 1, погрешность прямо пропорциональна квадрату шага, что говорит о корреляции теоретических оценок и реальных расчётов. Относительная погрешность возрастает по направлению от центра к границе области. Время расчёта растет быстрее экспоненты.

Таблица 1

Относительная погрешность сеточных функций

Шаг сетки	Относительная погрешность в центре области, %	Относительная погрешность в углах области, %	Время расчёта, с
1/10	2,94	22,2	7
1/20	1,2	7,08	14
1/30	0,6	3,45	54
1/40	0,355	2,03	191
1/50	0,235	1,33	502
1/60	0,166	0,935	1108
1/70	0,124	0,693	2121
1/80	0,096	0,533	4533
1/90	0,076	0,422	15749
1/100	0,0624	0,342	33087

4. Экстраполяция Ричардсона

В области Ω построим последовательность сеток $\Omega_h^k = \{(x_i, y_j) = \left(i\frac{h}{k}, j\frac{h}{k} \right)\}, \quad i = 0,..,nk,\ j = 0,..,nk,\ k = 1,..,p$. Каждая сетка последовательности содержит узлы первой сетки. Для каждой сетки последовательности вычислим сеточную функцию w_h^k. Экстраполированное решение представим в виде линейной комбинации сеточных функций w_h^k в общих узлах $W_h^p = \sum_{k=1}^{p} a_k^p w_h^k$. Весовые коэффициенты a_k^p находятся из системы уравнений, полученной разложением по чётным степеням h_k^2 [2]:

$$\sum_{k=1}^{p} a_k^p = 1, \quad \sum_{k=1}^{p} a_k^p h_k^{2j} = 0, \quad j = 1,..., p-1. \tag{10}$$

Параметр h_k будем измельчать по формуле $h_k = h/k$, тогда система (16) приводится к виду

$$\sum_{k=1}^{p} a_k^p = 1, \quad \sum_{k=1}^{p} \frac{a_k^p}{k^{2j}} = 0, \quad j = 1,..., p-1. \tag{11}$$

Решая данную систему, получим формулу для весовых коэффициентов

$$a_k^p = 2\frac{(-1)^{p-k}k^{2p}}{(p+k)!(p-k)!}. \qquad (12)$$

В табл. 2 приведены весовые коэффициенты, вычисленные по этой формуле для различных значений p [3].

Таблица 2

Весовые коэффициенты a_k^p

p \ k	1	2	3	4	5	6
2	$-\frac{1}{3}$	$\frac{4}{3}$				
3	$\frac{1}{24}$	$-\frac{16}{15}$	$\frac{81}{40}$			
4	$-\frac{1}{360}$	$\frac{16}{45}$	$-\frac{729}{280}$	$\frac{1024}{315}$		
5	$\frac{1}{8640}$	$-\frac{64}{945}$	$\frac{6561}{4480}$	$-\frac{16384}{2835}$	$\frac{390625}{72576}$	
6	$-\frac{1}{302400}$	$\frac{8}{945}$	$-\frac{2187}{4480}$	$\frac{65536}{14175}$	$-\frac{9765625}{798336}$	$\frac{17496}{1925}$

По табл. 2 видно, что чем из большего числа сеточных функций составляется улучшенное решение, тем весомее вклад последней функции w_h^p и незначительнее – первой w_h^1. Экстраполированное решение мы получаем только в узлах исходной сетки Ω_h, поэтому она должна содержать как можно больше узлов. С другой стороны, увеличение количества узлов исходной сетки приводит к значительному (в p^2 раз) увеличению количества узлов конечной сетки. Поэтому данный метод применим в тех случаях, когда нужно узнать очень точное значение в определённых точках.

В табл. 3 приведены расчёты относительной погрешности для экстраполяций Ричардсона. Для удобства сравнения результатов шаг последней сетки выберем одинаковым, равным 1/60.

Таблица 3

Относительная погрешность экстраполированных сеточных функций

Количество сеточных функций	Шаг первой сетки	Относительная погрешность в центре области, %	Время расчёта, с
2	1/30	0,0218	1162
3	1/20	0,00821	1313
5	1/12	0,00216	1806
6	1/10	0,00184	1876

Анализ табл. 3 показывает высокую эффективность метода. Комбинация всего лишь 2 сеточных функций уменьшает погрешность в 7 раз, увеличивая время расчёта на 5%. Комбинация 6 функций – в 90 раз, при этом время расчёта увеличивается менее чем в 2 раза. Однако использование в экстраполяции большего числа функций не приведет к увеличению точности. Таким образом, существует некоторое оптимальное число, обеспечивающее максимальную точность.

Заключение

С помощью метода конечных разностей построена неявная конечно-разностная схема численного решения краевой задачи для бигармонического уравнения. Получено решение тестового примера в виде сеточных функций для различных значений шага сетки. Проанализирована погрешность, полученная в ходе вычислительных экспериментов. Методом экстраполяции Ричардсона построены аппроксимации, точность которых на 2 порядка превосходит точность входящих в них сеточных функций.

Список литературы

1. Попов, М. И. Интегрирование бигармонического уравнения по неявной схеме / М. И. Попов // Вестник Воронежского государственного университета инженерных технологий. – 2018. – Т. 80. – № 2(76). – С. 114–118.

2. Марчук, Г. И. Повышение точности решений разностных схем. / Г. И. Марчук, В. В. Шайдуров. – М. : Главная редакция физико-математической литературы издательства «Наука», 1979. – 320 с.

3. Численные методы решения задач об изгибе квадратной пластины / М. Ю. Смирнов [и др.] // Вестник Поволжского государственного технологического университета. Серия: Материалы. Конструкции. Технологии. – 2021. – № 3. – С. 111–124.

Классификация неисправностей пассивных радиочастотных устройств по S-параметрам при помощи нейронных сетей

С. В. Перова

Студент магистр

А. В. Акимов

Старший преподаватель

А. В. Гречишкин

Доцент

Введение

Одной из наиболее трудоемких стадий в процессе производства изделий радиоэлектронной промышленности является проверка и отладка (регулировка) после сборки. Это связано в первую очередь с тем, что такие изделия являются высокотехнологичными и подразумевают проверку целого комплекса разнородных параметров. Помимо этого, в результате ошибок, обычно обусловленных человеческим фактором, в ходе сборки возникают неисправности: непропаи, короткие замыкания, нарушение расположения компонентов изделия или несоответствие номиналов и т.д. Также процесс проверки и отладки высокотехнологичных изделий требует высококвалифицированного специалиста-отладчика, который должен визуально осматривать, проверять компоненты в отдельности, заменять их и перепроверять работу устройства. Подготовка такого специалиста требует много времени.

Ценную информацию о причинах неисправностей можно получить, сосредоточившись на конкретных параметрах рассеяния (S-параметрах) [1] моделей с внесенными искажениями, которые будут размечены в зависимости от типа неисправности [2]. S-параметры описывают электрические свойства цепей при воздействии различных входных сигналов. Форма кривых S-параметров может указывать на специфические особенности сборки электронных устройств. Эффективная диагностика неисправностей пассивных радиочастотных устройств позволит сократить издержки производства и способствует

© Перова С. В., Акимов А. В., Гречишкин А. В., 2025

повышению производительности и поддержанию надежности устройств.

В данной работе предлагается методика диагностики неисправностей с анализом первопричин с использованием S-параметров, обеспечивающая раннее обнаружение и высокую точность диагностики. Будет рассмотрен вариант архитектуры нейронной сети для классификации неисправностей по сгенерированным S-параметрам модели 2-портового низкочастотного фильтра Кауэра-Чебышева 5-го порядка (рис. 1), далее испытуемое устройство (ИУ). Нумерация компонентов цепи на рис. 1 сквозная (номер расположен после имени компонента).

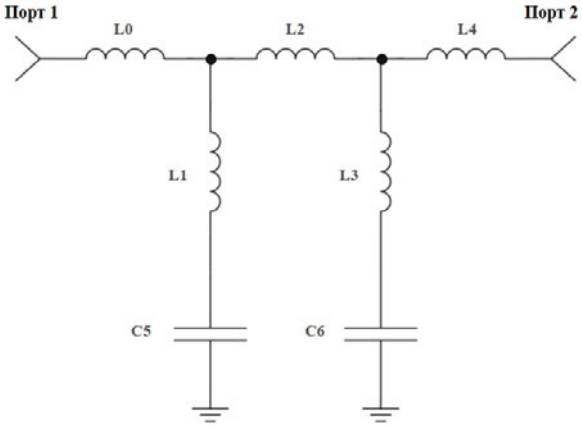


Рис. 1. Электрическая схема моделируемого 2-портового низкочастотного фильтра Кауэра-Чебышева 5-го порядка

1. Моделирование S-параметров

S-параметры описывают реакцию цепи N портов (в данном случае 2-х) на различные сигналы. Для измерения S-параметров тестируемых устройств, работающих в высокочастотном диапазоне, используется векторный анализатор цепей (ВАЦ) как необходимое в процессе проверки и отладки средство измерения. Для порта i: a_i и b_i — комплексные амплитуды падающей и отраженной волны, которые зависят от напряжения V_i, тока I_i и нормированы через характеристический импеданс Z_0 (50 Ом):

$$a_i = \frac{V_i + Z_0 I_i}{2\sqrt{Z_0}}, \quad b_i = \frac{V_i - Z_0 I_i}{2\sqrt{Z_0}} \tag{1}$$

Коэффициенты отражения S_{11}, S_{22} и коэффициенты передачи S_{12}, S_{21} для 2-портовой цепи рассчитываются следующим образом:

$$S_{11} = \left.\frac{b_1}{a_1}\right|_{a_2=0}, \quad S_{12} = \left.\frac{b_1}{a_2}\right|_{a_1=0}, \quad S_{21} = \left.\frac{b_2}{a_1}\right|_{a_2=0}, \quad S_{22} = \left.\frac{b_2}{a_2}\right|_{a_1=0} \tag{2}$$

Значения S-параметров являются комплексными числами, поэтому размерность входных данных нейронной сети можно записать как `(n_points, 8)`, где каждой из 501 `n_points` частотной точки из диапазона от 0 до 140 МГц будет соответствовать массив `[S11_re, S12_re, S21_re, S22_re, S11_im, S12_im, S21_im, S22_im]`. В случае симметричных фильтров, которые спроектированы таким образом, что их геометрия и электрические параметры одинаковы относительно их входа и выхода, S_{11} и S_{22} равны. Параметры S_{21} и S_{12} у исправного устройства будут также совпадать по величине и фазе, что выполняется для пассивных и линейных устройств [1]. Появление различий между параметрами, которые должны совпадать, свидетельствует о наличии некоторого дефекта.

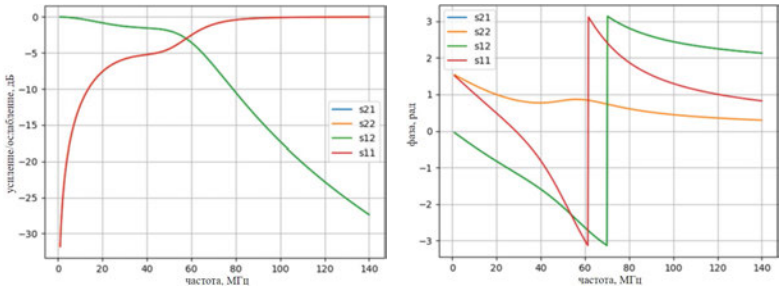
Для моделирования ИУ и соответствующих его возможным состояниям S-параметров на основе его схемы (рис. 1) и номиналов компонентов написана программа на Python с использованием scikit-rf [3], пакета для симуляции различных RF-компонентов. В программе вычисление S-параметров ИУ осуществляется на основе S-параметров всех входящих в него компонентов. То есть, после задания номиналов и указания при необходимости неисправности формируются все используемые компоненты. В результате их соединения по заданной схеме получена цепь с S-параметрами моделируемого ИУ.

2. Генерация неисправностей и обучающей выборки

В качестве дефектных состояний выбранного количества компонентов, а именно 7, в составе моделируемого устройства рассматриваются отсутствие контакта в одной из точек подключения компонентов ИУ и короткое замыкание в одной из точек подключения компонентов ИУ. В продолжение исследований предполагается расширение рассматриваемых неисправностей классом неисправностей с неправильным номиналом каждого из компонентов ИУ в отдельности, а также классами, представленными сочетаниями нескольких неисправностей. На рис. 2 представлен пример отсутствия контакта в

одной точке, при моделировании схемы фильтра компонент L3 заменен цепью с параметрами:

$$S_{11}=S_{22}=1, \quad S_{12}=S_{21}=0 \qquad 1$$

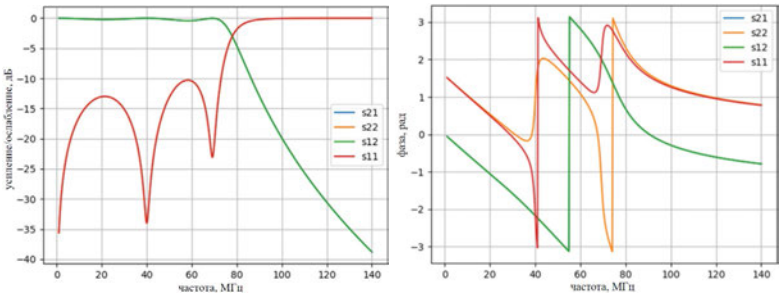


а б

Рис. 2. Графики S-параметров ИУ при отсутствии контакта в компоненте L3: а – амплитуда в дБ, б – фаза в радианах

Класс неисправности с коротким замыканием (рис. 3) можно смоделировать, добавив вместо некоторого компонента цепь с параметрами:

$$S_{11}=S_{22}=0, \quad S_{12}=S_{21}=1 \qquad (3)$$

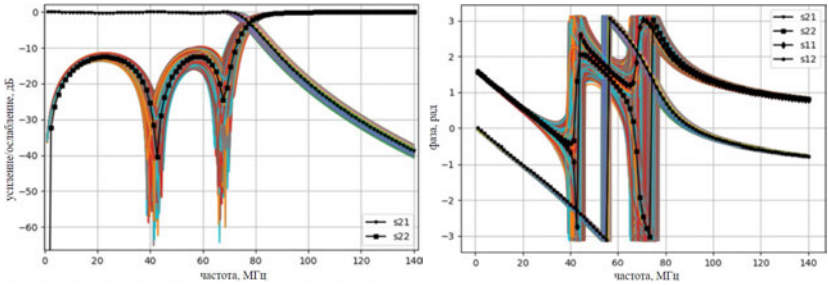


а б

Рис. 3. Графики S-параметров ИУ при коротком замыкании в L3: а – амплитуда в дБ, б – фаза в радианах

Экземпляры класса исправных устройств рассчитываются на основе допустимого разброса номиналов в ±5% для всех компонентов смоделированного устройства.

Проводилось сто симуляций (рис. 4), в которых из равномерно распределенных разбросов номиналов каждого компонента случайно выбирались значения. На их основе рассчитывалась модель фильтра и амплитуды и фазы ее S-параметров. На легенде отдельно обозначены линии S-параметров для точных значений номиналов компонентов.



а б

Рис. 4. Моделирование исправного состояния ИУ, 100 симуляций: а – амплитуда в дБ, б – фаза в радианах

В случае с наличием неисправности в некотором компоненте вместо его S-параметров в схему подставляются S-параметры соответствующей неисправности. Остальным компонентам также присваивается значение из допустимого диапазона. Таким образом, рассматривается 15 классов, из которых 14 соответствуют двум типам неисправностей, проявляющимся в каждом из 7 компонентов по отдельности. Они условно обозначаются в данной работе через номер компонента от 0 до 6, как на схеме с рис. 1, и состояние (open, short), например `01_short`. Исправное состояние имеет метку `class_correct`. Каждый класс содержит по 100 экземпляров.

3. Архитектура нейронной сети и ее оценка

Перед обучением сгенерированные экземпляры были предобработаны. При генерации некоторых неисправностей значения S-параметров были близки к нулю. При их переводе в дБ функцией десятичного логарифма библиотеки NumPy возвращалась минус бесконечность. Она заменяется фиксированным значением -70, выбранным эмпирическим путем и находящимся около нижней границы амплитуд генерируемых S-параметров. После данные стандартизировались, то есть для каждого признака вычиталось среднее значение и проводилось масштабирование на стандартное отклонение, что позволило привести их к единому масштабу и улучшить сходимость

модели при обучении. Также было проведено разделение на обучающую, валидационную и тестовую выборки.

Для создания модели нейронной сети применяется `tensorflow.keras` [4] – надстройка над фреймворком TensorFlow, предоставляющая высокоуровневый API для создания и обучения нейронных сетей на языке Python. В предлагаемой архитектуре (рис. 5) вначале выделяются локальные признаки по частотам. Для этого применяются два блока свертки с одномерными сверточными слоями `Conv1D` [5] (функция активации `ReLU`) и слоями нормализации `BatchNormalization`. Слои свертки используют различное количество фильтров. Этот параметр преобразует размерность значений признаков `features` внутри нейронной сети. В качестве слоя дискретизации используется `AveragePooling`, с которым модель демонстрирует наилучшую точность. Его параметр, отвечающий за размер окна усреднения, также изменяет размерность `n_points`.

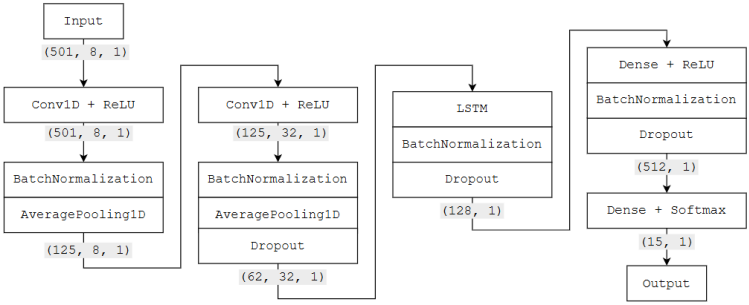
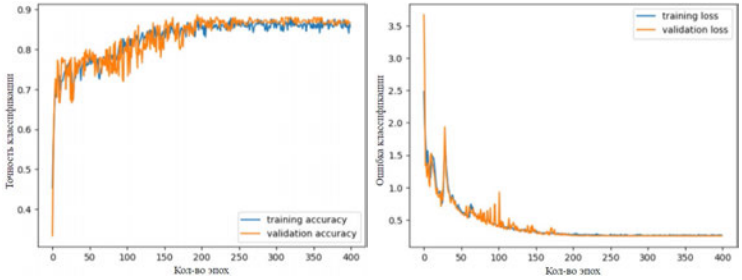


Рис. 5. Предлагаемая архитектура с использованием слоев из `tensorflow.keras.layers`, обозначение размерности выходных данных – `(n_points, features, batch_size)` и `(features, batch_size)`

Далее для повышения устойчивости модели добавлен слой `Dropout`, выход которого подается на `LSTM`, анализирующий частотные последовательности. Слой `BatchNormalization` выравнивает распределение значений перед прохождением полносвязных слоев `Dense` (функция активации `Softmax`) для выделения финальных признаков классификации. На выходе последнего слоя формируются значения вероятностей принадлежности к каждому из 15 классов возможных состояний ИУ.

Обучение осуществлялось в течение 400 эпох при размере батча `batch_size` 32 с использованием оптимизатора `Adam` и функции потерь `sparse_categorical_crossentropy`. Для уменьшения начальной скорости обучения 0.001 применяется класс `ReduceLROnPlateau` из модуля `tensorflow.keras.callbacks`.

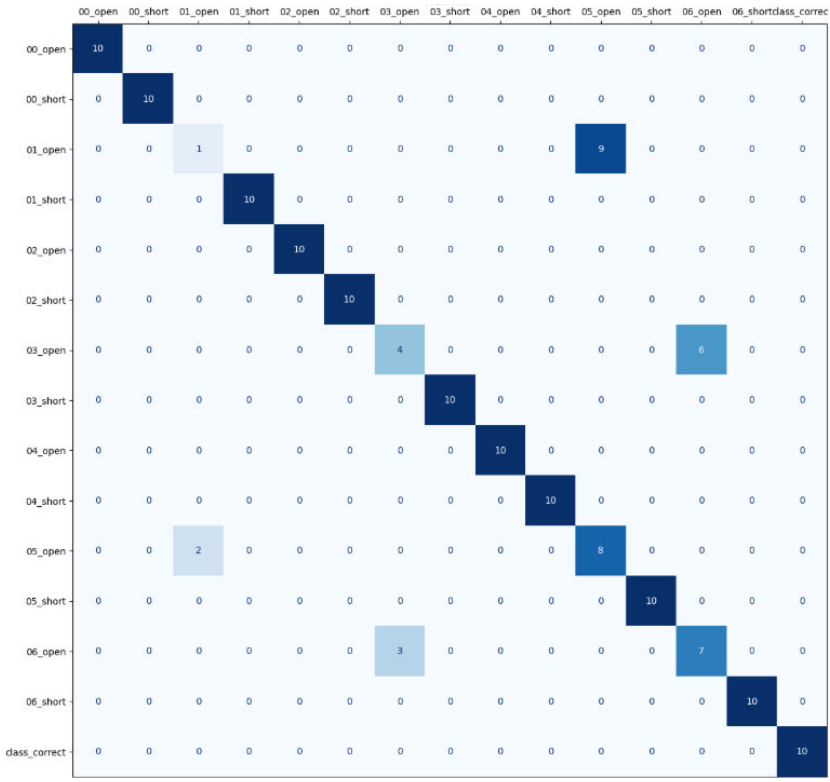
Анализ графиков обучения на тренировочном и валидационном наборах показывает отсутствие переобучения. Точность классификации на обоих наборах постепенно стабилизируется на высоком уровне (рис. 6) в диапазоне значений 0.85-0.87 относительно первых эпох с точностью около 0.3, а ошибка — на низком после приблизительно 200 эпох, что свидетельствует о хорошем обобщении предложенной модели. Для тестирования использовалось 150 экземпляров, по 10 образцов из каждого класса. В результате доля правильно классифицированных состояний ИУ по S-параметрам на тестовой выборке составила 0.86.



а б

Рис. 6. Визуализация процесса обучения: а – точность классификации, б – ошибка классификации

При рассмотрении матрицы ошибок (рис. 7) было определено два явных перепутывания пар классов: отсутствие контактов в L1 и C5, а также в L3 и C6, что связано с высокой схожестью значений S-параметров этих классов неисправностей. Этот случай позволяет рассматривать возможность объединения каждой из пар классов неисправностей в один.



сверху предсказанные классы, слева действительные

Рис. 7. Матрица ошибок

Заключение

В данной работе предложена архитектура нейронной сети для классификации неисправностей пассивных радиочастотных устройств по S-параметрам. Дальнейшее ее развитие позволит создать технологию автоматизированной проверки и отладки, что подразумевает либо полное отсутствие человека, либо его присутствие для выполнений функций контроля. Соответственно применение такой технологии позволит сократить издержки и время производства при крупносерийном выпуске оборудования.

Список литературы

1. Pozar, D. M. Microwave engineering / D. M. Pozar. – 4th ed. – Hoboken : John Wiley & Sons, 2011. – 736 p. – ISBN 978-0-470-63155-3.

6. Kang, T. Y. An empirical study on fault detection and root cause analysis of indium tin oxide electrodes by processing S-parameter patterns / T. Y. Kang, H. Lee, S. Suh // IEEE Transactions on Device and Materials Reliability. – 2024. – Vol. 24, № 3. – P. 380-389. – doi: 10.1109/TDMR.2024.3415049.

7. Scikit-rf Documentation [Электронный ресурс] // Scikit-rf. – Режим доступа: https://scikit-rf.readthedocs.io/en/latest/

8. Module: tf.keras [Электронный ресурс] // TensorFlow. – Режим доступа: https://www.tensorflow.org/api_docs/python/tf/keras

9. Coupling matrix extraction of microwave filters by using one-dimensional convolutional autoencoders / Y. Zhang [et al.] // Frontiers in Physics. – 2021. – Vol. 9. - doi: 10.3389/fphy.2021.716881.

Алгоритм оценки тематической близости на основе взвешенного сравнения тегов

А. А. Полев

Студент магистр

А. А. Вахтин

Доцент

Введение

Современные цифровые платформы, включая социальные сети, медиа–сервисы и электронные библиотеки, сталкиваются с необходимостью эффективной персонализации контента для пользователей. Рекомендательные системы играют ключевую роль в этом процессе, помогая находить релевантные материалы на основе анализа пользовательских предпочтений и семантических характеристик объектов. Одним из наиболее распространённых и универсальных способов описания контента являются теги – ключевые слова или метки, которые в сжатой форме отражают тематическую принадлежность изображений, статей, видео или других типов данных.

Однако традиционные методы сравнения тегов, основанные на бинарном сопоставлении (наличие или отсутствие определённого тега), часто оказываются недостаточно точными, поскольку не учитывают степень важности каждого признака. В реальных сценариях одни теги могут играть более значимую роль в определении тематики, чем другие, что требует введения весовых коэффициентов. Кроме того, при формировании рекомендаций важно не только максимизировать релевантность, но и обеспечивать управляемый уровень разнообразия, чтобы избежать эффекта "эхо–камеры"[1] и расширять интересы пользователя.

В данной работе предлагается алгоритм оценки тематической близости между двумя наборами тегов, учитывающий их весовые характеристики. Такой подход позволяет более гибко определять степень схожести объектов, улучшая точность рекомендаций и адаптируя их под конкретные задачи – будь то строго тематическое соответствие или, напротив, умеренное варьирование контента. Рассматриваемая методика может быть применена в различных

© Полев А. А., Вахтин А. А., 2025

областях, включая анализ текстовых данных, обработку мультимедийного контента и построение интеллектуальных систем рекомендаций.

1. Постановка задачи

В данной статье рассматривается разработка алгоритма, позволяющего сравнивать интересы пользователя с содержанием изображения (или иного объекта), описанного с помощью тегов. Алгоритм должен:

- учитывать совпадение тегов между пользователем и объектом;
- учитывать вес (важность) каждого тега;
- возвращать значение от 0 до 1, где 1 – полное совпадение интересов, 0 – полное различие;
- позволять управлять степенью релевантности при подборе рекомендаций.

2. Алгоритм взвешенного сравнения тегов

Предложенный алгоритм[2] представляет собой модификацию коэффициента Жаккара[3], в которую включены веса тегов.

Введём следующие обозначения:

- T_1 – множество тегов пользователя;
- T_2 – множество тегов изображения;
- $W(t)$ – вес тега t;
- $\sum w(T)$ – сумма весов тегов в множестве T;
- $\sum w(T_1 \cap T_2)$ – сумма весов общих тегов;
- $\sum w(T_1 \cup T_2)$ – сумма весов всех уникальных тегов из двух множеств.

Тогда формула оценки тематической близости выглядит так:

$$Q = \sum w(t), t \in (T1 \cap T2) / \sum w(t), t \in (T1 \cup T2)$$

Данный алгоритм пригоден для реализации на различных языках программирования, включая Java[4]. Ниже представлен код, реализующим алгоритм (см. листинг).

Листинг

Реализация алгоритма

```
private static final double RARE_TAG_BOOST_THRESHOLD = 1.3; //
Порог веса для редкого тэга
    private static final double RARE_TAG_BOOST_SCORE = 0.7; //
Балл, присуждаемый, если редкий тэг совпал
    Map<String, Double> tagWeights = Map.of(
            "anime", 0.9,
            "girl", 0.6,
            "vehicle", 1.0,
```

Окончание листинга

```
            "car", 0.8,
            "jdm", 1.1,
            "motorcycle", 1.0,
            "retro", 1.3,
            "text", 0.4
    );

    public double weightedSimilarity(Set<String> imageTags,
Set<String> userInterestTags) {
        double intersectionWeight = imageTags.stream()
                .filter(userInterestTags::contains)
                .mapToDouble(tag ->
tagWeights.getOrDefault(tag, 1.0))
                .sum();

        double unionWeight = imageTags.stream()
                .mapToDouble(tag ->
tagWeights.getOrDefault(tag, 1.0))
                .sum()
                + userInterestTags.stream()
                .mapToDouble(tag ->
tagWeights.getOrDefault(tag, 1.0))
                .sum()
                - intersectionWeight;

        double baseScore = (unionWeight == 0) ? 0.0 :
intersectionWeight / unionWeight;

        // Rare Tag Boost: если хотя бы один общий тег
является "редким" и его вес >= порога, то применяем буст
        boolean hasRareBoost = imageTags.stream()
                .filter(userInterestTags::contains)
                .anyMatch(tag -> tagWeights.getOrDefault(tag,
1.0) >= RARE_TAG_BOOST_THRESHOLD);

        if (hasRareBoost && baseScore < RARE_TAG_BOOST_SCORE)
{
            return RARE_TAG_BOOST_SCORE;
        }

        return baseScore;
    }
```

3. Пример работы алгоритма

Для демонстрации работы алгоритма и применения механизма редких, но релевантных тегов (Rare Tag Boost) приведены конкретные примеры на основе заданного набора тегов, указанных в листинге.

Для примера, порогом включения в рекомендации будет значение большее 0.5, а порогом значимости для редкого релевантного тега (Rare Tag Threshold) будет значение большее 1.3.

Пример расчета с совпадением тегов и применением Rare Tag Boost представлен ниже, на рис. 1.

```
Теги изображения: ["motorcycle", "vehicle", "rider", "retro"]
Совпадающие: "vehicle", "retro"
Расчёт:
Вес пересечения: vehicle (1.0) + retro (1.3) = 2.3
Вес объединения:
userTags: vehicle (1.0) + retro (1.3) + anime (0.9) = 3.2
imageTags: motorcycle (1.0) + vehicle (1.0) + rider (0.9) + retro (1.3) = 4.2
unionWeight = 3.2 + 4.2 - 2.3 = 5.1
Q = 0.45098039215686275
```

Рис. 1. Результат расчета первого примера

Вывод – хотя значение Q получилось меньше 0.5, тег retro имеет вес 1.3, что позволяет включить изображение в рекомендации по механизму Rare Tag Boost.

Для повышения разнообразия в подборке добавлено дополнительное правило: если значение Q ниже порога, но есть совпадение хотя бы по одному тегу с весом более RARE_TAG_BOOST_THRESHOLD (в нашем случае 1.3), объект включается в рекомендацию. Это позволяет учитывать редкие, но значимые интересы пользователя даже при общей низкой схожести.

Далее, на рис. 2, представлен пример расчета при почти полном различии тегов.

```
Теги изображения: ["car", "skyline", "vehicle", "jdm"]
Совпадающие: "vehicle"
Расчёт:
Вес пересечения: vehicle (1.0) = 1.0
Вес объединения: 6.1
Q = 1.0 / 6.1 = 0.1639344262295082
```

Рис. 2. Результат расчета второго примера

Вывод – значение Q меньше 0.5 и совпавший тег имеет вес менее 1.3, а значит, изображение не включается в рекомендации.

Заключение

В статье представлен подход к построению системы рекомендаций объектов на основе взвешенного сравнения тегов, отражающих интересы пользователя и тематику контента. Алгоритм использует модифицированный коэффициент Жаккара, учитывающий важность тегов, что позволяет точнее выявлять релевантные объекты.

Дополнительно реализован механизм Rare Tag Boost, позволяющий учитывать редкие интересы пользователя при низкой общей схожести. Это делает систему более чувствительной к индивидуальным предпочтениям и повышает разнообразие рекомендаций.

В дальнейшем алгоритм может быть расширен за счёт обучения весов тегов на основе пользовательских взаимодействий или добавления контекстных фильтров, а также за счёт применения методов оптимизации производительности для работы с большими объёмами данных [5].

Список литературы

1. Что такое алгоритмы? Как предотвратить эхо-камеры и обеспечить безопасность в Интернете [Электронный ресурс] : сайт. – Режим доступа: https://www.internetmatters.org/ru/hub/news-blogs/what-are-algorithms-how-to-prevent-echo-chambers/

2. Алгоритмы для рекомендательных систем [Электронный ресурс] : сайт. – Режим доступа: https://sky.pro/wiki/python/algoritmy-i-modeli-dlya-rekomendatelnyh-sistem/

3. Семантический поиск: от простого сходства Жаккара к сложному SBERT [Электронный ресурс] : сайт. – Режим доступа: https://habr.com/ru/companies/skillfactory/articles/566414/

4. Руководство по языку программирования Java [Электронный ресурс] : сайт. – Режим доступа: https://metanit.com/java/tutorial/

5. Четыре способа оптимизации ПО [Электронный ресурс] : сайт – Режим доступа: https://habr.com/ru/companies/ruvds/articles/775868/

Новый подход к организации программной архитектуры гибридного вычислительного кластера ФКН ВГУ

А. Г. Потапов

Студент магистр

А. В. Романов

Старший преподаватель

Введение

После окончания поддержки наиболее популярных для суперкомпьютеров операционных систем CentOS 7 [1] и CentOS 8 многие коллективы вычислительных центров оказались перед проблемой смены программного обеспечения для своих компьютерных кластеров. Основные проблемы в этом случае связаны с поддержкой такого оборудования, как сопроцессор Intel Xeon Phi, интерконнект Mellanox Infiniband ConnectX-3 и GPU-ускорители NVIDIA Tesla ранних версий. К сожалению, наличие таких девайсов фактически исключает возможность применения на вычислительном кластере современного программного обеспечения без существенной потери производительности машины. Единственным решением в данном случае является использование операционных систем из числа снятых с поддержки, но допускающих использование последних версий драйверов для такого оборудования.

В данной статье рассматривается подход к организации архитектуры программного обеспечения кластера ФКН ВГУ с учётом наличия на борту сопроцессоров Xeon Phi и NVIDIA Tesla K80, а также оптоволоконного интерконнекта InfiniBand Mellanox ConnectX-3.

Статья организована следующим образом. В первой части даётся описание программно-аппаратных средств кластера на момент окончания поддержки операционной системы CentOS 7. Во второй части дано описание ограничений на выбор программных средств вследствие наличия в кластере оборудования, снятого с официальной поддержки. В третьей части будет предложена новая программная архитектура с учётом возможностей и потребностей суперкомпьютерного центра ВГУ.

1. Изначальная структура кластера

Аппаратная структура вычислительного кластера ФКН ВГУ имеет следующий вид. Вычислительное поле состоит из 10 узлов, на каждом

из которых установлено два 12-ядерных процессора Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz. Первые 7 узлов доукомплектованы двумя сопроцессорами Intel(R) Xeon(R) Phi 7120P, содержащих по 61 ядру, способному работать в четырёхпоточном режиме. Таким образом, общее число потоков может достигать 244. Узлы 8-10 содержат по два графических ускорителя NVIDIA Tesla K80 с 4992 CUDA-ядрами. Управление работой кластера осуществляет головной узел с двумя 6-ядерными процессорами Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz. Кроме того, вход на суперкомпьютер осуществляется через шлюз, на котором установлено два 6-ядерных процессора Intel(R) Xeon(R) CPU X5650 @ 2.67GHz.

Дисковое хранилище кластера включает RAID-10 с файловой системой ZFS, состоящая из 8 HDD 4TB 7200. Поверх ZFS развёрнута сетевая файловая система BeegFS, которая используется для хранения пользовательских данных и математических пакетов. Четыре аналогичных диска используются для системного резервного копирования. На каждом из вычислительных узлов установлены SSD Intel 256GB. Кроме того, кластер располагает оперативной памятью суммарным объёмом в 1.2 TB.

Интерконнект кластера представлен оптоволоконной сетью Mellanox Technologies MT27500 Family [ConnectX-3] 56GB/s.

На кластере была установлена операционная система CentOS 7. Загрузка вычислительных узлов осуществлялась по сети, при этом семь из них загружали ядро, поддерживающее Intel Xeon Phi, а три других – ядро с поддержкой GPU NVIDIA Tesla. В качестве очереди исполнения используется SLURM.

Схематичное изображение изначальной структуры кластера представлено на рис. 1 [2].

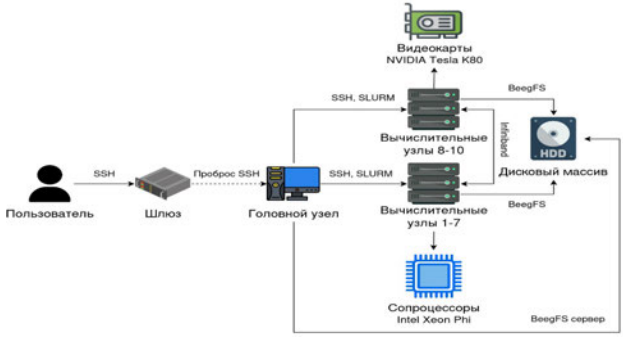


Рис. 1. Изначальная структура кластера ФКН ВГУ

2. Выбор подходящей операционной системы

В состав кластера входят математические сопроцессоры Xeon Phi в PCI-E исполнении. Однако в 2018 году компания Intel официально прекратила поддержку данной платформы, вследствие чего версия стека программного обеспечения и драйверов фактически ограничена версией MPSS 3.8.6. В отличие от видеокарт NVIDIA Tesla, драйвера сопроцессоров Xeon Phi требуют тесной интеграции с ядром операционной системы Linux и перестают работать при малейших его изменениях. Последняя версия ядра, на которой можно установить драйвер для Xeon Phi – 4.4 [3].

В документации к MPSS-стеку в число официально поддерживаемых операционных систем входит Red Hat Enterprise Linux версий 6.7 – 7.3 (ядра 2.6 – 3.10) и SUSE Linux Enterprise Server [4] с версии 11 SP4 по 12 SP2 (ядра 3.0 – 4.4). Вследствие вышеописанных ограничений развёртывание подсистемы Xeon Phi на системах Linux с версией ядра выше 4.4 успеха не имеет.

Вторым ограничивающим фактором является интерконнект кластера семейства ConnectX-3, который ограничивает использование драйверов OFED версией 5.1 [5], что вносит свои коррективы в выбор операционной системы. В качестве примера можно привести попытку развёртывания подсистемы Xeon Phi на системе Ubuntu версий 18.04 LTS и 20.04 LTS, которые являются максимально доступными версиями Ubuntu с поддержкой OFED данной версии. Для этих двух операционных систем в Интернете можно найти неофициальные патчи, обеспечивающие возможность установки MPSS для ядер этих операционных систем. Тем не менее, попытки использования этого патча успеха не имели. Кроме того, настройка InfiniBand на системе 20.04 была сопряжена с дополнительными сложностями, связанными с переходом между режимами Datagram и Connected mode.

В результате было решено отклонить варианты использования официально не поддерживаемых MPSS систем и остановить свой выбор на SUSE Linux Enterprise Server 12 SP2, которая содержит более новое ядро 4.4 по сравнению с используемым 3.10, а также обладает мощным набором инструментов для администрирования в программе YaST.

3. Новый подход к организации программной архитектуры

Существует два возможных подхода к развёртыванию операционной системы на вычислительных узлах. В первом случае загрузка происходит по сети со специально выделенного раздела на жёстких дисках, что обеспечивает синхронизацию системы, упрощает её администрирование и сопровождение, так как все операции происходят на головном узле.

Однако вследствие того, что возможность обновления операционной системы SLES 12 SP2 исчерпана, а количество узлов невелико, острая необходимость в использовании данного подхода отсутствует. Следовательно, было решено использовать имеющиеся SSD-диски для загрузки каждого из вычислительных узлов в отдельности.

Отдельного внимания заслуживает выбор распределённой файловой системы для обеспечения совместного доступа узлов к пользовательским данным и некоторым математическим пакетам. С одной стороны, использование настроенного в данный момент сервера BeegFS позволяет получить максимальную производительность и отлично подходит для параллельных вычислительных систем.

Однако его использование не позволяет подключить к общему файловому хранилищу сопроцессоры Xeon Phi, поскольку не входит в комплект программного обеспечения, доступного для установки на архитектуре Knights Corner. Единственным возможным решением для них является файловый сервер NFS, однако поддержка двух файловых серверов одновременно может привести к потере производительности и неоправданному усложнению структуры разделов на массиве жёстких дисков. Следовательно, наиболее вероятным на данный момент является использование в будущем исключительно NFS-сервера, способного обеспечить равноправный доступ всех компонентов к общим ресурсам и возможность использования Xeon Phi в Native-режиме с выделением для них отдельного раздела в SLURM.

Также одним из наиболее значимых решений в новой структуре кластера может стать значительное расширение функций шлюза посредством использования технологии виртуализации. Использование вычислительных мощностей данного компонента будет направлено на создание нескольких виртуальных служебных узлов, отвечающих за поддержание работы файлового сервера, компиляцию программ, управление заданиями и учётными записями пользователей. Функция же головного узла при этом редуцируется до файлового хранилища.

Схематичное изображение новой структуры кластера даёт рис. 2.

Заключение

В ходе проделанной работы было установлено, что наиболее оптимальным вариантом архитектурно-программного обеспечения, которое позволит в дальнейшем осуществлять эксплуатацию кластера ФКН ВГУ, является установка операционной системы SLES 12 SP 2, организация файлового сервера на базе бывшего главного узла суперкомпьютера и развёртывание виртуальных узлов управления компиляцией и распределённой файловой системы на базе сервера, выполняющего в данный момент функции шлюза.

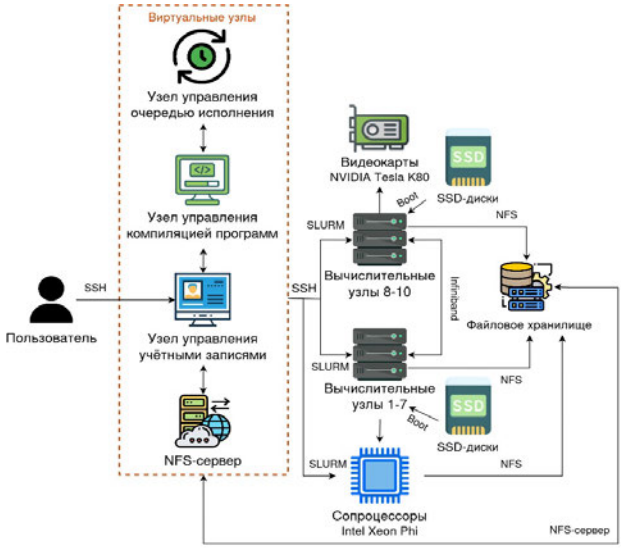


Рис. 2. Новая структура кластера ФКН ВГУ

Программное обеспечение новых версий, необходимое для проведения научных вычислений и обучения студентов ФКН ВГУ, планируется собирать из исходных кодов на отдельном разделе жёсткого диска независимо от инструментария базовой операционной системы. Выбранный подход позволит сохранить функциональность и производительность вычислительного кластера ВГУ, упростить его администрирование и сопровождение программного обеспечения.

Список литературы

1. What to know about CentOS Linux EOL [Электронный ресурс]. – Режим доступа: https://www.redhat.com/en/topics/linux/centos-linux-eol

2. Flaticon [Электронный ресурс]. – Режим доступа: https://www.flaticon.com/ru/

3. Релиз ядра Linux 4.4 [Электронный ресурс]. – Режим доступа: https://www.opennet.ru/opennews/art.shtml?num=43652

4. SUSE Linux Enterprise Server 12 SP2. Release Notes: [Электронный ресурс]. – Режим доступа: https://www.suse.com/releasenotes/x86_64/SUSE-SLES/12-SP2/index.html

5. General Support in MLNX_OFED [Электронный ресурс. – Режим доступа: https://docs.nvidia.com/networking/display/ofedv490170/

6. general+support+in+mlnx_ofed

Реализация универсального парсинга XML-документов в сфере госзакупок

А. В. Ролдугин

Студент бакалавр

К. А. Маковий

Доцент

Введение

Цифровая трансформация государственного сектора стимулирует переход к модели «открытого государства», где качественные, полные и своевременные данные являются ключевым ресурсом повышения прозрачности и эффективности управления. Особое место в этом контексте занимают открытые данные о государственных закупках, объём которых в Российской Федерации насчитывает сотни гигабайт.

Однако практическая работа с этими данными сталкивается с рядом технологических сложностей. Информация о закупках публикуется в виде XML-документов, причём структура файлов регулярно изменяется: появляются новые версии XSD-схем (XML Schema Definition), добавляются новые атрибуты, вводятся дополнительные уровни вложенности. Для сбора данных применяется трехуровневый процесс “извлечение-преобразование-загрузка” (Extract-Transform-Load – ETL). Однако такие ETL-процессы, полагающиеся на «жёстко» запрограммированные настройки отображения элементов XML в столбцы таблицы БД, часто устаревают и требуют постоянного ручного обслуживания. Это приводит к задержкам в загрузке новых пакетов данных и росту сложности поддержки программного обеспечения.

Цель настоящего исследования — разработка гибкого парсера, способного обрабатывать XML-документы любой структуры, поступающие из Единой информационной системы в сфере закупок (далее – ЕИС). В качестве промежуточной структуры хранения данных после парсинга будет использоваться способ хранения по модели EAV (Entity Attribute Value) [1], отражающий древовидную структуру документов и не накладывающий ограничений на глубину вложенности или состав элементов. Следующим шагом, согласно документации ЕИС

© Ролдугин А. В., Маковий К. А., 2025

[2], будет произведена трансформация данных из промежуточной структуры в реляционную модель хранения для целей дальнейшего интеллектуального анализа данных

1. Анализ существующих подходов

Классический подход к извлечению данных из XML опирается на средства автогенерации кода. Например, с помощью JAXB [3, 4] для каждого семейства документов создается доменный слой из java-классов, сгенерированный по XSD. Также проектируются нормализованные SQL-таблицы, хранящие извлеченные данные. Достоинством является высокая читабельность кода и эффективность запросов. Основной недостаток — модификация схемы: добавление нового атрибута или тега потребует изменения кода.

Существует также ряд промышленных средств по генерации преобразований из XSD в реляционную модель. Например, инструменты ORACLE XML DB [5] и IBM DB2 PureXML [6] способны эффективно хранить и работать с XML, но требуют фиксации версии XSD. Кроме того, в исследованиях 2024 гг. встречаются подходы на основе машинного обучения — например, построение решающих деревьев на основе XPath-структур [7] и алгоритмы автоматического сопоставления по семантической близости [8]. Несмотря на интересные результаты, эти методы нуждаются в значительном объёме размеченных примеров.

Таким образом, существующие подходы обладают рядом существенных проблем:

− большое количество настроек, которые необходимо производить вручную;

− количество генерируемых классов быстро растет с выходом новых схем;

− сложность обучения моделей машинного обучения.

2. Постановка задачи

Предметной областью настоящей работы являются открытые XML-документы, публикуемые в ЕИС. По состоянию на май 2025г. в обращении находилось около 200 [9] актуальных семейств XML-документов, соответствующих различным XSD-схемам. Средняя глубина вложенности составляет 3 уровня, используются атрибуты, пространства имен.

Ключевыми функциональными требованиями к проектируемому решению являются:

− поддержка любых новых схем без ручной правки кода;

− извлечение полного дерева элементов и атрибутов с сохранением иерархии;

− масштабируемость по числу файлов и разнообразию структур.

Формализация задачи

Пусть задана совокупность XML-документов $D = \{d_1, d_2, ..., d_n\}$, каждый из которых валиден относительно некоторой XSD-схемы S_i. Требуется построить функцию $P : D \to T$, где T — реляционная таблица в EAV-формате, такая, что для любого d_i:

1. Каждый элемент $e \in d_i$ представляется как запись
   $(r, f, p, n, v, t) \in T$,

где:

r — код Record_ID, генерируемый на основе положения элемента в дереве; f — идентификатор File_ID, привязанный к d_i; p — XPath-путь от корня до e_i; n, v, t — имя, значение и тип извлеченного элемента e.

2. Для любой допустимой модификации XSD S_i' функция должна адаптироваться без изменения программного кода, при условии доступности схемы в момент парсинга.
3. Результирующая таблица T должна быть пригодна для восстановления исходной структуры d_i с точностью до порядка элементов и namespace-меток.

Модель хранения данных

Для хранения результатов парсинга используется промежуточная таблица raw_xml следующей структуры:

Таблица

Структура SQL-таблицы raw_xml

Поле	*Описание*
record_id	Иерархический идентификатор, отражающий положение элемента в дереве XML (например, '001-002-001').
file_id	Уникальный идентификатор исходного XML-документа
path	Полный XPath к тегу или атрибуту (например, '/contract/customer/address/@zip')
element_name	Имя XML-тега или атрибута
element_value	Значение в строковом представлении
type	Тип элемента: тег или атрибут

Выбор конкретных полей модели обусловлен необходимостью точного и однозначного представления вложенной XML-структуры.

Поле record_id служит для идентификации узла в контексте древовидной иерархии документа. Оно кодируется на основе позиции элемента относительно родительских узлов, что позволяет воссоздавать не только порядок, но и структуру вложенности. Кроме того, в целях оптимизации операций соединения (join) строк будет использоваться выравнивание каждого уровня иерархии до 3 символов, исходя из предположения, что в документах не встречается 1000 и более тегов.

Поле file_id связывает конкретную запись с исходным XML-документом. Это позволит хранить всю необходимую метаинформацию о поступающих файлах.

Поле path содержит абсолютный XPath к тегу или атрибуту.

Имена и значения элементов (поля element_name и element_value) обеспечивают хранение содержательной информации, при этом значения приводятся к строковому представлению, что унифицирует процесс обработки.

Наконец, поле type фиксирует тип обработанной записи, различая элементы и атрибуты.

Таким образом, предложенный подход обеспечивает:

− высокую масштабируемость – XML-документ любой структуры может быть сохранен в универсальную таблицу БД;

− независимость от конкретной XSD-схемы.

3. Программная реализация

Ключевая идея модуля парсинга, который осуществляет трансформацию XML-документа в EAV-представление — рекурсивный обход дерева XML с одновременным построением иерархического пути и сохранением информации о каждом элементе и атрибуте.

Основные этапы алгоритма:

1. Загрузка и разбор XML-файла: использование SAX (Simple API for XML) парсера для получения корневого узла XML-документа.
2. Начало итеративной обработки:

− Обход начинается с корневых тегов.

− Каждому элементу присваивается иерархический record_id.

3. Формирование XPath-пути к элементу.
4. Извлечение содержимого элемента (если есть).
5. Рекурсивная обработка дочерних тегов.

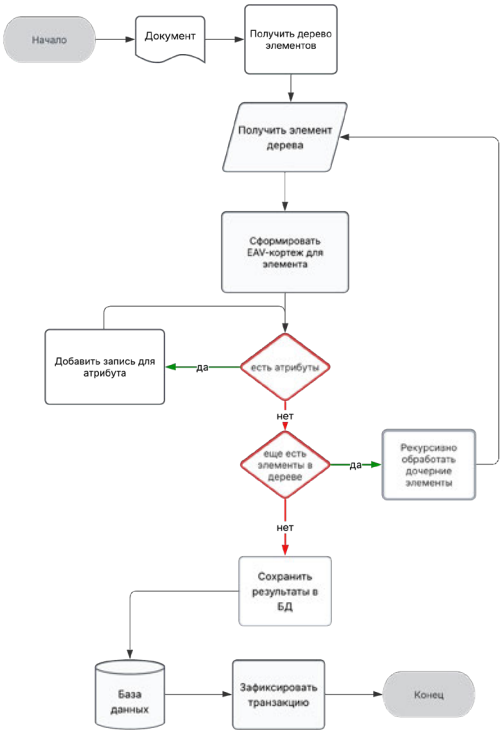


Рис. 1. Блок-схема алгоритма парсинга XML-документов ЕИС

Извлечённые записи далее передаются в слой хранения, реализованный в виде реляционной СУБД.

Листинг 1

DDL-запрос для создания промежуточной таблицы

```
CREATE TABLE core.raw_xml (
    record_id     TEXT NOT NULL,
    file_id       BIGINT NOT NULL,
    path          TEXT NOT NULL,
    element_name  TEXT NOT NULL,
    element_value TEXT,
    type          TEXT NOT NULL CHECK (type IN ('element',
'attribute'))
) PARTITION BY HASH (file_id);

-- Анонимная функция для создания 32 секций
DO $$
```

```
BEGIN
  FOR i IN 0..31 LOOP
    EXECUTE format(
      'CREATE TABLE core.raw_xml_p%s PARTITION OF core.raw_xml
FOR VALUES WITH (MODULUS 32, REMAINDER %s);',
      i, i
    );
  END LOOP;
END $$;

-- глобальные индексы (PostgreSQL 15+)
CREATE INDEX idx_raw_xml_el_name ON raw_xml (element_name);
-- покрывающий индекс
CREATE INDEX raw_xml_ix ON raw_xml (path, record_id);
```

В листинге 1 представлен DDL-запрос для СУБД PostgreSQL для создания промежуточной таблицы core.raw_xml, предназначенной для хранения разобранных элементов XML-документа. Созданная таблица секционирована по полю file_id для равномерного распределения данных по 32 секциям. Также созданы индексы по полям path, element_name, record_id. Это обеспечивает высокую производительность запросов, что критически важно при обработке большого массива данных, поступающих из ЕИС.

4. Полученные результаты

На рис. 2 представлена структура XML-документа, соответствующего уведомлению об отзыве заявки в рамках процедуры государственных закупок. Документ включает в себя информацию о номере закупки (purchaseNumber), идентификаторе документа (docNumber), дате публикации в ЕИС (docPublishDTInEIS), а также ссылки на подробную информацию в ЕИС и на внешнем ресурсе. Все элементы снабжены пространствами имён.

```
<ns2:epNoticeApplicationCancel schemeVersion="13.0">
    <ns9:commonInfo>
        <ns9:purchaseNumber>0131100004523000003</ns9:purchaseNumber>
        <ns9:docNumber>ИЭА1</ns9:docNumber>
        <ns9:docPublishDTInEIS>2023-02-03T08:32:29.225+03:00</ns9:docPublishDTInEIS>
        <ns9:href>https://zakupki.gov.ru/epz/order/notice/ea20/view/common-info.html?regNumber=0131100004523000003</ns9:href>
    </ns9:commonInfo>
    <ns9:extPrintFormInfo> ···
    </ns9:extPrintFormInfo>
    <ns9:applicationInfo>
        <ns9:appNumber>185</ns9:appNumber>
        <ns9:appDT>2023-01-31T09:31:44+03:00</ns9:appDT>
        <ns9:appRating>5</ns9:appRating>
    </ns9:applicationInfo>
    <ns9:protocolInfo>
        <ns9:protocolNumber>1</ns9:protocolNumber>
        <ns9:protocolNumberExternal>0131100004523000003-2</ns9:protocolNumberExternal>
        <ns9:versionNumber>1</ns9:versionNumber>
    </ns9:protocolInfo>
    <ns9:hrefExternal>https://www.sberbank-ast.ru/ViewDocument.aspx?id=958765163</ns9:hrefExternal>
</ns2:epNoticeApplicationCancel>
```

Рис. 2. Исходный xml-документ

Результат работы универсального XML-парсера демонстрирует (рис. 3) извлечение всех тегов и атрибутов в EAV-представление.

record_id	file_id	path	element_name	element_value	type
001	1000	epNoticeApplicationCancel@schemeVersion	schemeVersion	13.0	attribute
001/001/001	1000	epNoticeApplicationCancel/commonInfo/purchaseNumber	purchaseNumber	0131100004523000003	element
001/001/002	1000	epNoticeApplicationCancel/commonInfo/docNumber	docNumber	ИЗА1	element
001/001/003	1000	epNoticeApplicationCancel/commonInfo/docPublishDTInEIS	docPublishDTInEIS	2023-02-03T08:32:29.225+03:00	element
001/002/003	1000	epNoticeApplicationCancel/extPrintFormInfo/fileType	fileType	xml	element
001/003/001	1000	epNoticeApplicationCancel/applicationInfo/appNumber	appNumber	185	element
001/003/002	1000	epNoticeApplicationCancel/applicationInfo/appDT	appDT	2023-01-31T09:31:44+03:00	element

Рис. 3. Результат парсинга

Каждая строка таблицы соответствует отдельному тегу или атрибуту исходного документа и содержит следующие поля: идентификатор записи (Record_ID), идентификатор файла (File_ID), путь к элементу в иерархии XML (Path), имя элемента (Element_Name) и его значение (Element_Value). Такой подход обеспечивает унифицированное хранение данных вне зависимости от структуры исходного XML и позволяет реализовать гибкий механизм поиска и анализа информации.

Заключение

В данной работе была рассмотрена методология гибкого парсинга открытых данных в сфере государственных закупок, способная масштабироваться на сотни разнородных XML-документов без необходимости изменения программного кода. Ключевым элементом решения выступает EAV-модель, где минимальный, но информативный набор полей (Record_ID, File_ID, Path, Element_Name, Element_Value, Type) обеспечивает сохранность иерархического контекста и стабильность при модификации схем.

Были даны рекомендации по проектированию таблицы в РСУБД с применением секционирования и индексирования столбцов для повышения эффективности запросов.

Предложенный модуль парсинга готов для интеграции с ЕИС для получения данных. Однако в качестве дальнейшего направления исследования можно рассмотреть возможность расширения функциональности и на другие типы данных древовидной структуры (например, JSON).

Список литературы

1. Żabiński K., Zielosko B. Improved eav-based algorithm for decision rules construction //Entropy. – 2023. – V. 25. – №. 1. – P. 91.

2. Альбом ТФФ ЕИС версии 15.2 [Электронный ресурс] – Режим доступа:

https://zakupki.gov.ru/epz/main/public/download/downloadDocument.html?id=42185

3. Java. Работа с XML-документами [Электронный ресурс] : Хабр. – Режим доступа: https://habr.com/ru/companies/simbirsoft/articles/704818/

4. Java Architecture for XML Binding [Электронный ресурс]: ORACLE Java Documentation – Режим доступа: https://docs.oracle.com/javase/8/docs/technotes/guides/xml/jaxb/index.html

5. Oracle Corporation. Oracle XML DB Developer's Guide. Oracle® Database 19c Documentation [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/19/adxdb/index.html

6. IBM. DB2 11.1: PureXML Guide [Электронный ресурс] – IBM Redbooks, 2016. – Режим доступа: https://www.redbooks.ibm.com/abstracts/sg247915.html

7. Sheetrit, E., Brief, M., Mishaeli, M., Elisha, O. ReMatch: Retrieval Enhanced Schema Matching with LLMs [Электронный ресурс] // arXiv preprint arXiv:2403.01567. – 2024. – Режим доступа: https://arxiv.org/abs/2403.01567

8. SMUTF: Schema Matching Using Generative Tags and Hybrid Features [Электронный ресурс] : Information Systems. – 2025. – P. 102570 // arXiv preprint arXiv:2402.01685. – Режим доступа: https://arxiv.org/abs/2402.01685

9. Интеграционные схемы ЕИС версии 15.2 [Электронный ресурс] – Режим доступа: https://zakupki.gov.ru/epz/main/public/download/downloadDocument.html?id=42187

Сравнительный анализ колоночных и реляционных СУБД при выполнении агрегатных запросов

А. Г. Сапелкин

Студент магистр

Н. К. Самойлов

Старший преподаватель

Введение

В настоящее время обработка больших объёмов данных требует использования высокопроизводительных систем управления базами данных, способных эффективно выполнять аналитические и агрегатные запросы. Одним из ключевых факторов, влияющих на производительность таких запросов, является архитектура хранения данных. Наиболее распространёнными подходами к организации хранения информации в СУБД являются строковая и колоночная архитектуры.

Реляционные СУБД, использующие строковую организацию, исторически являются доминирующими решениями для хранения и обработки данных. Они обеспечивают высокую эффективность при выполнении транзакционных операций, однако в условиях выполнения большого количества аналитических запросов, особенно агрегатного характера, их производительность может быть ограничена из-за особенностей доступа к данным и способа их хранения.

Колоночные СУБД, напротив, ориентированы на аналитические нагрузки и обеспечивают более высокую скорость выполнения агрегатных операций за счёт хранения данных по столбцам. Это позволяет значительно сократить объём читаемых данных, использовать эффективные схемы сжатия, а также оптимизировать использование кэш-памяти процессора.

В данной работе проводится сравнительный анализ производительности агрегатных запросов в колоночных и строковых СУБД. В качестве представителя колоночных СУБД рассматривается система ClickHouse, широко применяемая в задачах аналитической обработки.

© Сапелкин А. Г., Самойлов Н. К., 2025

1. Колоночные СУБД и ClickHouse

Колоночные СУБД хранят данные по столбцам, в отличие от традиционных реляционных систем, где данные сохраняются построчно. Это даёт преимущество при аналитических и агрегатных запросах, особенно при работе с большими объёмами данных.

В колоночных СУБД данные каждого столбца хранятся последовательно, что позволяет считывать только необходимые столбцы, снижая объём операций ввода-вывода. В реляционных СУБД же при обработке запроса считывается вся строка, даже если в запросе участвует только один столбец, что увеличивает нагрузку на систему хранения.

Колоночная архитектура также эффективна с точки зрения сжатия данных, поскольку в каждом столбце хранятся данные одного типа. Это упрощает использование алгоритмов сжатия, что снижает объём хранимой информации и ускоряет обработку запросов.

Колоночные СУБД особенно эффективны при аналитических запросах, таких как агрегирование и фильтрация. Однако при выполнении операций, требующих обновления строк, они могут уступать реляционным СУБД.

ClickHouse представляет собой высокопроизводительную колоночную СУБД, ориентированную на аналитические запросы в реальном времени. Она не поддерживает полную ACID-транзакционность, что объясняется её фокусом на аналитические задачи [1].

ClickHouse использует множество оптимизаций, таких как сжатие данных, индексирование, эффективная работа с in-memory структурами и параллельное выполнение запросов. Система поддерживает вертикальное и горизонтальное масштабирование, что позволяет использовать её в распределённых средах с высокой нагрузкой.

К числу недостатков колоночных СУБД, включая ClickHouse, относится, во-первых, их неэффективность при выполнении операций обновления и удаления. В ClickHouse поддержка таких операций ограничена. Изменения данных выполняются не напрямую, а через механизм «мягких» удалений с последующим фоновым перерасчетом данных, что приводит к задержкам и увеличенному потреблению ресурсов. В большинстве случаев рекомендуется использовать подход с вставкой новой версии данных, а не их модификацией.

Во-вторых, в ClickHouse отсутствует полноценная поддержка ACID-транзакций. Это означает, что система не гарантирует атомарность, согласованность, изолированность и долговечность в той степени, в какой это делают традиционные реляционные СУБД. Такая

архитектура оправдана в аналитических задачах, но делает систему неприменимой в критически важных транзакционных сценариях.

В-третьих, в ClickHouse отсутствует поддержка операции обновления строк в традиционном смысле. Вместо этого для внесения изменений используется механизм мутаций, при котором создаётся новая версия данных, а старая версия помечается как устаревшая и удаляется фоновыми процессами. Этот подход позволяет поддерживать высокую производительность на чтение, однако обновления требуют перезаписи целых блоков данных, что может быть ресурсоёмким процессом.

Кроме того, ограниченная поддержка внешних ключей, триггеров и других механизмов обеспечения целостности данных также относится к числу недостатков ClickHouse. Это снижает возможности моделирования сложных взаимосвязей между таблицами и требует переноса части логики обеспечения целостности на уровень приложения.

2. Сравнительный анализ эффективности работы колоночных и реляционных СУБД

Для проведения сравнительного анализа производительности агрегатных запросов в колоночных и классических строковых реляционных СУБД были созданы две идентичные по структуре таблицы, реализованные соответственно в ClickHouse и PostgreSQL.

Ниже приводится пример запросов по созданию таблицы в PostgreSQL (листинг 1) и ClickHouse (листинг 2).

Листинг 1

Создание таблицы в PostgreSQL

```
CREATE TABLE customers (
    id SERIAL PRIMARY KEY,
    name TEXT,
    date_created TIMESTAMP DEFAULT NOW(),
    tags TEXT[]
);
```

Листинг 2

Создание таблицы в ClickHouse

```
CREATE TABLE customers (
    id UInt32,
    name String,
    date_created TIMESTAMP DEFAULT now(),
    tags Array(String),
    PRIMARY KEY (id)
) ENGINE = MergeTree();
```

Ключевым отличием при создании таблиц является указание специализированного движка MergeTree в ClickHouse, который отвечает за эффективное хранение и обработку данных в столбцовом формате, оптимизируя выполнение аналитических и агрегатных запросов. Данный движок обеспечивает эффективное индексирование и высокую скорость чтения больших массивов данных за счёт их последовательного хранения по столбцам, в отличие от PostgreSQL, который по умолчанию хранит данные построчно и ориентирован на транзакционные операции [2].

В обеих таблицах было предварительно сгенерировано и сохранено по 100 миллионов записей, содержащих идентификаторы, имена, даты создания записей и наборы тегов. Без дополнительных настроек баз данных перед созданием и заполнением таблиц, данные на диске занимают 2 и 15 гигабайтов в ClickHouse и PostgreSQL соответственно.

Для иллюстрации структуры данных и проверки корректности заполнения таблиц можно выполнить выборку первых трех записей при помощи аналогичного для ClickHouse и PostgreSQL запроса «SELECT * FROM customers LIMIT 3».

Результаты данной выборки представлены в таблице ниже.

Таблица

Фрагмент содержания созданной таблицы customers

id	name	date_created	tags
0	customer_0	2025-02-12 18:19:53	['tag84', 'tag19', 'tag51', 'tag91', 'tag1', 'tag34', 'tag98', 'tag49']
1	customer_1	2025-02-13 11:31:12	['tag94', 'tag58', 'tag32', 'tag97', 'tag15', 'tag64', 'tag40', 'tag6', 'tag94']
2	customer_3	2025-02-14 18:20:42	['tag70', 'tag43', 'tag12', 'tag1', 'tag49']

Наполнение атрибута tags производилось автоматически при помощи запроса и PLpgSQL скрипта в ClickHouse и PostgreSQL соответственно таким образом, что атрибут содержит случайное количество записей (от 1 до 10), каждый элемент которого имеет вид «tagN», где N – случайное число от 1 до 100.

Для оценки эффективности выполнения агрегатных запросов были проведены тестовые замеры с использованием идентичного по логике и структуре агрегатного запроса, включающего группировку данных по дате создания, подсчёт количества записей и вычисление среднего значения длины имени. Данный запрос (листинг 3) одновременно сочетает операции группировки, агрегации и сортировки результатов по убыванию даты создания.

Листинг 3

Пример запроса с агрегацией, группировкой и сортировкой

```
select date_created, count(*), avg(length(name)) from
customers
group by date_created
order by date_created desc;
```

По результатам выполненных тестов запрос, выполненный в ClickHouse, продемонстрировал время выполнения порядка 0.5 секунд. В то же время аналогичный запрос в PostgreSQL выполнился за 15 секунд, что существенно медленнее колоночной СУБД.

Аналогичная ситуация наблюдается и для более простого запроса (листинг 4).

Листинг 4

Пример более простого запроса с агрегацией

```
select avg(length(name)) from customers;
```

План запроса в ClickHouse приведен ниже (листинг 5).

Листинг 5

План запроса с агрегацией в ClickHouse

```
Expression ((Projection + Before ORDER BY))
  Aggregating
    Expression (Before GROUP BY)
      SettingQuotaAndLimits (Set limits and
      quota reading from storage)
        ReadFromMergeTree
```

Можно также получить план запроса в PostgreSQL (листинг 6).

Листинг 6

План запроса с агрегацией в PostgreSQL

```
Finalize Aggregate (cost=1021141.07..1021141.08 rows=1
width=32)
  -> Gather (cost=1021140.86..1021141.07 rows=2 width=32)
     Workers Planned: 2
    -> Partial Aggregate (cost=1020140.86..1020140.87 rows=1
        width=32)
          -> Parallel Seq Scan on customers
(cost=0.00..928367.24
                  rows=18354724
              width=14)
```

Выполнение такого запроса занимает 0.2 и 15 секунд для ClickHouse и PostgreSQL соответственно.

Также, можно провести сравнительный анализ скорости выполнения в зависимости от числа записей. В качестве примера используется запрос, приведенный ниже (листинг 7).

Листинг 7

Пример запроса с агрегацией и проверкой массива в PostgreSQL

```
SELECT COUNT(*)
FROM customers
WHERE 'tag42' = ANY(tags);
```

Аналогичный запрос может выполнить и в ClickHouse (листинг 8).

Листинг 8

Пример запроса с агрегацией и проверкой массива в ClickHouse

```
SELECT COUNT(*)
FROM customers
WHERE has(tags, 'tag42');
```

Результаты выполнения этого запроса в обоих СУБД при числе записей 1, 5, 10, 50 и 100 миллионов приведены на рисунке.

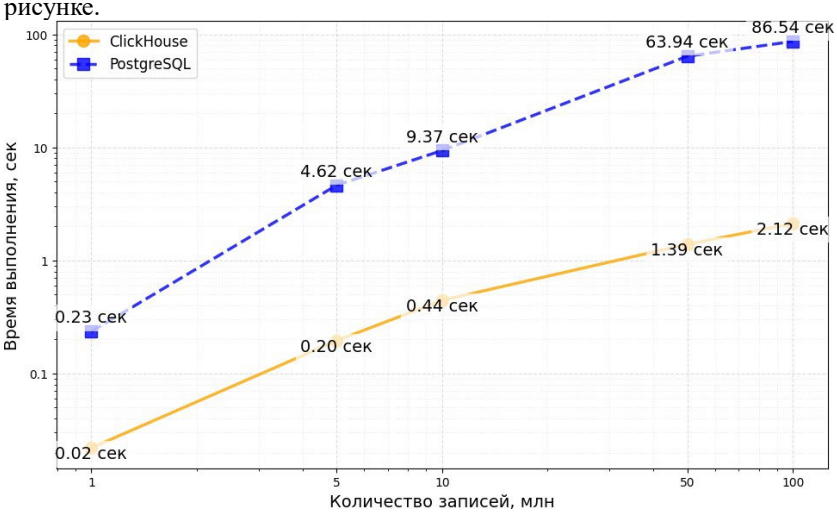


Рисунок. Время выполнения агрегатного запроса выборки с использованием массива в условии выборки на логарифмической шкале при разном количестве записей в PostgreSQL и ClickHouse на логарифмической шкале.

Причина столь значительной разницы во времени выполнения объясняется различиями архитектуры используемых СУБД. ClickHouse,

являясь колоночной базой данных, оптимизирован именно для выполнения аналитических и агрегатных запросов [3]. Данные в такой системе хранятся по столбцам, что позволяет обрабатывать только необходимые для конкретного запроса столбцы, минимизируя количество операций ввода-вывода и существенно повышая скорость обработки больших объемов данных.

PostgreSQL же, как строковая реляционная база данных, хранит информацию построчно, что вынуждает систему выполнять чтение всех строк целиком даже для выполнения агрегатных операций над отдельными столбцами. В результате этого, PostgreSQL расходует большее количество ресурсов на обработку и чтение данных, что непосредственно отражается на времени выполнения сложных аналитических запросов.

К тому же, в PostgreSQL используется многоверсионная архитектура хранения данных, при которой удалённые или обновлённые строки не удаляются физически из таблиц, а помечаются как неактуальные [4]. Это позволяет обеспечивать изолированность транзакций, но приводит к накоплению устаревших данных. Со временем их количество может значительно увеличиться, что негативно сказывается на производительности запросов, особенно при сканировании таблиц. Для устранения этой проблемы необходимо периодически выполнять процедуру вакуумизации, которая требует дополнительных вычислительных ресурсов и может влиять на доступность системы при больших объёмах данных.

Дополнительно была проведена оценка производительности запросов, включающих операции поиска в массивах. Для этого были использованы аналогичные по логике запросы в ClickHouse и PostgreSQL, задачей которых являлось определение количества записей, содержащих определённый элемент массива.

Запрос, реализованный в ClickHouse с использованием функции has(), был выполнен за 1,7 секунды. Аналогичный по функциональности запрос в PostgreSQL с использованием конструкции ANY занял значительно больше времени, составив 1 минуту и 26 секунд.

С целью оптимизации выполнения запроса в PostgreSQL была применена индексация с использованием специализированного GIN-индекса для массивов [5].

Однако даже с учётом использования GIN-индекса, время выполнения такого запроса в PostgreSQL хоть и снизилось до 46 секунд, но всё ещё остается существенно большим по сравнению с аналогичным запросом в ClickHouse.

Функция has() в ClickHouse демонстрирует высокую эффективность при проверке наличия конкретного элемента в массивах, что обусловлено рядом технологических и алгоритмических особенностей реализации этой функции.

Одной из основных причин является оптимизация чтения данных, реализуемая ClickHouse. В случае использования функции has(), ClickHouse способен выполнять частичное обращение непосредственно к вспомогательным подстолбцам, таким как size0, содержащим информацию о размере массивов. Благодаря этому достигается существенное сокращение объёма данных, обрабатываемых при выполнении запроса, что ведёт к ускорению выполнения операции поиска.

Дополнительно в ClickHouse реализована возможность применения специализированных индексов, таких как индекс на основе структуры данных Bloom-фильтр. Эти индексы предназначены для быстрой проверки наличия элементов в массивах и позволяют значительно снизить количество операций ввода-вывода и объём обрабатываемых данных. Использование таких индексов непосредственно влияет на ускорение выполнения запросов с функцией has().

Заключение

В ходе данной работы было проведено комплексное сравнение производительности и особенностей функционирования колоночных систем управления базами данных на примере ClickHouse и традиционных реляционных СУБД на примере PostgreSQL. Были рассмотрены основные архитектурные и алгоритмические отличия колоночных баз данных от классических строковых СУБД, влияющие на производительность выполнения различных видов запросов.

Исследование включало проведение сравнительного анализа выполнения агрегатных запросов, таких как группировка данных, подсчёт количества записей и вычисление средних значений. В результате тестирования установлено значительное превосходство колоночной СУБД ClickHouse в скорости обработки подобных запросов, что обусловлено спецификой её архитектуры, оптимизированной для аналитических задач и последовательного чтения данных.

Также было исследовано выполнение запросов, связанных с обработкой массивов, и выявлены существенные различия в производительности между ClickHouse и PostgreSQL. ClickHouse демонстрирует значительно более высокую скорость выполнения таких запросов благодаря наличию специализированных механизмов оптимизации чтения данных и возможности использования индексов на основе структуры данных Bloom-фильтра. В PostgreSQL была

рассмотрена возможность ускорения выполнения аналогичных запросов при помощи GIN-индексации, однако даже после создания индекса производительность PostgreSQL существенно уступала производительности ClickHouse.

Список литературы

1. Vijay, R. Anand Up and Running with ClickHouse / R. A. Vijay. – Boston: BPB Publications, 2022. – 280 p.

2. PostgreSQL 16 Administration Cookbook / G. Ciolli [et al.]. – Mumbai, India: Packt, 2023. – 636 p.

3. Column-Oriented Database Systems [Электронный ресурс] : учеб. курс / Междунар. конф. по базам данных Very Large Data Bases (VLDB). – Режим доступа: https://www.cs.umd.edu/~abadi/talks/Column_Store_Tutorial_VLDB09.pdf

4. Routine Vacuuming [Электронный ресурс] : офиц. док. / PostgreSQL Global Development Group. – Режим доступа: https://www.postgresql.org/docs/current/routine-vacuuming.html

5. GIN Indexes [Электронный ресурс] : офиц. док. / PostgreSQL Global Development Group. – Режим доступа: https://www.postgresql.org/docs/current/gin.html

Модификации метода искусственных иммунных систем для решения задач негладкой оптимизации

Ю. В. Сафонова

Студент магистр

А. В. Атанов

Доцент

Введение

Общая задача оптимизации заключается в том, чтобы найти глобальный минимум функции $f(\boldsymbol{x})$ на множестве D, то есть такую точку $x^* \in D$, что $f\left(\boldsymbol{x}^*\right) = \min\limits_{\boldsymbol{x} \in D} f\left(\boldsymbol{x}\right)$.

Классические методы теории оптимизации используют частные производные первого и второго порядков. В связи с этим функции, для нахождения максимумов или минимумов которых эти методы применяются, обычно предполагаются гладкими, выпуклыми или обладающими другими «удобными» свойствами. Напомним, что функция называется гладкой, если все её частные производные существуют и являются непрерывными функциями. В таком случае для решения задачи используются классические методы (градиентного спуска, Ньютона и т.д.) [1]. Если же функция не гладкая (такие возникают в машинном обучении, в нейронных сетях и т.д.), то решение задачи существенно сложнее. Для решения таких задач классические методы гладкой оптимизации не подходят. Известно несколько подходов к исследованию задач негладкой оптимизации, которые включают субградиентные методы, методы пучка, методы градиентной выборки и др. (см. [1, 2]). Перспективным направлением в исследованиях оптимизации негладких функций является применение метаэвристических методов, которые не используют ни производные, ни какие-либо их обобщения (см. [3, 4]).

В данной работе один из таких методов — метод искусственных иммунных систем (см. [4, 5]) — исследуется на применимость к задачам негладкой оптимизации.

© Сафонова Ю. В., Атанов А. В., 2025

1. Метод искусственной иммунной системы

Искусственная иммунная система (AIS) — это алгоритм оптимизации, основанный на принципах работы биологической иммунной системы (см. [4,5]).

Основными элементами искусственной иммунной системы являются:

– антиген — целевая функция оптимизации $f:\mathbb{R}^n \to \mathbb{R}$, где n — размерность задачи;

– антитело — потенциальное решение $\boldsymbol{x}_i \in \mathbb{R}^n$, где $i = 1,\ldots,N$ (N – размер популяции).

– аффинность — мера качества решения: $A(\boldsymbol{x}_i) = \dfrac{1}{1+f\left(\boldsymbol{x}_i\right)}$.

Рассмотрим шаги метода искусственной иммунной системы.

Шаг 1. Инициализация популяции.

Генерируется начальная популяция антител:

$$P^{(0)} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\},$$

где $\boldsymbol{x}_i$ равномерно распределены в заданной области поиска.

Шаг 2. Вычисление аффинности.

Для каждого антитела рассчитывается аффинность $A_i = A(\boldsymbol{x}_i)$.

Шаг 3. Клонирование.

Лучшие антитела отбираются и клонируются пропорционально их аффинности. Количество клонов для $\boldsymbol{x}_i$:

$$N_c^{(i)} = \left\lfloor \frac{A_i}{\sum_{j=1}^{N} A_j} \cdot N_t \right\rfloor,$$

где $N_t = \lfloor \beta \cdot N \rfloor$, $\beta \in (0,1]$ – коэффициент клонирования.

Шаг 4. Мутация.

Каждый клон подвергается мутации с вероятностью, обратно пропорциональной аффинности:

$$\boldsymbol{x}_i^{'} = \boldsymbol{x}_i + \alpha \cdot \boldsymbol{r} \cdot (1 - A_i),$$

где $\boldsymbol{r} \sim \mathrm{N}(0,1)$, α – коэффициент мутации.

Шаг 5. Селекция.

Из объединённой популяции (исходные антитела + мутировавшие клоны) выбираются N лучших решений.

Шаг 6. Обновление популяции

Часть худших антител заменяется новыми случайными решениями (для поддержания разнообразия).

Алгоритм выполняется до достижения предельного числа итераций или заданной точности.

2. Модификации метода искусственной иммунной системы

Для более тщательного исследования подхода к оптимизации на основе искусственной иммунной системы в работе предлагаются три модификации указанного метода. Рассмотрим подробнее каждую из них.

1. Метод искусственной иммунной системы с адаптивной мутацией

Идея модификации с адаптивной мутацией заключается в динамическом регулировании коэффициента мутации в зависимости от прогресса алгоритма. Если лучший результат не улучшается в течение нескольких итераций, коэффициент мутации увеличивается, чтобы выйти из локального оптимума. В противном случае коэффициент мутации постепенно ослабевает для более точного исследования области.

Выделим основные параметры, характеризующие алгоритм AIS с адаптивной мутацией:

$\alpha_0 \in (0,1)$ – начальный коэффициент мутации;

$T \in N$ – порог стагнации (число итераций без улучшения);

$\eta_+ \in (0,1)$ – коэффициент увеличения мутации;

$\eta_- \in (0,1)$ – коэффициент уменьшения мутации.

Правило изменения коэффициента мутации на каждой итерации t:

$$\alpha_{t+1} = \begin{cases} \min(\alpha_{\max}, \alpha_t + \eta_+), \text{стагнация} \geq T, \\ \max(\alpha_{\min}, \alpha_t \cdot \eta_-), \text{стагнация} < T. \end{cases}$$

Здесь $\alpha_{\max}$, $\alpha_{\min}$ – задаваемые допустимые границы изменения коэффициенты мутации. Стагнация определяется как число последовательных итераций без улучшения лучшего решения.

2. Метод искусственной иммунной системы с памятью

Идея модификации с памятью заключается в сохранении лучших решений в специальной памяти, которая используется для генерации новых кандидатов, что позволяет избегать потери перспективных решений.

Выделим основной параметр, характеризующий алгоритм AIS с память: $S \in N$ – максимальный размер памяти.

В указанной модификации предусмотрен функционал обновления памяти. На каждой итерации t выполняются следующие шаги:

1) Нахождение лучшего решения текущей популяции:

$$\boldsymbol{x}_{best}^{(t)} = \arg\min_{\boldsymbol{x}\in P_t} f(\boldsymbol{x}),$$

где $P_t = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$ – популяция на итерации t.

2) Добавление лучшего решения $\boldsymbol{x}_{best}^{(t)}$ в память с использованием следующей формулы:

$$M_t = M_{t-1} \bigcup \{\boldsymbol{x}_{best}^{(t)}\}.$$

3) В случае, если $|M_t| > S$, удаление самого старого решения из памяти.

После генерации мутированных клонов C_t формируется новая популяция на основе N лучших решений из множества $P_t \bigcup C_t \bigcup M_t$.

3. Метод искусственной иммунной системы с локальным поиском

Идея модификации с локальным поиском заключается в периодическом применении локального поиска к лучшему найденному решению, что позволит точнее исследовать перспективные области.

Выделим два основных параметра, характеризующие алгоритм AIS с локальным поиском: $p \in [0,1]$ – параметр, указывающий на релевантность применения локального поиска; $K \in N$ – число шагов локального поиска.

Математическая интерпретация локального поиска для мутированного клона $\boldsymbol{x}_i'$ представлена в виде:

$$\boldsymbol{x}_i^* = \arg\min_{\boldsymbol{x}\in B(\boldsymbol{x}_i', \varepsilon)} f(\boldsymbol{x}),$$

где $B(\boldsymbol{x}_i', \varepsilon)$ – ε -окрестность решения $\boldsymbol{x}_i'$.

Для локального поиска в работе была использована одна из модификаций алгоритма Бройдена–Флетчера–Гольдфарба–Шанно (L-BFGS-B) (см. [1, 6]).

В алгоритме также представлена гибридная мутация с вероятностью p, формула которой имеет вид:

$$\boldsymbol{x}_i^{final} = \begin{cases} \boldsymbol{x}_i'', f(\boldsymbol{x}_i'') < f(\boldsymbol{x}_i'), \\ \boldsymbol{x}_i', f(\boldsymbol{x}_i'') \geq f(\boldsymbol{x}_i'). \end{cases}$$

3. Примеры решения задач негладкой оптимизации с помощью метода искусственной иммунной системы и его модификаций

Реализации программного кода осуществлялась на языке программирования Python [7].

Для проверки работы разработанных алгоритмов были выбраны негладкие функции [2], использующиеся как тестовые при оценке работы оптимизационных алгоритмов.

Для каждой из указанных функций с использованием классического метода AIS и его модификаций вычислялось значение минимума и сравнивались графики сходимости.

Пример 1. Функция максимума квадратов (MAXQ):

$$f(x_1,...,x_{20}) = \max(x_i^2),\ i = 1,\ldots,20.$$

Наименьшее значение этой функции равно $f^* = 0$. График функции для случая двух переменных представлен на рис. 1.

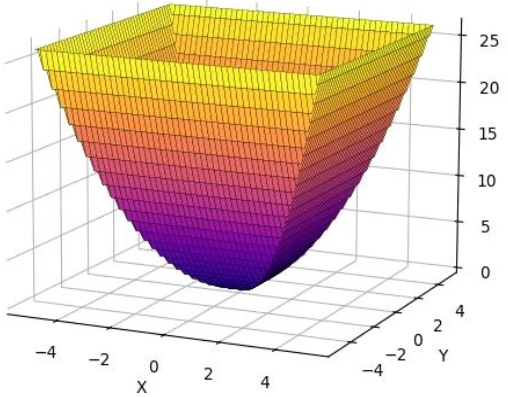


Рис. 1. График функции MAXQ для двух переменных

В табл. 1 представлены результаты работы алгоритмов для 20-мерной функции максимума квадратов. Для получения более точного результата проводилась серия запусков программы. В табл. 1 представлены лучшие (минимальные) значения целевой функции, а также средние значения, которые позволяют оценить, насколько текущие генерируемые решения близки к лучшему. График сходимости алгоритмов показан на рис. 2.

Таблица 1

Результаты работы программы для функции MAXQ

Метод	**Лучшее значение f**	**Среднее значение f**
Классический AIS	8.836833	10.444313
AIS с адаптивной мутацией	5.114172	7.335692
AIS с памятью	0.096237	0.098114
AIS с локальным поиском	0.000126	0.000688

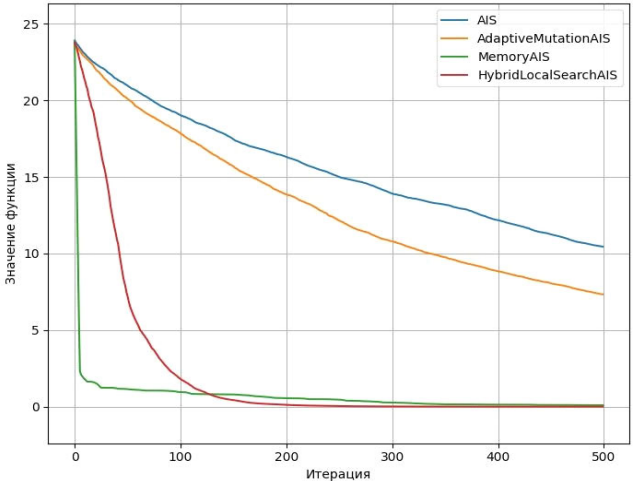


Рис. 2. График сходимости для 20-мерной функции MAXQ

Как видно по полученным результатам, AIS с памятью и AIS с локальным поиском в данном случае сработали лучше всего. При этом классический AIS и AIS с адаптивной мутацией за 500 итераций не достигли оптимального значения.

Пример 2. Функция Гилла

$$F(\boldsymbol{x}) = \max\{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x})\},\ 1 \le i \le 10,$$

$$f_1(\boldsymbol{x}) = \sum_{i=1}^{10}(x_i - 1)^2 + 10^{-3}\sum_{i=1}^{10}\left(x_i^{\,2} - \frac{1}{4}\right)^2,$$

$$f_2(\boldsymbol{x}) = \sum_{i=2}^{30}\left[\sum_{j=2}^{10} x_j (j-1)\left(\frac{i-1}{29}\right)^{j-2} - \left(\sum_{j=1}^{10} x_j \left(\frac{i-1}{29}\right)^{j-1}\right)^2 - 1\right]^2 +$$

$$+x_1^{\,2} + (x_2 - x_1^{\,2} - 1)^2,$$

$$f_3(\boldsymbol{x}) = \sum_{i=2}^{10} \left[100(x_i - x^2{}_{i-1})^2 + (1 - x_i)^2 \right],$$

Наименьшее значение функции Гилла: $f^* \approx 9.7857721$.

В табл. 2 представлены результаты работы алгоритмов для 10-мерной функции Гилла. График сходимости изучаемых алгоритмов показан на рис. 3.

Таблица 2

Результаты работы программы для функции MAXQ

Метод	**Лучшее значение f**	**Среднее значение f**
Классический AIS	8268.050477	21543.920213
AIS с адаптивной мутацией	2087.931944	6274.620266
AIS с памятью	26.635347	26.640191
AIS с локальным поиском	9.866341	10.062273

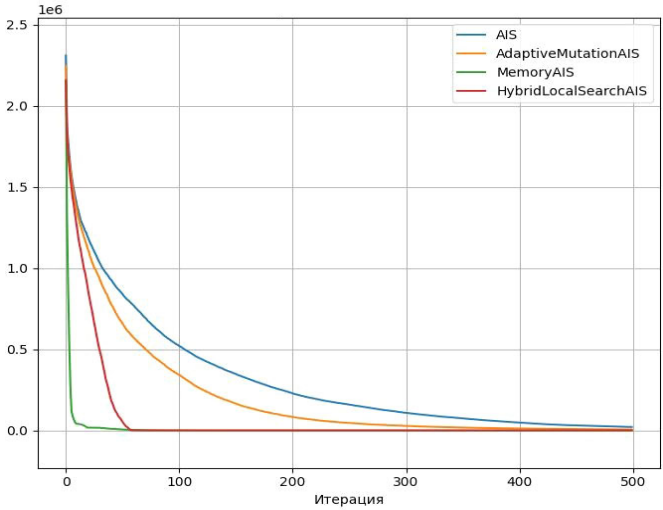


Рис. 3. График сходимости для 10-мерной функции Гилла

Как видно по полученным результатам, для рассматриваемого примера AIS с локальным поиском показал наилучшие результаты.

Классический AIS и AIS с адаптивной мутацией за 500 итераций не достигли оптимального значения целевой функции.

Заключение

В ходе работы разработаны и исследованы модификации метода искусственных иммунных систем для решения задач негладкой оптимизации. Модификации алгоритма были применены к решению задач негладкой оптимизации на основе самостоятельно разработанной программной реализации представленных методов, а также проведено тестирование метода искусственных иммунных систем и его модификаций на различных задачах негладкой оптимизации.

Как показало тестирование разработанных алгоритмов, наилучшие результаты для задач негладкой оптимизации показывает метод искусственной иммунной системы с локальным поиском.

Список литературы

1. Поляк, Б. Т. Введение в оптимизацию / Б. Т. Поляк. – Изд. 2-е, испр. и доп. – М. : URSS, 2014. - 386 с.
2. Bagirov, A. Introduction to Nonsmooth Optimization: theory, practice and software / A. Bagirov, N. Karmitsa, M. M. Makela ; Springer International Publishing, 2014. – 372 p.
3. Саймон, Д. Алгоритмы эволюционной оптимизации : руководство / Д. Саймон ; пер. с англ. А. В. Логунова. – М. : ДМК Пресс, 2020. – 1002 с.
4. Пантелеев, А. В. Метаэвристические стратегии и алгоритмы глобальной оптимизации / А. В. Пантелеев, Д. В. Скавинская. – Факториал, 2023. – 564 с.
5. Dasgupta, D. Articial Immune Systems and Their Applications / D. Dasgupta. – Springer Berlin, 1999. – 306 p.
6. Nocedal, J. Numerical Optimization / J. Nocedal, S. J. Wright. – Springer, 2006. – 664 p.
7. Dawson, M. Python Programming for the Absolute Beginner / M. Dawson. – O'Reilly, 2014. – 416 p.

Искусственные иммунные системы в информационной безопасности

И. В. Синяков

Студент магистр

В. С. Тарасов

Старший преподаватель

Введение

Иммунной называется система органов, отвечающая за защиту организма от заболеваний путем обнаружения и уничтожения нано-генов и опухолевых клеток. Такая система способна распознавать множество разнообразных возбудителей, которые способны адаптироваться и эволюционировать.

Искусственная иммунная система (ИИС, Artifical Immune Systems, AIS), основанная на заимствованных из биологии принципах и алгоритмах функционирования иммунной системы человека, является подходом, реализуемым методом машинного обучения. Изучение иммунной системы человека продолжается и в данное время. В иммунологии существуют различные модели ее функционирования. В ходе машинного обучения ИИС реализуются на основе разных моделей и подходов. В качестве базовых концепций ИИС можно выделить негативный и положительный отбор, клональную селекцию, гиперклетки, иммунные сети, системы иммунного ответа. Названные подходы имеют между собой много общего. Более сложные концепции основываются на более простых.

1. Основные свойства иммунной системы человека

Несмотря на существенные различия в моделях, ИИС имеют много общих определяющих их применимость свойств, соответствующих свойствам иммунной системы человека, таким как распределенность, самоорганизуемость, ограниченность требований к вычислительным ресурсам.

Любая модель ИИС подразумевает распределенность как внутри модели иммунитета, так и при применении такой системы. Любая ИИС включает множество детекторов, каждый из которых обучается и применяется независимо от других. Это позволяет реализовать

© Синяков И. В., Тарасов В. С., 2025

детекторы как на разных ядрах центрального процессора одной вычислительной машины, так и, при необходимости, на различных вычислительных устройствах. Функциональные блоки ИИС могут помимо этого располагаться на различных устройствах.

Например, можно хранить базу информативных характеристик и выполнять обучение детекторов на сервере, а выполнять детектирование на множестве клиентских вычислительных машин.

Самоорганизуемость. ИИС подразумевает возможность автоматизированного или автоматического функционирования за счет автоматизации жизненного цикла популяции детекторов, возможности автоматического добавления в базу новых аномалий, возможности автоматизации реагирования на выявленные инциденты. ИИС способны непрерывно обновлять и совершенствовать базу детекторов.

Ограниченность требований к вычислительным ресурсам обусловливается возможностью разделения функциональных частей ИИС на отдельные модули, работа которых может осуществляться на отдельных вычислительных системах. Можно раздельно проводить предварительное исследование объектов (например, сетевых пакетов или потоков) на множестве сетевых устройств с пониженными требованиями к производительности аппаратного обеспечения, детальное исследование выявленных подозрительных объектов на отдельных вычислительных устройствах и, наконец, актуализацию действующих баз на сервере. Таким образом, на основе ИИС могут быть обеспечены высокая масштабируемость комплекса сетевой защиты в зависимости от масштабов сети и других требований, минимизация требований к производительности отдельных элементов системы, устранение необходимости повторения части одинаковых задач на схожих устройствах в крупной системе с множеством устройств. Необходимо отметить сравнительно простую природу отдельных детекторов [1].

2. Задачи, решаемые ИИС.

Как и любая система машинного обучения, иммунная система призвана решать задачи, которые не могут быть решены экспертами. При этом ИИС способны успешно решать не только базовые задачи машинного обучения, но и преодолевать ряд сложностей таких, как высокая вариативность и непредсказуемость условий и решений.

Среди задач, которые способны решать ИИС, различные задачи классификации и машинного обучения, в число которых входят: Обнаружение. Выявление вредоносного воздействия или потенциально опасного состояния часто является первым шагом работы системы защиты информации. В биологической иммунной системе обнаружение

также является первым шагом для выполнения защитной реакции. Тем не менее как в биологической, так и в искусственной иммунной системе для успешного обнаружения необходимо создание и постоянная актуализация популяции детекторов.

Обнаружение может быть не единственной задачей, решаемой ИИС, но часто является первым этапом защиты системы, вследствие сего его точность может иметь критичное значение для эффективности всей системы.

Классификация. В системах защиты информации классификация часто оказывается вторым этапом работы, на основе которого может быть выбран способ реагирования, адекватный угрозе.

Это полностью аналогично биологической иммунной системе. Однако классификация, может быть, и отдельной задачей для ИИС и использоваться для автоматизации или повышения эффективности обучения других методов машинного обучения.

Анализ данных. ИИС могут применяться для выявления паттернов в массивах данных как самостоятельно, так и в качестве инструментария для применения других методов машинного обучения.

В частности, ИИС способны работать с значительно изменяющимися паттернами и в изменяющихся условиях.

Обработка сигналов и данных. Как и другие методы машинного обучения, ИИС могут использоваться для обнаружения и обработки потоковых данных. При таком использовании их достоинством является высокая адаптивность к изменяющимся условиям.

Моделирование. ИИС могут быть использованы для моделирования поведения распределенных систем и популяций.

Поиск и оптимизация. ИИС показывают многообещающие результаты в осложненных случаях поиска или оптимизации благодаря способности обрабатывать быстроизменяющиеся данные с неочевидными взаимосвязями.

Базовой целью функционирования ИИС является обнаружение паттернов в потоке данных. В ходе реализации ИИС необходимо решение двух взаимосвязанных проблем: разработка детекторов, соответствующих характеру анализируемых данных, и организация их обучения и актуализации. Решение второй проблемы обеспечивается заимствованными из иммунологии механизмами, лежащими в Основе ИИС. Функционирование основных механизмов ИИС направлено на максимизацию аффинности детекторов к обнаруживаемым [1-4].

3. Архитектура искусственной иммунной системы

Архитектура ИИС может зависеть от характера рассматриваемых данных, выполняемых ИИС задач, требуемых результатов работы.

Система включает: подсистему обнаружения искомых паттернов. Целью такой подсистемы является обнаружение во входных данных требуемых объектов при помощи созданной популяции антител. В составе ИИС может быть несколько таких подсистем, реализованных на нескольких вычислительных устройствах, что обуславливает гибкость и масштабируемость такой системы; подсистему хранения характеристик и актуализации данных, предназначенную для хранения базы известных и обнаруженных ранее искомых объектов, и актуализации с ее помощью популяции антител.

Выявив среди входных данных объекты, сходные с искомыми, система оповещает администратора об инциденте. При обнаружении новых, неизвестных ранее экземпляров искомых объектов, система обнаружения также направляет их в систему хранения для того, чтобы при следующей итерации обновления популяции детекторов такие экземпляры тоже были учтены. Это необходимо потому, что группа новых экземпляров может иметь различия, вследствие которых старые детекторы обнаруживают только часть из них. В процессе обучения система должна постепенно научиться выявлять и новые объекты. Независимо от того, были ли изменения в базе сохраненных объектов, модуль актуализации популяции антител продолжает итеративно обновлять популяцию. Такой режим работы необходим для избегания возможного переобучения системы и поиска новых, более оптимальных и универсальных детекторов.

Область сетевой безопасности характерна высокой изменчивостью сетевого окружения, в котором должна выполнять работу система безопасности, значительной изобретательностью злоумышленников и непредсказуемостью возможных инцидентов и поведения пользователей.

К основным целям использования ИИС в области сетевой безопасности можно отнести обнаружение и устранение вредоносного трафика, включая трафик компьютерных атак, трафик с вредоносным программным кодом, трафик между пользователями и запрещенными ресурсами, и т.д.; обнаружение и предотвращение атак и неправомерных действий как внутренних пользователей сети, так и внешних: выявление и оперативное реагирование на неполадки и инциденты в работе сети;

Области применения ИИС в контексте сетевой безопасности обусловливаются как эффективностью при обработке больших потоков данных, так и определенной сложностью разработки и реализации. Среди основных приложений ИИС в области информационной безопасности можно выделить следующие: обеспечение безопасности внутренних сетей крупных компаний: защита данных, передаваемых

через узлы сети провайдера; выявление предпосылок к выходу из строя или повреждению оборудования связи; обнаружение ошибок в работе сети и своевременное информирование персонала об их возникновении; обнаружение новых, ранее не встречавшихся атак и иных вредоносных действий злоумышленников компаниями, специализирующимися на компьютерной безопасности; обеспечение контроля за трафиком; отслеживание запрещенного трафика с целью блокировок или проведения следственных мероприятий; мониторинг и контроль за использованием ресурсов сети пользователями, определенными приложениями.

Структура ИИС, предназначенной для обнаружения сетевых аномалий, сходна с представленной на рисунке. В качестве входных и выходных данных в ней выступает сетевой трафик.

Рисунок. Пример архитектуры ИИС

В случае систем на основе модели «свой-чужой» ИИС целесообразно рассматривать характеристики сетевых потоков или пакетов.

При использовании модели на основе теории опасности список параметров может быть расширен как параметрами текущего состояния других потоков или пакетов, так и информация о состоянии защищаемой системе, а также другая информация, которая может

сообщить о проводимой атаке. Обнаружение с помощью ИИС только аномальных объектов, как правило, является недостаточным для обеспечения требуемого уровня защищенности. В дополнение к обнаружению аномалий системе полезно определять тип атаки и по возможности предоставлять администратору более детализированную и очищенную от лишних данных информацию. Такая система содержит два типа антител, популяции которых выполняется соответственно обнаружение сетевых аномалий и классификацию выявленных аномалий по типам содержащихся в них атак.

В системе имеется три функциональные подсистемы, которые можно реализовать на отдельных вычислительных устройствах. Причем для конкретного применения в составе одной системы может иметься несколько подсистем обнаружения или классификации для обеспечения масштабируемости системы. Еще одним отличием такой системы от рассмотренной ранее в области защиты компьютерных сетей является то, что найденные аномальные объекты добавляются в базу данных после этапа классификации. Это позволяет проводить дальнейшее обучение, в том числе и детекторов, на отдельных подмножествах искомых объектов, повышая тем самым гибкость системы при обнаружении атак разных типов, значительно различающихся между собой [1-6].

4. Обзор существующих реализаций ИИС в области сетевой безопасности

На текущий момент на основе ИИС разработано значительное число систем обеспечения сетевой и компьютерной безопасности.

Рассмотрим некоторые из них:

1. Проект CFID (Computational Immunology for Fraud Detection), разработанный и реализованный в Великобритании, предназначен для защиты системы почтовой службы на базе ИИС (Artifical Immune System).

2. Сетевая система обнаружения атак Lisys, созданная европейскими специалистами, выполняет контроль сообщений TCP SYN в сетевом трафике, сообщая оператору о наличии подозрительных соединений. В ее работе используются 49-байтовые детекторы, обрабатывающие информацию об IP-адресах отправителя и получателя, а также портах отправления и назначения. При первоначальном запуске детекторы генерируются случайным образом и затем по мере их срабатывания на нормальном трафике удаляются. Помимо этого, каждый детектор имеет ограниченное время жизни, по завершении которого он уничтожается, что приводит со временем к полному пересозданию всех детекторов. Для снижения числа ложных

срабатываний разработчики добавили пороговое значение активации, при превышении которого происходит срабатывание системы.

3. Расширение к ядру системы Linux pH (Process Homeostasus), разработанное в США, позволяет обнаруживать и при необходимости замедлять поведение подозрительных программ. С этой целью pH создает профили системных вызовов, сделанных в системе различными программами. При первоначальном запуске на создание таких профилей уходит определенное время, по истечении которого расширение может выполнять свою основную работу. При обнаружении подозрительного процесса pH вначале добавляет ему задержку времени в очереди на ожидание выполнения вызовов, а затем, при необходимости, уничтожает вредоносный процесс. Поскольку контроль за всеми системными вызовами требует чрезвычайно большого числа вычислительных ресурсов, pH контролирует только вызовы с полным доступом к ресурсам компьютера.

4. Система STIDE (Sequence Time-Delay Embedding) аналогична pH, предназначена для обнаружения вторжений по необычным системным вызовам. На стадии обучения STIDE создает базу характеристик всех уникальных непрерывных системных вызовов, а затем делит их на фрагменты определенной фиксированной длины.

В процессе работы STIDE сравнивает события с имеющимися записями в базе и при обнаружении подозрительного процесса сообщает о нем, отмечая также сколько новых системных вызовов отличается от нормы.

5. Еще одним примером реализованной ИИС может служить экспериментальная система иммунной защиты информационных ресурсов национального оператора связи, разрабатываемая в проекте системы раннего предупреждения о компьютерном нападении. В работе системы используются готовые аппаратные компоненты [1-7].

5. Эффективность использования ИИС в сетевой безопасности

Искусственные иммунные системы (ИИС) демонстрируют уникальный баланс между эффективностью обнаружения аномалий и умеренными вычислительными требованиями, что делает их конкурентоспособными в задачах сетевой безопасности. По сравнению с традиционными алгоритмами, такими как Random Forest и нейросетями, ИИС показывают несколько меньшую точность (85–92% против 90–97% у LSTM), но превосходят их в сценариях с ограниченными ресурсами. Например, на датасете CICIDS 2017 ИИС достигла 89% точности, используя всего 70 МБ оперативной памяти и 1 Вт энергопотребления на обработку 1000 запросов, тогда как LSTM-

нейросеть требовала 1.5 ГБ памяти и 8 Вт, но обеспечивала более высокую точность (96%).

Ключевым преимуществом ИИС является способность обнаруживать неизвестные угрозы без предварительной разметки данных, что особенно ценно в условиях динамично меняющихся атак. Однако обучение ИИС занимает больше времени (2–5 секунд на 10К образцов) по сравнению с Random Forest (1–3 секунды), а настройка параметров (например, радиуса совпадения r) требует экспертизы. Нейросети, в свою очередь, превосходят ИИС по полноте (93–98% против 88–94%) и F1-мере (0.94–0.97 против 0.89–0.93), но их высокая ресурсоемкость ограничивает применение в IoT-устройствах или системах реального времени.

Гибридные модели, объединяющие ИИС и нейросети, демонстрируют оптимальные результаты: ИИС фильтрует подозрительные события с низкой нагрузкой на железо, а нейросеть классифицирует их с высокой точностью (до 98% при F1-мере 0.96–0.99). Это делает их перспективным решением для масштабных инфраструктур, где требуется как чувствительность к новым атакам, так и стабильная производительность [3-7]. Результат сравнения моделей представлен в виде таблицы.

Таблица

Сравнение ИИС с другими алгоритмами по ключевым метрикам

Название метрики	ИИС (NSA)	Random Forest	LSTM-нейросеть	Гибридный ИИС + LSTM
Accuracy	85–92%	90–95%	92–97%	94–98%
Recall	88–94%	91–96%	93–98%	95–99%
F1	0.89–0.93	0.92-0.95	0.94–0.97	0.96–0.99
Время обучения (на 10К образцов)	2–5 секунд	1–3 секунд	30–60 секунд	40–70 секунд
Использование RAM	50–100 МБ	30–80 МБ	500 МБ – 2 ГБ	700 МБ – 3 ГБ
Энергопотребление	0.5–1 Вт	0.3–0.8 Вт	5–10 Вт	6–12 Вт

Расчеты метрик эффективности алгоритмов производились на устройстве Apple MacBook Air с процессором M2 Silicon (8 ядер) и 8 ГБ оперативной памяти.

Заключение

В ходе исследования была проанализирована концепция искусственных иммунных систем (ИИС) и их применение в информационной безопасности. ИИС, основанные на принципах биологического иммунитета, демонстрируют высокую адаптивность, способность к обучению и эффективное обнаружение угроз.

Основные преимущества ИИС включают возможность выявления неизвестных атак, работу в режиме реального времени и автономное принятие решений. В отличие от традиционных систем защиты, основанных на сигнатурном анализе, ИИС способны распознавать ранее неизвестные угрозы за счет механизмов аномального поведения и самообучения.

Однако технологии ИИС все еще сталкиваются с рядом вызовов, таких как высокая вычислительная сложность, необходимость точной настройки параметров и риск ложных срабатываний. Для дальнейшего развития ИИС в сфере кибербезопасности требуется совершенствование алгоритмов, применение гибридных подходов и интеграция с другими методами защиты.

Таким образом, искусственные иммунные системы представляют собой перспективное направление в области кибербезопасности, способное повысить уровень защиты компьютерных систем от современных и будущих угроз. Дальнейшие исследования в этой области позволят улучшить их эффективность и внедрение в практическое применение.

Список литературы

1. Шелухин, О.И. Технологии машинного обучения в сетевой безопасности / Шелухин О.И., Ерохин С.Д., Полковников М.В. // Технологии – М.: Горячая линия – Телеком, 2024. – 360 с.

2. Использование искусственных иммунных систем для решения задачи символьной регрессии [Электронный ресурс]. – https://habr.com/ru/articles/181342/

3. Хайкин, С. Нейронные сети: полный курс / С. Хайкин. – 2-е изд., испр.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2006. – 1104 стр.

4. Дасгупта, Д. Искусственные иммунные системы и их применение / Д. Дасгупты; Пер. с англ. под ред А. А. Романюхи. – Москва: ФИЗМАТЛИТ, 2006. – 344 с.

5. Де Кастро, Л.Н. Artificial Immune Systems: A New Computational Intelligence Approach / Л. Н. де Кастро, Дж. Тиммис. – Springer, 2002. – 357 с.
6. Скобцов, Ю. А. Введение в искусственные иммунные системы: учебное пособие / Ю. А. Скобцов. – Санкт-Петербург: ГУАП, 2022. – 212 с.
7. Искусственные иммунные системы в информационной безопасности [Электронный ресурс]. – https://habr.com/ru/companies/otus/articles/463891/

Генерация изображений цветов по текстовому описанию с помощью условных GAN и постобработка изображений с помощью Real-ESRGAN

Д. В. Сливкин

Студент магистр

А. Ю. Иванков

Доцент

Введение

Генерация изображений по текстовому описанию — одна из самых актуальных и сложных задач в области искусственного интеллекта и компьютерного зрения. Она требует не только глубокого понимания семантики текста, но и способности синтезировать визуально правдоподобные изображения, соответствующие этому описанию. Такие системы находят применение в дизайне, искусстве, медицине, автоматизации творчества и даже в научных исследованиях, где требуется визуализация объектов по их описанию.

Прорыв в этой области произошёл с появлением генеративно-состязательных сетей (GAN) [1-3], которые позволили синтезировать реалистичные изображения с нуля. В дальнейшем были предложены модели условных генеративно-состязательных сетей (cGAN) [4], где генерация осуществляется с учётом дополнительной информации — например, текстового описания. Одними из первых успешных работ стали StackGAN [5], AttnGAN [6], которые показали возможность генерации изображений высокого качества по сложным текстовым запросам.

Несмотря на успехи, задача генерации изображений по тексту остаётся крайне сложной. Основные вызовы: - Семантическое соответствие: изображение должно точно отражать смысл текста. - Визуальное качество: изображение должно быть фотореалистичным, без артефактов. - Разнообразие: одна и та же фраза может соответствовать множеству разных изображений. - Контроль над деталями: возможность управлять отдельными аспектами изображения через текст. - Вычислительные ресурсы: обучение и инференс требуют значительных вычислительных мощностей.

© Сливкин Д. В., Иванков А. Ю., 2025

Примеры успешных моделей:

1. StackGAN: первая модель, показавшая возможность поэтапной генерации изображений высокого разрешения по тексту.

2. AttnGAN: внедрение механизма внимания (attention) для более точного соответствия между словами и частями изображения.

3. DALL-E, Imagen, Stable Diffusion: масштабные модели, способные создавать изображения любого содержания и стиля.

Генерация изображений по тексту — это не только вызов для исследователей, но и мощный инструмент для индустрии. Такие технологии позволяют автоматизировать творческие процессы, ускорять разработку новых продуктов, делать образование и науку более наглядными и доступными. Кроме того, они открывают новые горизонты для взаимодействия человека и машины, где текст становится универсальным интерфейсом для создания визуального контента.

В данной работе реализован проект по генерации изображений цветов на основе текстовых описаний с помощью условных GAN (cGAN). Особенностью проекта является полный автоматизированный пайплайн: от сбора и разметки данных до постобработки изображений с помощью современных методов сверхразрешения (Real-ESRGAN).

2. Подготовка и описание данных

Для обучения модели был использован датасет изображений цветов, включающий тысячи фотографий различных видов растений. Изображения были получены из открытых источников, таких как специализированные датасеты (например, Oxford 102 Flowers [7]), а также из общедоступных коллекций. Важно, чтобы изображения были разнообразными по ракурсу, освещению, фону и видам цветов, что способствует лучшей генерализации модели [5-6].

Предобработка и аугментация изображений

Перед использованием изображения проходят несколько этапов предварительной обработки, перечисленных далее.

1. Изменение размера: все изображения приводятся к единому размеру (128×128 пикселей) для унификации входа в модель.

2. Обрезка вокруг центрального фрагмента: применяется для сохранения пропорций и фокусировки на объекте (цветке).

3. Нормализация: значения пикселей приводятся к диапазону $[-1,1]$, что ускоряет и стабилизирует обучение модели.

4. Аугментация: при необходимости используются случайные повороты, отражения, изменение яркости и контраста для увеличения разнообразия обучающей выборки.

Генерация текстовых описаний с помощью BLIP

Для каждого изображения автоматически генерируется текстовое описание с помощью модели BLIP (Bootstrapped Language Image Pretraining) [8]. BLIP — это современная мультимодальная модель, способная создавать осмысленные и детализированные подписи к изображениям. Она обучена на огромных коллекциях пар “изображение — текст” и может формировать описания, отражающие цвет, форму, детали и контекст объекта.

Пример работы BLIP [8]:

– вход: фотография красного цветка с жёлтой серединкой;

– выход: «A red flower with a yellow center blooming in a green field».

Эмбеддинг описания: что это и зачем нужен

Текстовое описание — это строка, которую необходимо преобразовать в числовой вектор признаков для подачи в нейросеть. Такой вектор называется эмбеддингом. В проекте используется модель SentenceTransformer (например, all-mpnet-base-v2), которая переводит текст в компактное векторное представление фиксированной размерности (например, 768). Данная модель обладает рядом преимуществ.

– Модель обучена на задачах семантического сопоставления, что позволяет ей хорошо улавливать смысл текста.

– Эмбеддинги устойчивы к перефразировкам и отражают не только отдельные слова, но и общий контекст описания.

Пример эмбеддинга:

– описание: «A white daisy with a green stem»;

– эмбеддинг (первые 8 значений из 768): [0.12, -0.34, 0.56, 0.08, -0.22, 0.91, -0.15, 0.44, …].

Формат хранения данных и структура датасета

Все данные хранятся в формате HDF5, что позволяет эффективно работать с большими коллекциями изображений и связанных с ними метаданных. Для каждого изображения сохраняются (табл. 2): само изображение (в виде массива байтов), сгенерированное текстовое описание, эмбеддинг описания (вектор), класс объекта (например, “flower”).

3. Архитектура генератора

Генератор — это сложная нейросетевая архитектура, задача которой — синтезировать изображение, максимально соответствующее заданному текстовому описанию и обладающее высоким визуальным

качеством. В данной работе используется глубокий генератор с residual-блоками и self-attention [6].

Таблица 1

Структура данных

ID	Изображение	Описание	Эмбеддинг (первые 5 значений)	Класс
000001	[PNG bytes]	A red flower with a yellow center	0.11, -0.23, 0.45, 0.09, 0.67	flower
000002	[PNG bytes]	A white daisy with a green stem	0.12, -0.34, 0.56, 0.08, 0.91	flower
…	…	…	…	…

Математическая постановка

Пусть z^{d_z} — латентный вектор шума, e^{d_e} — эмбеддинг текстового описания. Генератор реализует отображение:

$$G : z, e \to x_{fake}^{CHW}.$$

Dense-проекция и объединение признаков

– Латентный вектор z и эмбеддинг e проходят через отдельные линейные (dense) слои:

$$z' = W_z z + b_z,$$

$$e' = W_e e + b_e.$$

– Полученные векторы конкатенируются:

$$h = [z'; e'].$$

– Итоговый вектор (h) подается на dense-слой, формирующий тензор начального размера (например, 4×4×N каналов) [2]:

$$T_0 = (W_h h + b_h).$$

Upsample-блоки

– Каждый upsample-блок увеличивает разрешение изображения в 2 раза:

$$T_{i+1} = \left(T_i\right).$$

– Обычно используется транспонированная свёртка (ConvTranspose2d), BatchNorm и LeakyReLU:

$$T_{i+1} = \left(\left(\left(T_i\right)\right)\right).$$

Residual-блоки

– В каждом upsample-блоке встроен residual-блок:

$$R(T) = \big(T + F(T)\big).$$

– Это позволяет сохранять информацию между слоями, облегчает обучение глубоких сетей и способствует генерации сложных текстур.

Self-Attention

– Self-Attention внедряется на нескольких уровнях:

$$A(T) = T + F(T).$$

– Attention позволяет учитывать взаимосвязи между удалёнными частями изображения, что важно для согласованности структуры (например, расположения лепестков).

Финальный слой

– После всех upsample-блоков применяется свёрточный слой:

$$x_{fake} = T_{final}.$$

– Активация tanh нормализует значения пикселей в диапазон $[-1,1]$.

Схема архитектуры генератора

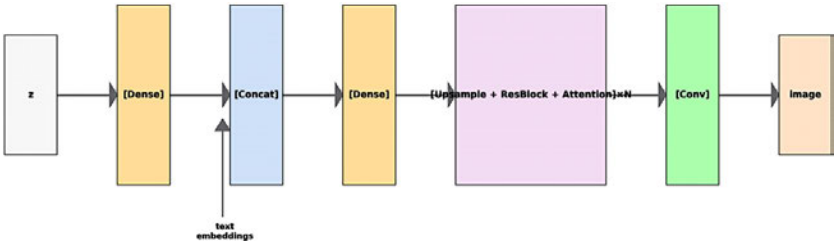


Рис. 1. Схема архитектуры генератора

Альтернативные архитектурные решения

– Можно использовать skip-соединения (U-Net), что улучшает детализацию.

– Использование других attention-механизмов (multi-head, cross-attention).

– Вариации нормализации: InstanceNorm, LayerNorm вместо BatchNorm.

Роль каждого компонента

– Dense-проекция: обеспечивает начальное смешивание признаков шума и текста.

– Upsample-блоки: поэтапно увеличивают разрешение, формируя структуру изображения.

– Residual-блоки: способствуют стабильности и глубине сети.

– Self-Attention: улучшает глобальную согласованность и соответствие тексту.

– Финальный слой: формирует итоговое изображение.

4. Архитектура дискриминатора

Дискриминатор — это сверточная нейросеть, задача которой — отличать реальные изображения от сгенерированных, а также (опционально) учитывать текстовое описание.

Входные данные

– Изображение (x^{CHW})

– (Опционально) эмбеддинг текста (e)

Сверточные блоки

– Последовательность Conv2d-слоёв с увеличением числа каналов и уменьшением пространственного разрешения: $F_{i+1} = F_i$

– InstanceNorm стабилизирует обучение, SpectralNorm предотвращает взрыв градиентов.

Финальный слой

– Последний свёрточный слой сжимает карту признаков до одного значения (скаляр): $s = F_{final}$

– Это значение отражает вероятность того, что изображение реально.

Извлечение признаков

– Для дополнительных лоссов (feature matching) дискриминатор может возвращать промежуточные признаки с разных уровней сети.

Схема архитектуры дискриминатора

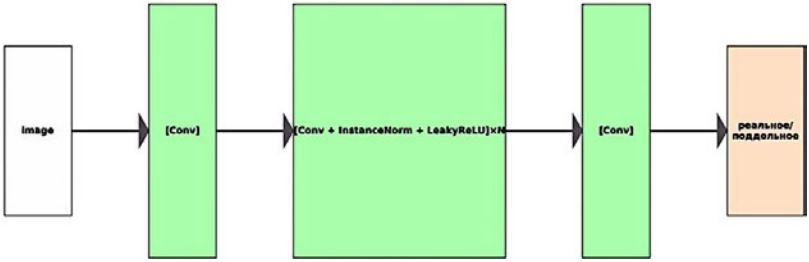


Рис. 2. Схема архитектуры дискриминатора

Обоснование выбранного подхода

– SpectralNorm и InstanceNorm обеспечивают устойчивость и стабильность обучения.

– Глубокая архитектура позволяет дискриминатору выявлять сложные паттерны и артефакты.

– Возможность извлекать признаки для дополнительных функций потерь.

Альтернативы

– Использование PatchGAN (оценка реалистичности по патчам).

– Внедрение attention-механизмов в дискриминатор.

– Классический BatchNorm вместо InstanceNorm.

5. Функции потерь и обучение

Обучение GAN — это тонкий баланс между генератором и дискриминатором. Для достижения высокого качества используются несколько функций потерь [2].

Adversarial Loss (Hinge Loss)

– Для дискриминатора:

$$L_D = [(0,1 - D(x_{real}))] + [(0,1 + D_{fake})]$$

– Для генератора: $L_G = -[D(x_{fake})]$

– Hinge Loss способствует более стабильному обучению, предотвращает «затухание» градиентов.

Перцептивная функция потерь (Perceptual Loss)

– Сравнивает не сами пиксели, а признаки, извлечённые из промежуточных слоёв предобученной сверточной сети (например, VGG16):

$$L_{perc} = u \mid l(x_{real}) - l(x_{fake}) \mid_1$$

– Позволяет генератору создавать изображения, которые визуально ближе к реальным [2].

Feature Matching Los

– L1-расстояние между признаками, извлечёнными дискриминатором из реальных и сгенерированных изображений:

$$L_{fm} = i \mid f_i(x_{real}) - f_i(x_{fake}) \mid_1$$

– Способствует устойчивости и разнообразию генерации.

Diversity Loss

– Стимулирует генератор создавать разнообразные изображения:

$$L_{div} = _{ij} \| z_i - z_j \|_2 - \mid G(z_i) - G(z_j) \|_2 \mid$$

– Особенно важен для предотвращения коллапса мод (mode collapse).

Градиентный штраф (Gradient Penalty)

– Для стабилизации обучения используется градиентный штраф:

$$L_{gp} = [(\mid D(image) \mid)_2 - 1)^2]$$

– Где (*image*) — интерполированное изображение между реальным и сгенерированным.

Итоговая функция потерь

– Общая функция потерь генератора:

$$L_{total} = L_{adv} + percL_{perc} + fmL(f_m) + divL_{div}$$

– Балансировка ()-коэффициентов позволяет управлять вкладом каждой функции потерь.

Влияние функций потерь на обучение

– Adversarial Loss: основной драйвер генерации реалистичных изображений.

– Perceptual Loss: улучшает визуальное качество, делает изображения более “естественными”.

– Feature Matching: способствует устойчивости и разнообразию.

– Diversity Loss: предотвращает коллапс мод, увеличивает вариативность.

– Gradient Penalty: стабилизирует обучение, предотвращает взрыв градиентов.

Примеры ситуаций

– Если генератор выдает однотипные изображения — увеличить вес diversity loss.

– Если изображения выглядят размытыми — усилить perceptual loss.

– Если обучение нестабильно — увеличить gradient penalty.

6. Эксперименты и результаты

– Обучение проводилось на GPU (DirectML/AMD).

– Использовались различные веса для регуляризации и стабилизации.

– В процессе сохранялись чекпоинты, графики потерь, примеры изображений.

Пример графика потерь: loss_plot (рис. 3)

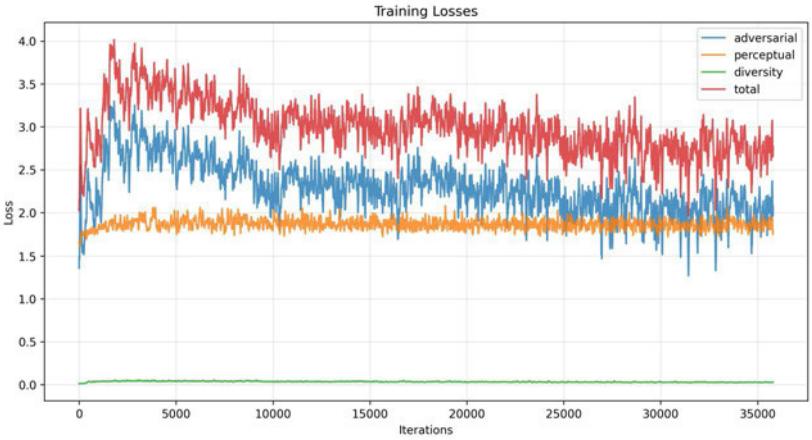


Рис. 3. График потерь на 50 эпохе

7. Постобработка задачи сверхразрешения цифровых изображений

Хотя GAN способен генерировать реалистичные изображения, их разрешение существенно ограничено (например, 128×128 пикселей). Для большинства прикладных задач это является существенным ограничением, так как такие изображения не могут передать достаточный объем информации об объектах сцены и её мелких деталях. Решением данной проблемы является компенсация этой информации путём восстановления высокочастотной составляющей изображения на основе эффекта сверхразрешения.

Рассмотрим модель системы формирования цифровых изображений на вход которой поступает дискретное представление двумерного сигнала $I_{HR} \in \mathbb{R}^{L \times L}$, характеризующего распределение интенсивности светового потока в плоскости датчиков камеры:

$$I_{LR} = H(I_{HR}) + n = D\left(B(I_{HR})\right) + n,$$

где *не доступный для наблюдения* сигнал I_{HR} является изображением с высоким разрешением (ВР), в то время как его *наблюдаемая версия* с низким разрешением (НР) $I_{LR} \in \mathbb{R}^{M \times M}$, $M < L$, возникает в результате воздействия на I_{HR} эффектов размытия, обусловленных влиянием атмосферных искажений и оптической системы, понижающей дискретизации $D()$, характеризующей ограниченное количество фотоприемников в матрице камеры, и аддитивного шума n, возникающего вследствие недостаточного объема светового излучения, попадающего на поверхность фотоприёмника в силу их малого размера и краткого времени съемки одного кадра.

Перечисленные эффекты приводят к потере высокочастотной составляющей исходного сигнала, что проявляется в низкой детализации наблюдаемого изображения. Повышение детализации возможно в результате восстановления потерянной информации на основе обратного преобразования $G_{SR} = H^{-1}$:

$$I_{HR} \approx I_{SR} = H^{-1}\left(I_{LR}\right) = G_{SR}(I_{LR}),$$

где I_{SR} является оценкой изображения ВР, получаемой на основе дополнительной информации о наблюдаемой сцене, которую, в частности, может обеспечить соответствующая модель глубокой нейронной сетей.

В данной работе для повышения разрешения использовалась Real-ESRGAN (Real-Enhanced Super-Resolution Generative Adversarial Network) — современная модель для, способная восстанавливать детали и улучшать визуальное восприятие изображений, которая является усовершенствованной версией ESRGAN, обученной на большом числе реальных изображений. В архитектуре данной модели используется вложенные остаточные блоки (Residual-in-Residual Dense Block) и специальные техники для борьбы с артефактами, что позволяет восстанавливать детали даже на сложных текстурах и получать изображения высокого качества.

Интеграция Real-ESRGAN в пайплайн

После генерации изображения GAN (например, 128×128), оно передаётся в Real-ESRGAN, где увеличивается до 512×512 или 1024×1024 пикселей. Такой подход позволяет получить итоговые изображения, пригодные для публикаций, печати и дальнейшего анализа.

Общая схема пайплайна показана на рис. 4.

$$Text \rightarrow [TextEncoder] \rightarrow [GAN] \rightarrow (128x128) \rightarrow$$
$$\rightarrow [Real-ESRGAN] \rightarrow (512x512) \rightarrow FinalResult$$

Рис. 4. Схема пайплайна интеграции

Практические аспекты интеграции

– Для интеграции Real-ESRGAN требуется предварительно обученная модель и соответствующие веса.

– Постобработка проводится автоматически после генерации каждого изображения.

– Изображения до и после повышения разрешения сохраняются раздельно. При необходимости переноса данных достаточно сохранить лишь исходные сгенерированные изображения, на основе которых в дальнейшем можно воспроизвести более объемные версии в высоком разрешении. Также это облегчает возможность их визуального сравнения в различных разрешениях.

– Такой подход особенно полезен для подготовки публикаций, презентаций и иных вариантов практического применения, например, в области создания дизайна.

Визуальное сравнение

На рис. 5 показаны примеры сгенерированных изображений на основе текстовых описаний: «yellow flower with orange center» (желтый цветок с оранжевой серединкой) (рис. 5,а), «pink flower with green leaves» (розовый цветок с зелёными листьями) (рис. 5,б). Сгенерированные изображения удовлетворяют текстовым запросам, но в силу малого разрешения (128x128), способны передать лишь общую информацию о сцене, которой явно недостаточно для того, чтобы различить небольшие элементы.

На рис. 6 показаны детализированные версии предыдущих изображений, полученные в результате четырехкратного повышения разрешения с использование модели Real-ESRGAN до размеров 512x512.

а б

Рис. 5. Результаты генерации для различных вариантов текстового описания: а – «yellow flower with orange center», б – «pink flower with green leaves»

а б

Рис. 6. Результаты генерации с последующим четырехкратным повышением разрешения: а – результат изображения на рис. 5,а, б – результат повышения разрешения для изображения на рис. 5,б

Заключение

В работе реализован полный пайплайн генерации изображений на основе текстовых описаний сцены с помощью условных GAN, автоматической генерации текстовых описаний и постобработки с

помощью Real-ESRGAN. Архитектура учитывает современные подходы (Self-Attention, Feature Matching, Perceptual Loss), что позволяет получать реалистичные и разнообразные изображения, пригодные для практического использования в области создания дизайна.

Список литературы

1. Generative Adversarial Networks / I. J. Goodfellow [et al.] // arXiv:1406.2661 [stat.ML] – DOI: https://doi.org/10.48550/arXiv.1406.2661. – Режим доступа: https://arxiv.org/pdf/1406.2661
2. Отырба Р. Р. Модели и методы N-shot обучения и их применение в задачах семантической сегментации изображений: систематизированный обзор. Ч. I – zero-shot обучение / Р. Р. Отырба, А. А. Сирота // Вестник ВГУ. Серия: Системный анализ и информационные технологии. – 2024. – № 1. – С. 149–165. – DOI: https://doi.org/10.17308/sait/1995-5499/2024/1/149-165. – ISSN 1995-5499.
3. Отырба Р. Р. Модели и методы N-shot обучения и их применение в задачах семантической сегментации изображений: систематизированный обзор. Ч. II, one-shot и few-shot обучение / Р. Р. Отырба, А. А. Сирота // Вестник ВГУ. Серия: Системный анализ и информационные технологии. – 2024. – № 2. – С. 152–172. – DOI: https://doi.org/10.17308/sait/1995-5499/2024/2/152-172. – ISSN 1995-5499.
4. Mirza, M. Generative Adversarial Networks / M. Mirza, S. Osindero // arXiv:1411.1784 [cs.LG] – DOI: https://doi.org/10.48550/arXiv.1411.1784. – Режим доступа: https://arxiv.org/pdf/1411.1784
5. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks / H. Zhang [et al.] // arXiv:1612.03242 [cs.CV] – DOI: https://doi.org/10.48550/arXiv.1612.03242. – Режим доступа: https://arxiv.org/pdf/1612.03242
6. AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks / T. Xu [et al.] // arXiv:1711.10485 [cs.CV] – DOI: https://doi.org/10.48550/arXiv.1711.10485. – Режим доступа: https://arxiv.org/pdf/1711.10485
7. 102 Category Flower Dataset [Электронный ресурс] : датасет. – Режим доступа : https://www.robots.ox.ac.uk/~vgg/data/flowers/102/
8. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation / Li. Junnan [et al.] // arXiv:2201.12086 [cs.CV] – DOI: https://doi.org/10.48550/arXiv.2201.12086. – Режим доступа: https://arxiv.org/pdf/2201.12086

Исследование технологии Kotlin Multiplatform при разработке web-приложений

П. В. Смирных

Студент магистр

А. Л. Гавшин

Старший преподаватель

Введение

Современная практика программной инженерии характеризуется широким разнообразием языков и технологий, каждые из которых обладает определёнными достоинствами и ограничениями. Однако использование нескольких языков программирования в одном проекте часто приводит к увеличению сложности разработки, снижению читаемости кода и необходимости освоения различных инструментальных средств. В этой связи актуальным становится поиск унифицированного подхода, позволяющего реализовать полноценное приложение на основе одного языка программирования.

Одним из таких решений является язык Kotlin, сочетающий современный синтаксис, высокую производительность и расширенные возможности мультиплатформенной разработки. Представленный в 2011 году компанией JetBrains, Kotlin активно развивается и поддерживается сообществом, а также получил официальную поддержку со стороны Google. Несмотря на внешнее сходство с Java, Kotlin предоставляет возможность безопасно работать с null, расширенные функциональные возможности и уникальную возможность трансляции кода в JavaScript, что делает его перспективным инструментом для разработки FullStack-приложений [1]. Использование Kotlin Multiplatform позволяет разрабатывать клиентскую и серверную части приложения на одном языке, обеспечивая повторное использование логики между платформами.

В этой работе представлен анализ возможностей использования технологии Kotlin Multiplatform в разработке клиент-серверных приложений. Рассмотрены её основные преимущества и недостатки, оценены перспективы развития, а также проведено сравнение с другими технологиями, применяемыми в области FullStack-разработки. Особое

© Смирных П. В., Гавшин А. Л., 2025

внимание уделено интеграции Kotlin Multiplatform с фреймворком Ktor и использованию единого языка программирования для реализации как клиентской, так и серверной частей приложения.

1. Архитектурный обзор технологии Kotlin Multiplatform

Kotlin Multiplatform представляет собой механизм, позволяющий создавать общие модули кода, которые могут быть использованы на различных платформах — от мобильных (Android, iOS) до веб- и серверных приложений [2]. Одним из ключевых сценариев применения технологии является совместное использование бизнес-логики между клиентской и серверной частью приложения.

На рис. 1 демонстрируется то, какие файлы создаются при инициализации проекта. Также здесь можно увидеть отдельные модули, отвечающие за определенные задачи.

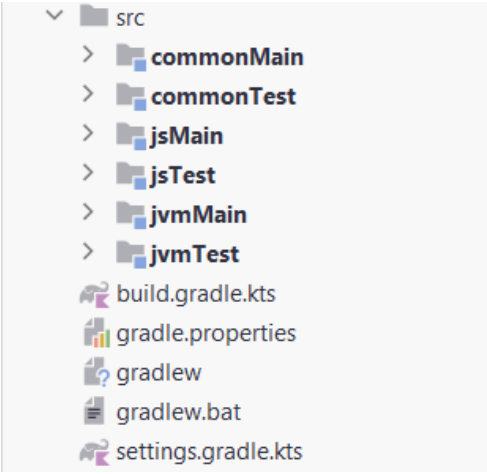


Рис. 1. Структура проекта на момент создания

Структура проекта Kotlin/MPP выражена созданием 6 модулей [3]:

- commonMain – содержит общую логику, которая может быть использована на всех целевых платформах;
- commonTest – модуль автоматизированных тестов для проверки работы общих компонентов;
- jvmMain – реализация серверной части приложения, ориентированная на выполнение на Java Virtual Machine;
- jvmTest – тестирование серверных компонентов;

– jsMain – реализация клиентской части приложения, предназначенная для запуска в браузере;
– jsTest – тестирование клиентских компонентов.

Проект сопровождается конфигурационными файлами, используемыми системой сборки Gradle. Формат файлов зависит от используемого DSL: в данном случае применяется Kotlin DSL, что выражается в использовании файлов с расширением .kts. Также есть возможность выбора формата .properties или .yaml для хранения параметров конфигурации.

Основные файлы конфигурации включают:

– settings.gradle.kts – в нем описываются константы, пример: имя проекта, версия какого-либо подключаемого в зависимостях модуля в build.gradle.kts [4];
– gradle.properties – содержит параметры настройки среды разработки, баз данных, безопасности и других аспектов [5];
– build.gradle.kts – главный файл сборки, в котором указываются зависимости, настройки плагинов, версии используемых технологий и различные задачи.

На рис. 2 представлена диаграмма развертывания системы, состоящая из трёх основных компонентов.

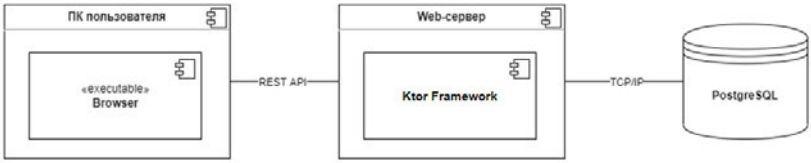


Рис. 2. Диаграмма развертывания

Клиентская часть – реализована в виде веб-интерфейса, взаимодействующего с сервером посредством REST API;

Серверная часть – построена на основе фреймворка Ktor, обеспечивающего маршрутизацию запросов, обработку данных и взаимодействие с базой данных;

База данных – отдельный компонент, реализующий хранение и обработку данных, взаимодействует с серверной частью через протокол TCP/IP.

2. Сравнительный анализ технологий разработки FullStack-приложений

Разработка клиент-серверных приложений предполагает выбор технологического стека, который обеспечивает баланс между

производительностью, простотой поддержки и возможностями масштабирования. На сегодняшний день существует несколько устоявшихся экосистем, каждая из которых имеет свои особенности [6].

Для сравнения были выбраны следующие технологии:

– Kotlin Multiplatform — как современное решение, позволяющее создавать единые модули для клиента и сервера;
– JS-ориентированные стеки (MERN/MEAN) — популярные решения для веб-разработки на основе JavaScript/TypeScript;
– Python-экосистема (Django/Flask + React) — активно используется в образовательных и MVP-проектах;
– Java-экосистема (Spring Boot + Angular/React) — доминирующая технология в корпоративной среде;
– Go-экосистема (Gin/Echo + Svelte/React) — набирающий популярность стек, ориентированный на высокую производительность.

Анализ проводился по трём ключевым критериям: наличие единой кодовой базы, поддержка нескольких платформ и уровень производительности. Результаты представлены в таблице.

Таблица

Сравнительный анализ технологий

Технология	Единая кодовая база	Поддержка нескольких платформ	Производительность
Kotlin Multiplatform	есть	есть	высокая
JS-ориентированный стек	есть	нет	средняя
Python-экосистема	нет	нет	средняя
Java-экосистема	нет	нет	высокая
Go-экосистема	нет	нет	очень высокая

Интерпретация оценок:

1. Единая кодовая база. Kotlin Multiplatform позволяет использовать один язык программирования (Kotlin) для реализации всех частей приложения — от мобильного клиента до серверной логики. Помимо него только JavaScript может применяться в разных частях одного приложения, но для этого потребуется использовать несколько технологий для написания клиента, сервера, настройки сетевого взаимодействия и прочего. Все остальные подходы требуют использования разных языков программирования для клиента и сервера.

2. Поддержка нескольких платформ. Kotlin Multiplatform демонстрирует наибольшую гибкость, поддерживая разработку под Android, iOS, веб и десктоп. Остальные технологии в основном ограничиваются веб-платформой.

3. Производительность. Высокий уровень производительности показывают Kotlin Multiplatform, Java и Go за счёт использования JVM или компиляции в нативный код. JS-стек и Python уступают по данному параметру, особенно в задачах, требующих интенсивных вычислений или обработки большого объёма данных.

Заключение

В рамках исследования была проведена оценка возможностей технологии Kotlin Multiplatform в контексте разработки клиент-серверных приложений. Было реализовано приложение, демонстрирующее эффективность использования единого языка программирования для реализации полного цикла разработки — от клиентского интерфейса до серверной логики и взаимодействия с базой данных. Применение фреймворка Ktor позволяет организовать гибкую архитектуру, допускающую дальнейшее расширение функциональности.

Рассмотренные технологии активно развиваются. Регулярные обновления, расширение функциональности, исправление ошибок и развитие документационной базы свидетельствуют о высоком потенциале Kotlin Multiplatform в качестве основы для современных FullStack-решений. Разработчикам доступны обучающие материалы, примеры проектов и подробная техническая документация, что способствует популяризации и внедрению этих технологий в профессиональной среде.

Проведённый сравнительный анализ позволил установить ключевые достоинства Kotlin Multiplatform как универсального инструмента для FullStack-разработки. Он выгодно отличается от JS-ориентированных, Python-, Java- и Go-экосистем своей мультиплатформенностью, единым языком программирования и гибкой архитектурой. Эти характеристики делают его перспективным выбором для создания современных клиент-серверных приложений.

Список литературы

1. Котлин для JavaScript [Электронный ресурс] : статья. – Режим доступа : https://kotlinlang.ru/docs/js-overview.html

2. Доклад Яндекса [Электронный ресурс] : статья. – Режим доступа : https://habr.com/ru/company/yandex/blog/542454

3. Kotlin Multiplatform Documentation [Электронный ресурс] : документация на программное обеспечение. – Режим доступа : https://kotlinlang.org/docs/multiplatform.html
4. Gradle User Manual [Электронный ресурс] : документация на программное обеспечение. – Режим доступа : https://docs.gradle.org/current/userguide/userguide.html
5. Ktor Documentation [Электронный ресурс] : документация на программное обеспечение. – Режим доступа : https://ktor.io/docs/welcome.html
6. Popular Tech Stacks for Full-Stack Development in 2024 [Электронный ресурс] : статья. – Режим доступа : https://dev.to/adityabhuyan/popular-tech-stacks-for-full-stack-development-in-2024-1ogj

Исследования возможностей GeoServer в контексте разработки геоинформационных систем

П. В. Смирных

Студент магистр

А. Л. Гавшин

Старший преподаватель

Введение

Геоинформационные системы (ГИС) играют важную роль в обработке, хранении, анализе и визуализации пространственных данных. С развитием интернет-технологий все большее значение приобретают веб-ГИС, позволяющие предоставлять геопространственные данные и сервисы через веб-интерфейс, что делает их доступными широкому кругу пользователей. Для реализации таких решений необходимы серверные компоненты, способные обрабатывать запросы на отображение картографических данных, выполнять пространственные операции и предоставлять API для взаимодействия с клиентскими приложениями.

Одним из наиболее популярных решений в этой области является GeoServer – свободно распространяемый серверный продукт с открытым исходным кодом, реализующий стандарты OGC (Open Geospatial Consortium), такие как WMS, WFS, WCS и другие [1].

В этой работе представлены результаты анализа возможностей использования программного обеспечения GeoServer в качестве основы для разработки и реализации веб-ГИС. Представлены ключевые аспекты архитектуры, функциональные возможности и технологические преимущества GeoServer. Также проведено сравнение с другими популярными решениями в области веб-ГИС, такими как MapServer, QGIS Server и ArcGIS Server.

1. Обзор архитектуры и функциональных возможностей GeoServer

GeoServer имеет модульную архитектуру, которая позволяет расширять функциональность системы за счет дополнительных компонентов [2]. В основе системы лежит каталог данных, где хранится информация о слоях, стилях, рабочих пространствах и других ресурсах. Основные компоненты включают:

© Смирных П. В., Гавшин А. Л., 2025

– WMS (Web Map Service) – сервис, предоставляющий растровые данные;
– WFS (Web Feature Service) – сервис, предоставляющий векторные данные в форматах GML, GeoJSON, CSV и других;
– WCS (Web Coverage Service) – сервис, предоставляющий многомерные растровые данные (пример – снимки со спутника);
– CSW (Catalogue Service for the Web) – сервис управления метаданными;
– REST API – интерфейс для программного доступа к данным и конфигурации;
– плагины – модули, добавляющие новые возможности.

Такая организация позволяет GeoServer адаптироваться под разнообразные задачи.

Важным аспектом при разработке ГИС является возможность использования источников данных разных типов. GeoServer предоставляет эту возможность, что делает его гибким решением в условиях различных инфраструктур. Вот некоторые примеры поддерживаемых источников данных:

– векторные данные: Shapefile, GeoPackage, PostGIS, Oracle Spatial, SQL Server;
– растровые данные: GeoTIFF, ECW, JPEG2000, NetCDF, GRIB;
– сетевые источники: сторонние WMS, WFS и WCS сервисы, ArcGIS REST;
– NoSQL / Big Data: MongoDB, Hana, Cassandra;
– табличные данные: CSV, Excel, JDBC;
– облачные хранилища: Amazon S3, Google Cloud Storage, Azure Blob Storage (через плагины).

Такая поддержка позволяет использовать GeoServer в гетерогенных средах, где данные могут находиться в разных форматах и местах.

Источник данных в GeoServer также именуется хранилищем. Каждое хранилище относится к рабочему пространству имен. Минимальная единица, с которой происходит работа – слой. Он относится к одному хранилищу, может быть разного типа и хранить разное количество информации. Например, слой может быть векторным и описывать координаты лишь одной точки, а может быть растровым и содержать детальное изображение территории всей планеты.

Также в GeoServer есть возможность объединять слои в группы даже в случае, если они заданы в разных координатных системах. Результат получения группы слоев в растровом формате показан на рисунке.

Рисунок. Группа слоев на базе растрового слоя, дополненного векторными слоями автомагистралей, крупных населенных пунктов и административно-территориального деления

Можно заметить, что линии границ и автомагистралей имеют разный цвет и форму. GeoServer позволяет настроить стили как для отдельного слоя, так и для слоя в составе группы слоев. Это дает возможность детальнее настраивать карту.

2. Сравнительный анализ технологий ГИС

Как и любой другой технологии у GeoServer существует ряд аналогов:

1. MapServer – это легковесное и высокопроизводительное open-source решение для создания веб-ГИС. Разрабатывается под лицензией MIT и ориентировано на создание базовых картографических сервисов. Настройка осуществляется через текстовые конфигурационные файлы, что делает его менее удобным для неквалифицированных пользователей. Поддерживает ключевые стандарты OGC, такие как WMS, WFS и WCS.
2. QGIS Server – это серверное расширение популярного десктопного ГИС-приложения QGIS. Использует ту же ядро-библиотеку, что и QGIS Desktop, что обеспечивает прямую интеграцию между рабочим столом и веб-сервисами. Подходит для организаций, уже использующих QGIS. Реализует WMS и частично WFS-T, но имеет ограниченную функциональность по сравнению с более мощными решениями.
3. ArcGIS Server — это коммерческое решение от компании ESRI, предназначенное для создания масштабируемых корпоративных

геосервисов. Интегрируется с другими продуктами ESRI (Portal for ArcGIS, ArcGIS Online), предлагает продвинутые аналитические и 3D-возможности. Требует лицензии и часто используется в крупных государственных и коммерческих проектах.

Анализ проводился по ключевым для ГИС критериям. Результаты представлены в таблице, данные для анализа были взяты из документации [3-5].

Таблица

Сравнительный анализ технологий ГИС

Критерий	GeoServer	MapServer	QGIS	ArcGIS
Открытая лицензия	да	да	да	нет
Поддержка OGC	полная	частичная	частичная	полная
Расширяемость	очень высокая	средняя	средняя	высокая
Быстродействие	высокое	очень высокое	среднее	высокое
Встроенный интерфейс	есть	нет	есть через QGIS Desktop	есть
Возможность облачного развертывания	есть	нет	нет	есть

Полученные результаты свидетельствуют о том, что GeoServer является одним из наиболее универсальных и доступных решений для разработки геоинформационных систем. Его ключевыми преимуществами являются:

- открытость и бесплатность;
- полная поддержка стандартов OGC;
- широкие возможности по интеграции с различными источниками данных;
- поддержка современных протоколов и форматов передачи данных.

Однако, как и любое ПО, GeoServer имеет свои ограничения. Например, его производительность может снижаться при работе с очень большими объемами данных без предварительной оптимизации. Кроме того, административный интерфейс, хотя и достаточно функционален, требует определенного уровня подготовки со стороны пользователя.

Заключение

В результате проведенного исследования установлено, что GeoServer представляет собой эффективное и надежное решение для разработки и внедрения веб-ГИС. Он сочетает в себе большое количество сервисов, поддержку различных форматов геопространственных данных и открытость, что делает его хорошим вариантом для научных, образовательных и коммерческих организаций.

Сравнительный анализ показал, что GeoServer занимает промежуточную позицию между легковесными решениями (MapServer) и коммерческими платформами (ArcGIS Server), предлагая разумный баланс между стоимостью, производительностью и гибкостью.

Список литературы

1. OGC Standards [Электронный ресурс] : документация на программное обеспечение. – Режим доступа : https://www.ogc.org/standards

2. GeoServer Documentation [Электронный ресурс] : документация на программное обеспечение. – Режим доступа : https://docs.geoserver.org

3. MapServer Project [Электронный ресурс] : документация на программное обеспечение. – Режим доступа : https://mapserver.org

4. QGIS Server Guide [Электронный ресурс] : документация на программное обеспечение. – Режим доступа : https://docs.qgis.org

5. ArcGIS Server Documentation [Электронный ресурс] : документация на программное обеспечение. – Режим доступа : https://enterprise.arcgis.com/ru/get-started/latest/windows/what-is-arcgis-enterprise-.htm

Анализ и сравнение встраиваемых операционных систем реального времени на базе STM32

В. А. Соколов
Студент магистр
А. Ю. Савинков
Профессор

Введение

Микроконтроллерные системы на базе STM32 находят широкое применение в самых различных отраслях: от промышленной автоматизации и робототехники до встраиваемых IoT-устройств, и носимой электроники. Рост сложности задач, предъявляемых к таким устройствам, требует эффективного управления многозадачностью, событийным взаимодействием, приоритетами и временем отклика. В этих условиях использование операционных систем реального времени (Real-Time Operating Systems, RTOS) становится необходимостью, особенно при разработке отказоустойчивых и масштабируемых приложений.

В отличие от простого программирования, подход с RTOS позволяет описывать логику устройства в виде набора задач, выполняемых параллельно с заранее заданными приоритетами. Дополнительно упрощается взаимодействие между задачами и периферией, повышается читаемость и сопровождаемость кода.

Наиболее востребованные RTOS в экосистеме STM32 включают в себя FreeRTOS, CMSIS-RTOS и Zephyr. Все три системы имеют открытый исходный код, активно развиваются и обеспечивают базовую поддержку многозадачности. Однако между ними существуют важные архитектурные различия, определяющие удобство разработки, область применения и требования к ресурсам микроконтроллера.

Цель данной статьи – провести обзор и сравнительный анализ трёх указанных RTOS с точки зрения архитектурных особенностей, степени интеграции в экосистему STM32 и применимости в различных сценариях. Статья ориентирована на практикующих разработчиков и студентов, осваивающих современные средства встраиваемого программирования.

© Соколов В. А., Савинков А. Ю., 2025

1. Архитектурные особенности RTOS

Операционные системы реального времени имеют схожую общую структуру, включающую ядро с планировщиком задач, таймеры, механизмы синхронизации (семафоры, мьютексы, события), очереди сообщений и средства управления временем [1]. Однако конкретная реализация этих компонентов, подход к конфигурации и поддержка современных стандартов могут сильно различаться.

1. FreeRTOS

FreeRTOS – одна из самых популярных RTOS с открытым исходным кодом, доступная под лицензией MIT. Основные особенности:

– Простой кооперативно-прерываемый планировщик;

– Поддержка приоритетов, семафоров, очередей, таймеров, событий;

– Минимальные накладные расходы – ядро может занимать менее 10 кБ памяти;

– Широкая поддержка в STM32CubeMX и STM32CubeIDE;

– Множество примеров и готовых проектов.

Архитектура FreeRTOS предполагает статическую и динамическую инициализацию задач. Каждая задача имеет собственный стек, идентификатор и приоритет. Планировщик может быть, как с вытеснением (preemptive), так и кооперативным. Системные тики могут задаваться как от SysTick, так и от других таймеров.

FreeRTOS не поддерживает пространственное разделение (memory protection), но может использоваться совместно с MPU (Memory Protection Unit) в защищённых приложениях. Отладка поддерживается через trace-макросы и сторонние инструменты (например, FreeRTOS+Trace).

2. CMSIS-RTOS

CMSIS-RTOS – это не столько самостоятельная RTOS, сколько стандарт API, разработанный компанией ARM в составе спецификации CMSIS (Cortex Microcontroller Software Interface Standard). Он представляет собой обёртку, обеспечивающую единый интерфейс доступа к функциям RTOS независимо от её реализации.

На практике наиболее часто используется реализация CMSIS-RTOS на базе RTX (от Keil), однако может использоваться и поверх FreeRTOS. Это позволяет:

– Легко переносить код между проектами;

– Работать в единой среде с HAL и другими компонентами CMSIS;

– Использовать стандартные имена и сигнатуры API-функций.

Особенностью CMSIS-RTOS является статическая диспетчеризация задач и строгая типизация объектов ядра. Интерфейс описан в виде функций osThreadNew(), osSemaphoreNew(), osMessageQueuePut() и т.д. Работа с таймерами, событиями и планировщиком ведётся также через универсальные вызовы.

Отдельного внимания заслуживает CMSIS-RTOS2 – более новая версия интерфейса, которая учитывает современные требования к безопасности, совместима с C++, поддерживает многопоточность и допускает асинхронную инициализацию объектов. Однако количество примеров и активность сообщества уступают FreeRTOS и Zephyr.

3. Zephyr RTOS

Zephyr – это мощная модульная RTOS с POSIX подобным API, разрабатываемая Linux Foundation и поддерживаемая множеством компаний (Intel, Nordic, ST и др.). В отличие от FreeRTOS и CMSIS-RTOS, Zephyr изначально проектировалась как полноценная кроссплатформенная платформа с поддержкой десятков архитектур и микроконтроллеров.

Её ключевые особенности:

– Микроядерная архитектура: разделение ядра, системных служб и пользовательских приложений;

– Модульная сборка: только нужные компоненты включаются в бинарный образ;

– Использование Device Tree, Kconfig и west для конфигурации;

– Поддержка BLE, TCP/IP, USB, файловых систем, безопасности и других программных систем.

Zephyr требует больше памяти и времени на освоение, но взамен предоставляет полноценную инфраструктуру для промышленного применения. Сильной стороной также является развитая система тестирования, CI/CD, поддержка CMake и возможность симуляции на QEMU.

Для STM32 Zephyr предоставляет множество пакетов поддержки плат (BSP) и драйверов. Тем не менее, интеграция требует отдельной настройки окружения, установки Zephyr SDK и понимания концепции построения проекта.

2. Интеграция RTOS в проекты STM32

Практическая интеграция RTOS в проект на базе STM32 зависит как от выбранной операционной системы, так и от используемой среды разработки [2]. Наиболее распространённым инструментом для разработки является STM32CubeIDE, сочетающий в себе STM32CubeMX и компилятор с отладчиком. Ниже рассмотрим особенности настройки каждой из RTOS в контексте STM32.

1. FreeRTOS

FreeRTOS имеет наивысшую степень интеграции в STM32CubeMX – его поддержка встроена напрямую [3]. Пользователь может в графическом интерфейсе выбрать RTOS (вкладка Middleware, пункт FreeRTOS), создать задачи, задать приоритеты, настроить таймеры, очереди и семафоры. CubeMX сгенерирует соответствующий каркас проекта, включая заголовочные и исходные файлы, инициализацию и шаблоны задач. Его основными преимуществами являются:

– Автоматическая генерация кода;
– Быстрая настройка планировщика;
– Поддержка символьной отладки задач;
– Совместимость с HAL и LL-драйверами ST.

Такая интеграция делает FreeRTOS первым выбором для большинства прикладных проектов на STM32, особенно в учебной и прототипной среде.

2. CMSIS-RTOS

CMSIS-RTOS не представляет собой самостоятельную RTOS, но может быть выбран в CubeMX в качестве API-слоя, поверх которого работает FreeRTOS или другая ОС [4]. В среде STM32CubeIDE можно указать тип интерфейса (CMSIS v1 или v2), после чего пользователь взаимодействует с системой через функции osThreadNew, osDelay, osSemaphoreAcquire и так далее. Его основными преимуществами являются:

– Независимость от реализации;
– Облегчённый перенос проектов между RTOS;
– Стандартизированная документация от ARM.

К недостаткам можно отнести следующие пункты:

– Ограниченная поддержка примеров;
– Некоторые функции реализуются поверх FreeRTOS с потерей производительности.

CMSIS-RTOS удобен в контексте образовательных задач и при разработке переносимых библиотек.

3. Zephyr

Интеграция Zephyr в STM32 требует отдельного подхода. Проекты создаются вне STM32CubeIDE, с использованием кроссплатформенной системы сборки CMake, конфигурации через Kconfig и описание оборудования через Device Tree.

Для запуска проекта пользователь создаёт рабочее окружение, указывает целевую плату, компилирует проект через командную строку и загружает бинарный файл через OpenOCD или st-link.

Сильной стороной Zephyr является гибкость и масштабируемость. Слабой – относительная сложность входа и необходимость изучения нового инструментария.

3. Сценарии применения и критерии выбора RTOS

Выбор RTOS зависит от требований к проекту: необходима ли сетевая функциональность, насколько важны ресурсоэффективность, защищённость, масштабируемость, доступность драйверов и инструментов. Ниже представлены типичные сценарии использования каждой ОС.

1. FreeRTOS

FreeRTOS рекомендуется для быстрого прототипирования, учебных и лабораторных проектов, однозадачных и средненагруженных систем, проектов с ограниченными ресурсами [5]. FreeRTOS обеспечивает хорошую документацию, поддержку сообщества, примеры, и легко осваивается благодаря интеграции в CubeMX.

2. CMSIS-RTOS

Подходит для образовательных курсов, где акцент на API, унифицированных библиотек, абстрагированных от конкретной RTOS, проектов, где возможно переключение между RTOS. Ограничения связаны с меньшей гибкостью и меньшим количеством примеров по сравнению с FreeRTOS.

3. Zephyr

Рекомендуется для: проектов с требованием BLE, Wi-Fi, USB, TLS, OTA, IoT-устройств с полной стековой инфраструктурой, промышленных контроллеров с надёжной защитой памяти и задач. Порог входа в Zephyr выше, но он оправдан, если необходимо использование современных протоколов и встроенной поддержки безопасности.

Заключение

Реализация многозадачности на микроконтроллерах STM32 становится всё более востребованной благодаря росту сложности встроенных систем и необходимости одновременного выполнения нескольких функций в реальном времени. Использование операционных систем реального времени (RTOS) позволяет не только упростить структуру кода, но и повысить надёжность и масштабируемость разрабатываемых решений.

Проведённый сравнительный анализ трёх подходов к построению RTOS-архитектуры – FreeRTOS, CMSIS-RTOS и Zephyr – позволяет сделать следующие выводы:

FreeRTOS является оптимальным выбором для большинства задач на STM32 благодаря лёгкости интеграции, поддержке со стороны ST и наличию множества примеров и документации. Он особенно эффективен в образовательных, лабораторных и прототипных проектах.

CMSIS-RTOS представляет собой универсальный интерфейс, стандартизированный компанией ARM, и подходит для проектов, где важна переносимость и унификация кода. Однако он требует более глубокого понимания архитектуры RTOS и ограничен в гибкости.

Zephyr предназначен для проектов промышленного уровня и IoT-систем, предоставляя современный набор функций, включая стеки BLE, TLS, файловые системы и механизмы защиты памяти. Его освоение требует большего времени и знаний, но он предлагает возможности, недоступные в упрощённых системах.

Выбор RTOS должен определяться характером задачи, ресурсами проекта и требуемым уровнем абстракции. Независимо от выбора, все рассмотренные RTOS демонстрируют высокую степень адаптации к платформе STM32 и открывают широкие возможности для построения сложных встраиваемых систем.

Список литературы

1. Valvano, J. W. Embedded Systems: Real-Time Operating Systems for ARM ® Cortex-M Microcontrollers: Textbook for EE445M/EE380L.6 / J. W. Valvano – Fourth Edition – CreateSpace Independent Publishing Platform – V. 3. – 569 p.

2. Carmine, N. Mastering STM32 / Carmine N. ; Leanpub, 2022 – 910 p.

3. STMicroelectronics [Электронный ресурс]. FreeRTOS и STM32: руководство пользователя в STM32CubeIDE. – Режим доступа: https://www.st.com/resource/en/user_manual/um1722-developing-applications-on-stm32cube-with-rtos-stmicroelectronics.pdf

4. ARM Developer [Электронный ресурс]. Документация CMSIS RTOS API v2. – Режим доступа: https://arm-software.github.io/CMSIS_5/RTOS2/html/index.html

5. Prototype Implementation of Temperature Control System with CAN and FreeRTOS on STM32F407 Discovery Boards [Электронный ресурс] : Venkat M., Mitchell N. / CAINE – Режим доступа: https://www.researchgate.net/publication/340314680

Разработка и анализ эффективности электромиографического интерфейса в рамках управления дроном

Р. А. Токарев

Студент магистр

Я. А. Туровский

Профессор

Введение

Современные беспилотные летательные аппараты (БПЛА) находят широкое применение в самых различных сферах — от индустрии развлечений до медицины и спасательных операций. Одним из перспективных направлений является использование интерфейсов взаимодействия "человек-машина" на основе биологической обратной связи, в частности — электромиографических сигналов (ЭМГ), позволяющих реализовать интуитивное управление устройствами при помощи мышечной активности.

В данной работе рассматривается реализация и апробация экспериментального программно-аппаратного комплекса для управления дроном DJI Tello EDU с использованием электромиографического интерфейса. Целью исследования стало создание простого и доступного метода управления дроном, не требующего традиционных органов управления, а также сравнение эффективности управления с помощью мышечной активности и традиционного клавиатурного интерфейса в задаче прохождения полосы препятствий.

Описание эксперимента

В ходе эксперимента электромиографический интерфейс управления (МИ) представлял собой программно-аппаратных комплекс, обеспечивающий регистрацию электромиограмы и превращение её в набор дискретных команд. Для этого использовались несколько кожных электродов, подключенных к человеку, который являлся оператором. Сигнал подается в программу для чтения и записи сигнала, после чего, путем передачи через локальный сервер, поступает в основную программу для его анализа и возможной подачи команды. В ходе

© Токарев Р. А., Туровский Я. А., 2025

обработки сигнал сперва децимируется, после чего сглаживается методом скользящего среднего:

$$\bar{X}(k) = \frac{1}{n}\sum_{t=k}^{n+k} X(t), \tag{1}$$

где n - размер окна (период сглаживания), а k — номер члена ряда, значение которого заменяется средним.

После этого находятся абсолютные значения методом вычитания соответствующих значений сглаженного и исходного сигнала по модулю. После этого все эти разности складываются, образуя накопленную сумму, которая, впоследствии, и будет служить индикатором степени активности мышц, который будет использоваться для управления дроном.

$$Sum = \sum_{i=1}^{k} \left| x_i - x_i' \right|, \tag{2}$$

где Sum - значение накопленной суммы значений, x_i - значения продецимированного сигнала, x_i' - значения сглаженного продецимированного сигнала.

Дрон управляется с помощью дискретной системы управление с секторальным представлением команд. Когда оператор приводит мышцу в напряжение, команда, которая в этот момент была активна, подается на дрон до тех пор, пока оператор не расслабит мышцу.

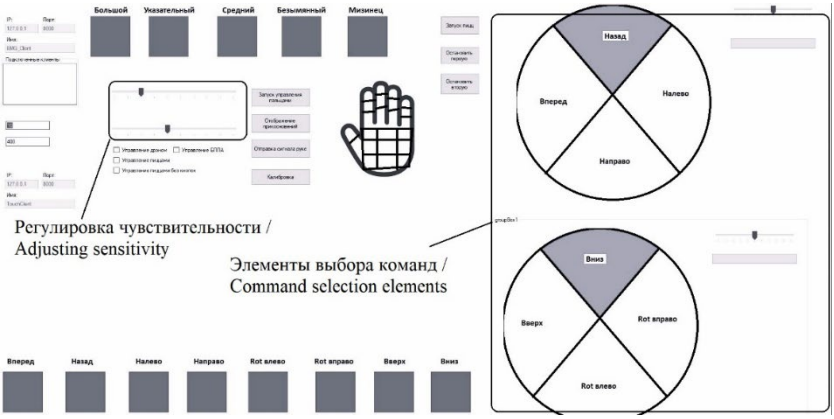


Рис. 1. Внешний вид программы

Также на интерфейсе присутствуют регулировка чувствительности, предназначенная для подбора необходимых параметров управления для людей с разными особенностями электрогенеза мышц. На экране у оператора находится вид с камеры дрона и необходимые элементы управления (рис. 2).

Рис. 2. Общий вид эксперимента

Каждая рука испытуемого управляла своим блоком команд в случае МИ. В случае клавиатурного интерфейса (КИ) положение рук пользователь выбирал исходя из своих предпочтений. Задачей испытуемого было с максимально возможной скоростью преодолеть полосу препятствий из трёх рядов по трое ворот в каждом, при этом в каждом ряду были ворота разного размера (большого среднего и малого) и расположенных на разной высоте(на земле, на высоте 80 см от пола по нижней границе ворот и на высоте 150 см от пола по нижней границе ворот). Оператора самостоятельно выбирал траектории перемещения по полосе препятствий, решая через какие ворота в каждом ряду он должен пройти.

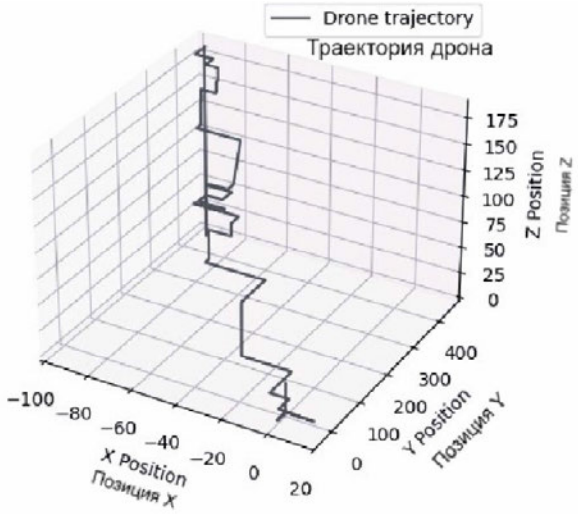


Рис. 3. Траектория полета дрона под управлением с клавиатуры

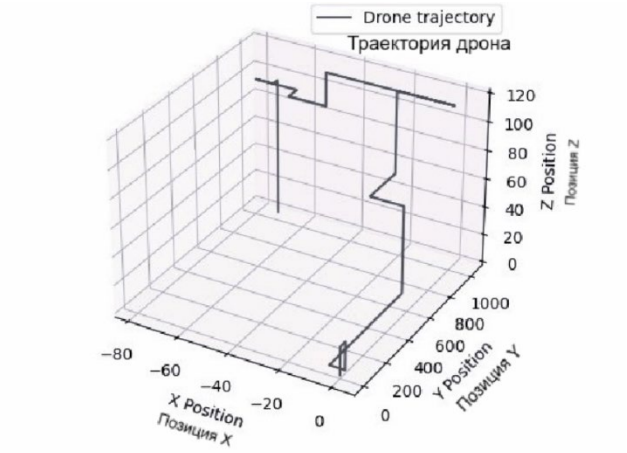


Рис. 4. Траектория полета дрона под управлением на основе мышечной активности

Заключение

В ходе эксперимента был реализован и протестирован электромиографический интерфейс управления дроном DJI Tello EDU.

Полученные результаты показали, что управление дроном с помощью мышечной активности возможно и обеспечивает достаточный уровень точности и отзывчивости при выполнении базовых навигационных задач.

Сравнительный анализ управления с помощью клавиатуры и ЭМГ-интерфейса показал, что хотя традиционный способ управления обеспечивает большую точность и скорость при прохождении маршрута, использование электромиографических сигналов представляет собой перспективное направление для создания доступных и адаптивных средств управления, особенно для пользователей с ограниченными возможностями.

Список литературы

1. Туровский, Я. А. Обработка сигнала электроэнцефалограммы на основе анализа частотных зависимостей и вейвлет-преобразования / Я. А. Туровский, С. Д. Кургалин, А. А. Вахтин // Радиоэлектроника. – 2012. – № 12. – С. 39-45.

2. Андреенко А. С. Обзор методов математической обработки электроэнцефалограмм / А. С. Андреенко // Известия ТРТУ. – 2000. – № 4(18). – С. 111-112.

3. Туровский, Я. А. Типологизация локальных максимумов в спектрах электроэнцефалограмм, полученных путем непрерывного вейвлет-преобразования / Я. А. Туровский, С. Д. Кургалин, А. Г. Семенов // Программные продукты и системы. – 2013. – № 4. – С. 57.

4. Бахарев, Б. В. Корреляционно-декрементный анализ энцефалограмм на основе моделирования биопотенциалов коры головного мозга / Б. В. Бахарев // Биофизика. – 2016. – Т. 61, № 4. – С. 793-798.

5. Туровский, Я. А. Моделирование процессов выделения частотных локальных минимумов в электроэнцефалограммах / Я. А. Туровский, С. Д. Кургалин, А. В. Максимов // Вестник Тамбовского государственного технического университета. – 2012. – Т. 18, № 4. – С. 827-833.

Разработка приложения для поиска документов и генерации ответов с расширенным поиском

А. А. Ушакова

Студент магистр

В. В. Гаршина

Доцент

Введение

Большие языковые модели показывают впечатляющие результаты в задачах обработки естественного языка. Однако они обладают рядом существенных ограничений: устаревание знаний после окончания обучения, невозможность учитывать закрытую корпоративную информацию, склонность к галлюцинациям при отсутствии достоверных источников.

Одним из подходов к решению этих проблем является генерация ответов, дополненная поиском. В данной работе рассматривается разработка приложения, обеспечивающего поиск релевантной информации и генерацию ответов на текстовые и голосовые запросы с использованием больших языковых моделей. Работа включает исследование существующих методов, реализацию системы, а также её оценку по различным критериям.

1. Обзор существующих подходов

Упрощённый метод с использованием большой языковой модели. Данный метод предполагает разбиение документа на логические части и использование большой языковой модели для определения наиболее релевантного фрагмента и генерации ответа. К его преимуществам относится простота реализации и высокая скорость работы. Этот способ может быть реализован даже без использования векторных баз данных и моделей векторизации, что делает его подходящим для небольших систем. Однако у подхода есть существенные недостатки: он плохо масштабируется при увеличении объема данных и неэффективен при работе с неструктурированными текстами.

Метод генерации с использованием извлечённой информации. В этом методе перед генерацией ответа осуществляется поиск релевантной информации во внешней векторной базе данных. Запрос

© Ушакова А. А., Гаршина В. В., 2025

преобразуется в векторное представление, на основе которого извлекаются фрагменты текста, и только затем формируется итоговый ответ. Преимущества метода заключаются в возможности работы с неструктурированными данными и масштабируемости на большие объёмы документов. Среди недостатков можно отметить неспособность обрабатывать сложные, многошаговые вопросы, необходимость настройки параметров поиска и увеличенную задержку при формировании ответа из-за обращений к базе данных.

Использование моделей с длинным контекстным окном. Данный подход использует модели, способные обрабатывать большие объемы текста без предварительного разбиения на фрагменты. Это особенно полезно при работе с неструктурированными документами. Преимуществами метода являются упрощённая архитектура системы, возможность работы с целыми документами и высокая гибкость. Среди недостатков - высокая стоимость таких моделей, требования к вычислительным ресурсам, а также сложность отслеживания, откуда именно была взята информация при генерации ответа.

2. Архитектура системы

Поскольку приватная информация обязывает использовать локальные модели, а запуск локальных больших языковых моделей требует существенных аппаратных ресурсов, было принято решение использовать базовый метод генерации с использованием извлечённой информации.

На рис. 1 представлена архитектура разрабатываемой системы. Аудиосигнал поступает на вход модели автоматического распознавания речи, в результате чего формируется текстовый вариант запроса.

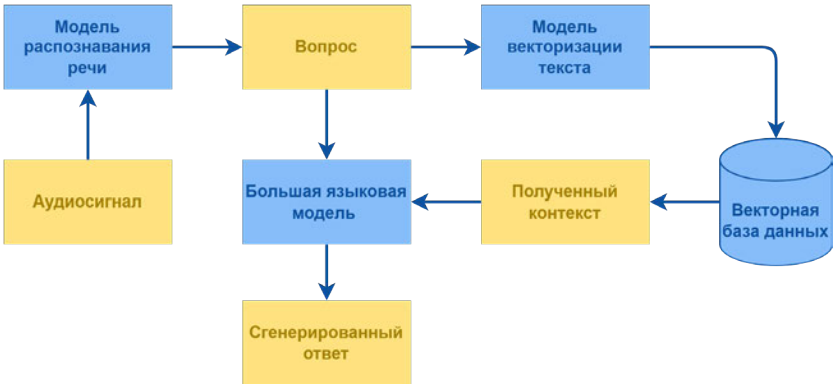


Рис. 1. Архитектура разрабатываемой системы

Альтернативно, запрос может быть изначально подан в текстовом виде. Далее текст запроса обрабатывается моделью векторизации с целью формирования его векторного представления. Полученный вектор используется для поиска в векторной базе данных, откуда извлекаются k наиболее релевантных фрагментов на основе косинусного расстояния между векторами запроса и документов. Извлечённые фрагменты совместно с исходным вопросом передаются в большую языковую модель, которая формирует финальный ответ.

3. Выбор средств реализации

- Язык программирования: Python
- Фреймворк: LangChain
- Векторная база данных: Chroma DB
- Большая языковая модель: Qwen2.5-7B-Instruct
- Модель распознавания речи: GigaAM-CTC

Выбор перечисленных средств был обусловлен их производительностью, доступностью исходного кода, активным сообществом разработчиков, а также возможностью использования в условиях ограничений на передачу конфиденциальных данных во внешние сети.

4. Оценка системы

Оценка разработанной системы включает два ключевых аспекта: точность извлечения релевантных фрагментов и качество сгенерированных ответов на основе этих фрагментов.

Эффективность модуля поиска оценивается по бинарному критерию. Если среди k извлечённых фрагментов есть оригинальный фрагмент, содержащий ответ на вопрос, результат считается успешным и получает оценку 1. В противном случае результат считается неуспешным и получает оценку 0.

Для оценки качества ответа применяется большая языковая модель - оценщик, которая сопоставляет вопрос, оригинальный ответ и сгенерированный текст. Оценка выставляется по шкале от 1 до 5, где 5 означает максимально точное, полное и связное соответствие исходному вопросу.

Данный метод позволяет автоматически и объективно измерить степень семантической точности модели, не прибегая к ручной проверке.

5. Эксперименты по настройке гиперпараметров

С целью повышения эффективности работы системы был проведён ряд экспериментов, направленных на выбор оптимальных параметров.

В табл. 1 представлены эксперименты с различным размером фрагментов. Оптимальным оказался размер 1000 символов - он обеспечивает баланс между полнотой контекста и семантической точностью. Далее при увеличении размера фрагмента наблюдается снижение релевантности, при уменьшении - потеря информативности.

Таблица 1

Результаты тестирования размера фрагментов

Размер фрагмента	Оценка поиска	Оценка качества ответа
100	0.63	2.99
300	0.64	3.0
500	0.65	3.02
700	0.60	2.86
800	0.56	3.0
900	0.62	3.12
1000	0.76	3.38
1100	0.64	3.18
1200	0.68	2.76

Также было исследовано влияние количества возвращаемых фрагментов. Результаты, представленные в табл. 2 демонстрируют улучшение показателей до значения $k = 21$, однако дальнейшее увеличение ограничено ресурсами памяти.

Таблица 2

Результаты тестирования количества фрагментов

Количество фрагментов	Оценка поиска	Оценка качества ответа
1	0.48	2.92
3	0.52	3.09
6	0.58	3.21
9	0.6	3.24
12	0.63	3.3
15	0.67	3.43
17	0.7	3.36
19	0.72	3.38
21	0.75	3.52

На рис. 2 показаны результаты тестирования моделей для векторизации текстов. Наиболее высокие результаты показала модель Multilingual-E5-large, продемонстрировавшая высокую способность к выделению семантически значимых фрагментов и устойчивую производительность на различных типах текстов.

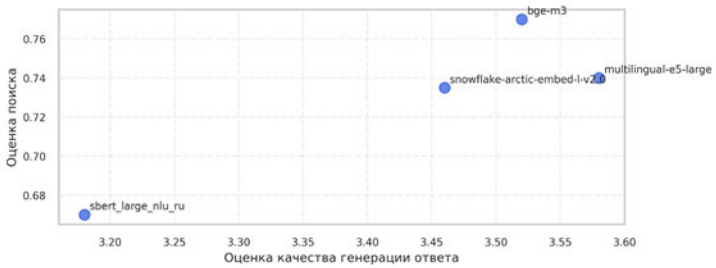


Рис. 2. Результаты тестирования моделей векторизации текста

6. Методы повышения качества поиска и релевантности фрагментов

Для повышения полноты поиска к семантическому подходу была добавлена компонентная поддержка лексического поиска. Используемый алгоритм BM25 позволяет находить фрагменты с точным совпадением ключевых токенов. Это особенно полезно в случаях, когда семантические модели не выявляют прямых соответствий из-за особенностей формулировки.

На рис. 3 представлен процесс гибридного поиска и ранжирования контекста. Перед генерацией ответа была внедрена модель ранжирования по категории.

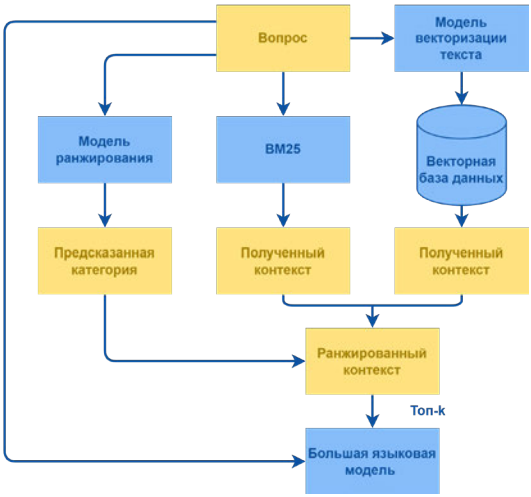


Рис. 3. Схема процесса гибридного поиска и ранжирования контекста

Сначала запрос классифицируется по тематике с использованием предварительно обученной модели логистической регрессии, затем извлечённые фрагменты ранжируются в соответствии с предсказанной категорией.

Также был проведён эксперимент по определению оптимального порогового значения уверенности классификатора. Оптимальным компромиссом между полнотой поиска и качеством генерации оказался порог 0.2, позволяющий отсечь наименее информативные контексты и одновременно сохранить важные сведения. Результаты данного эксперимента представлены на рис. 4, где продемонстрирована зависимость качества работы системы от выбранного порогового значения.

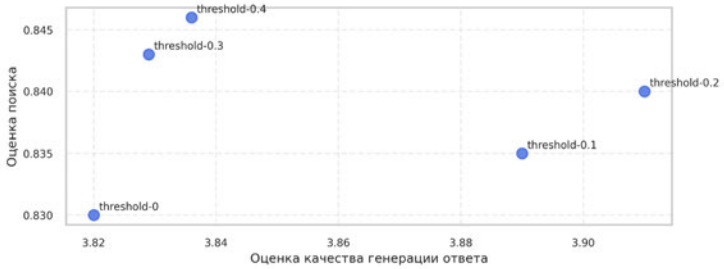


Рис. 4. Зависимость качества поиска и генерации от порогового значения уверенности классификатора

Заключение

В ходе работы был проведён обзор существующих подходов к генерации ответов, начиная с простых документных методов и заканчивая использованием векторных баз данных и моделей с длинным контекстным окном.

Разработано приложение, реализующее расширенный подход к поиску и генерации ответов с использованием локальных моделей. Проведена оценка качества системы, как на этапе поиска релевантных фрагментов, так и на этапе генерации итогового ответа.

Были проведены эксперименты, направленные на исследование влияния различных гиперпараметров, включая размер фрагментов, количество извлекаемых документов и порог уверенности классификации. В дополнение к базовому поиску реализован и обучен модуль ранжирования фрагментов, что позволило существенно повысить точность и релевантность генерируемых ответов.

Список литературы

1. Retrieval-Augmented Generation for Large Language Models: A Survey [Электронный ресурс]. – Режим доступа: https://arxiv.org/html/2312.10997v5

2. Searching for Best Practices in Retrieval-Augmented Generation [Электронный ресурс]. – Режим доступа: https://arxiv.org/html/2407.01219v1

3. GigaAM: класс открытых моделей для обработки звучащей речи [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/sberdevices/articles/805569/

4. Захарова, О. И. Семантический анализ и синтез текстовых данных / О. И. Захарова // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2023. – № 4. – С. 182-208. – DOI 10.17308/sait/1995-5499/2023/4/182-208. – EDN OUJELO.

5. Kamphuis, C. Which BM25 do you mean? A large-scale reproducibility study of scoring variants / C. Kamphuis, A. P. de Vries, L. Boytsov, J. Lin // European Conference on Information Retrieval. – 2020. – P. 28–34.

Сравнительный анализ архитектурных подходов к реализации CRS систем для гостиничного бизнеса

А. Н. Федоров

Студент магистр

П. С. Лысачев

Старший преподаватель

Введение

Современные гостиничные предприятия все чаще используют централизованные системы бронирования (CRS) для консолидации данных о номерах, тарифах и статусах резервирования в единой платформе. CRS интегрируются с Property Management System (PMS) и каналами дистрибуции (OTA, GDS), что позволяет оптимизировать доход, предотвращать овербукинг и оперативно реагировать на колебания спроса.

Традиционно CRS разворачиваются как монолитные приложения, в которых пользовательский интерфейс, бизнес-логика и база данных масштабируются и обновляются целиком, что при росте нагрузки приводит к высокой стоимости вертикального апгрейда, долгим релизам и единой точке отказа. Микросервисная архитектура разбивает CRS на независимые сервисы (inventory, pricing, reservations и др.), упрощая горизонтальное масштабирование и ускоряя цикл разработки, но порождая сетевые задержки и усложняя DevOps-процессы. Облачные микросервисы дополняют преимущества микросервисных решений автоматическим масштабированием, управляемыми сервисами и моделью оплаты «pay-as-you-go».

Цель работы — системно сравнить три архитектурных подхода (монолит, микросервисы и облачные микросервисы) в разработке CRS, чтобы определить как различия в архитектурных подходах влияют на ключевые характеристики CRS: масштабируемость, производительность, надежность, сложность DevOps-процессов и экономическую эффективность для гостиничного бизнеса.

1. Определение и ключевые функции CRS

CRS (Central Reservation System) — это централизованная система управления бронированиями в гостиничном бизнесе, объединяющая все

© Федоров А. Н., Лысачев П. С., 2025

каналы продаж и внутренние сервисы отеля в единой платформе. Основная задача CRS — обеспечить сквозную обработку данных по доступности номеров, тарифам и статусам резервирования, а также автоматизировать взаимодействие со сторонними каналами дистрибуции и внутренними ИТ-системами отеля [1].

Ключевые функции CRS:

1. Управление инвентарем: актуальные данные по номерам, автоматическая корректировка остатков, квотирование по каналам.
2. Динамическое ценообразование: прогноз спроса, настройка тарифов, автоматическое обновление цен с учетом сезона и конкурентов.
3. Управление каналами распространения: синхронизация с OTA и GDS, предотвращение овербукинга, централизованное управление каналами продаж.
4. Аналитика и отчетность: ключевые показатели (RevPAR — доход с номера; ADR — средняя цена за проданный номер; процент занятых номеров), дашборды, экспорт в BI-системы (Business Intelligence — аналитика и визуализация), отчеты по доходам, каналам продаж и эффективности тарифов.
5. Интеграция: связь с PMS, API для внешних решений, поддержка платежных систем, CRM и программ лояльности.

2. Архитектурные стили ПО

Ранние CRS чаще всего реализовывались по монолитной архитектуре, где пользовательский интерфейс, бизнес-логика и база данных находились в одном приложении. Такой подход действительно упрощал разработку и отладку: все компоненты масштабировались и обновлялись одновременно, а для эксплуатации требовалось минимальное количество DevOps-инструментов.

Микросервисная архитектура, напротив, разбивает CRS на автономные сервисы с собственным жизненным циклом и базой данных. Это обеспечивает гибкое горизонтальное масштабирование, ускоряет доставку изменений и повышает отказоустойчивость за счет изоляции сбоев. Однако межсервисные вызовы и сетевые задержки могут снижать общую производительность, а сложная оркестрация, CI/CD-конвейеры и системы мониторинга требуют развитых DevOps-процессов и увеличивают операционные затраты [2].

Облачные микросервисы объединяют преимущества микросервисов с возможностями автоматического масштабирования (Kubernetes, Serverless/FaaS), управляемыми облачными сервисами и моделью «pay-as-you-go» [3]. Это снижает капитальные затраты и позволяет быстро реагировать на пиковые нагрузки, но усиливает

зависимость от провайдера, усложняет контроль затрат и предъявляет более высокие требования к безопасности и наблюдаемости.

3. Архитектурная реализация CRS

Рассмотрим три варианта реализации CRS — монолитный, микросервисный и облачный микросервисный — с последующей оценкой по ключевым параметрам:

Монолитный прототип оформлен как единое FastAPI-приложение со всеми маршрутами (users, inventory, reservations, pricing, payments) и одной схемой базы данных PostgreSQL. Миграции схемы управляются Alembic, REST-эндпоинты обрабатываются в одном Uvicorn-процессе, а интеграция с Kafka реализована через aiokafka. Фронтенд React-SPA собирается в статические файлы и обслуживается через Nginx. Деплой выполняется через Dockerfile, CI/CD-конвейер включает сборку, запуск тестов, сбор образа, публикацию и развертывание.

Микросервисный прототип состоит из FastAPI API Gateway с JWT-аутентификацией и маршрутизацией, а также пяти ключевых микросервисов:

- User Service (регистрация, аутентификация, управление ролями)
- Inventory Service (CRUD номеров, проверка доступности)
- Reservation Service (создание/отмена бронирования)
- Pricing Service (расчет и динамическое обновление тарифов)
- Payment Service (интеграция с платежным шлюзом, обработка транзакций)

Каждый сервис имеет собственную PostgreSQL-БД, модели SQLAlchemy и миграции Alembic. Синхронное взаимодействие с клиентом и между сервисами ведется по HTTP/REST через API Gateway, а для событий (booking.created, payment.completed) используется Kafka. Оркестрация — Docker + Kubernetes с Helm-чартами и ConfigMap/Secret, балансировка нагрузки — Nginx.

Облачный микросервисный прототип реализован «инфраструктурой как код» через Terraform: создаются VPC, EKS, IAM-роли, MSK (Kafka) и RDS (PostgreSQL). Сервисы упакованы в Docker-образы, CI/CD публикует их в ECR и разворачивает в EKS-Pods с HPA. Для лёгких фоновых задач — AWS Lambda. Внешние вызовы идут через API Gateway с авторизацией Cognito. Событийный обмен идёт по MSK или встроенному Kafka-кластеру. Логи собираются в CloudWatch, метрики — в Prometheus и Grafana. Безопасность — через VPC-сети, Security Groups и шифрование RDS; доступ контролируется IAM-ролями, привязанными к Kubernetes-аккаунтам (IRSA).

4. Сравнительный анализ разработанных CRS

В рамках работы сравнение архитектурных подходов к CRS проводится по пяти основным параметрам:

1. Масштабируемость – способность системы обрабатывать растущий объем запросов за счет вертикального и горизонтального увеличения ресурсов, включая механизмы автоматического масштабирования (auto-scaling).

2. Производительность – среднее время отклика (Average Response Time, ART) и пропускная способность (Requests Per Second, RPS) при пиковых нагрузках.

3. Надежность и устойчивость к сбоям – изоляция и восстановление сервисов (self-healing, rollback) и время восстановления после отказа (MTTR).

4. DevOps-процессы – сложность CI/CD-конвейера и набор инструментов наблюдения (логирование, трассировка, метрики).

5. Экономическая эффективность (TCO) – капитальные (CapEx) и операционные (OpEx) затраты, модель оплаты (fixed-cost vs pay-as-you-go) и оптимизация использования ресурсов.

Рассмотрим результаты анализа:

График на рис. 1 отражает, как три архитектурных подхода реагируют на рост нагрузки в CRS с точки зрения времени отклика. Монолит демонстрирует наименьшие задержки при низкой нагрузке за счет отсутствия сетевых взаимодействий, однако его производительность резко деградирует после 1000 RPS: кривая отклика стремительно возрастает, указывая на исчерпание ресурсов. Микросервисная архитектура ведет себя устойчивее — с ростом нагрузки задержки растут более плавно, однако изначально они выше, поскольку каждый пользовательский запрос проходит через несколько сервисов, между которыми происходят сетевые вызовы. Облачно-микросервисный вариант демонстрирует наилучшее поведение: кривая отклика самая стабильная и пологая, даже при нагрузке до 2000 RPS, благодаря автоматическому масштабированию и балансировке трафика.

График на рис. 2 иллюстрирует различия в масштабируемости трех архитектурных подходов к построению CRS при увеличении количества экземпляров сервисов. Монолитная система демонстрирует практически полное отсутствие масштабируемости: вне зависимости от числа экземпляров, производительность остается на стабильном уровне, ограниченном характеристиками единого приложения.

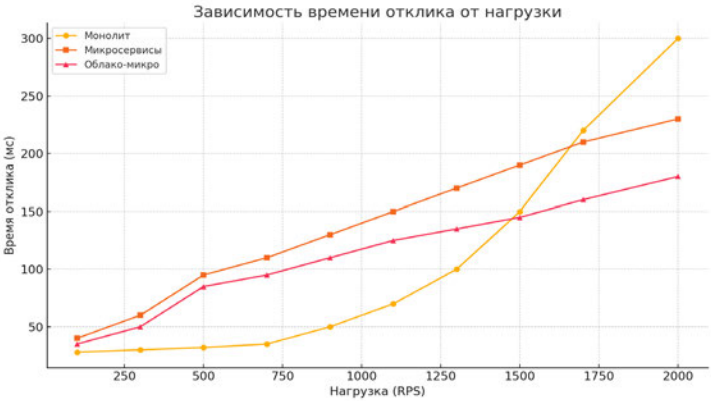


Рис. 1. Зависимость времени отклика от нагрузки

Микросервисная архитектура показывает почти линейный рост производительности — каждый добавленный экземпляр увеличивает общую способность системы обрабатывать запросы, поскольку нагрузка может эффективно распределяться между независимыми сервисами. Облачно-микросервисная архитектура демонстрирует наилучшие результаты: за счет автоматического масштабирования, высокой степени параллелизма и управляемых ресурсов облака производительность значительно превышает значения, достижимые в традиционных архитектурах.

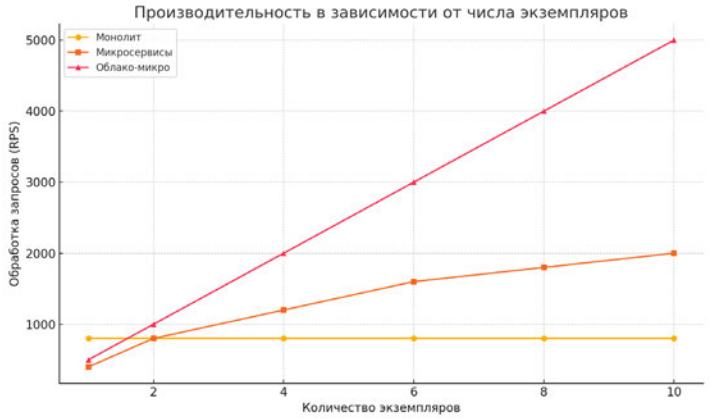


Рис. 2. Производительность в зависимости от числа экземпляров

Диаграмма на рис. 3 демонстрирует различия в распределении задержек отклика между архитектурами CRS по двум ключевым статистическим метрикам — медиане (P50) и 95-му перцентилю (P95).

Монолит обеспечивает минимальные и стабильные задержки до предела пропускной способности. В микросервисах P50 выше, а P95 значительно растёт из-за сетевых взаимодействий и «длинного хвоста». Облачно-микросервисная модель занимает среднее положение: показатели хуже монолита, но лучше микросервисов, а автоскейлинг и балансировка сглаживают пики и уменьшают разброс.

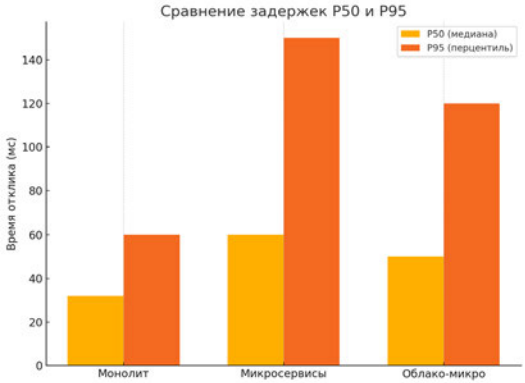


Рис. 3. Сравнение задержек P50 и P95

Для оценки надежности и эксплуатационной устойчивости системы централизованного бронирования (CRS) использован ряд технических метрик, отражающих поведение системы под нагрузкой и в условиях сбоев:

1. ART при 500 RPS (мс) — среднее время отклика (Average Response Time) при типовой нагрузке 500 запросов в секунду. Отражает производительность системы в нормальном режиме эксплуатации.
2. 95 %-й перцентиль ART (мс) — значение, в которое укладываются 95 % откликов. Позволяет оценить стабильность системы и выявить задержки в наихудших сценариях (например, при временных пиках нагрузки).
3. Максимальный устойчивый RPS — наибольшее количество запросов в секунду, которое система способна обрабатывать без заметной деградации производительности (роста задержек или потерь).
4. MTBF (Mean Time Between Failures) — среднее время между отказами. Чем выше значение, тем реже происходят сбои.

5. MTTD (Mean Time To Detect) — среднее время обнаружения сбоя системой мониторинга или администратором. Характеризует оперативность реакции на инциденты.

6. MTTR (Mean Time To Repair) — среднее время устранения сбоя и восстановления нормальной работы. Чем меньше MTTR, тем выше доступность сервиса.

7. Failover — время переключения на резервный компонент или узел в случае отказа текущего. Важный показатель отказоустойчивости архитектуры.

8. SLA (Service-Level Agreement) — целевой уровень доступности системы, выраженный в процентах времени за год, в течение которого сервис доступен пользователям.

9. Blast radius — зона воздействия при отказе одного компонента: доля системы, функционально затрагиваемая сбоем.

10. Self-healing — наличие механизмов автоматического восстановления (перезапуск, переключение, изоляция) без участия оператора.

Представленная ниже таблица отражает сравнительный профиль надежности, масштабируемости и устойчивости различных архитектур CRS при эксплуатационных нагрузках. Монолитная архитектура демонстрирует наименьшее среднее время отклика при умеренной нагрузке, но ее масштабируемость ограничена, а отказ одного узла приводит к полной недоступности системы. В результате среднее время между отказами (MTBF) ниже, а время на обнаружение и устранение инцидентов (MTTD, MTTR) значительно выше, чем в других подходах.

Таблица

Результаты анализа разработанных CRS систем

Метрика	**Монолит**	**Микросервисы**	**Облач. микросервисы**
ART при 500 RPS, мс	32 ± 5	95 ± 15	≈ 85 ± 12
95 %-й перцентиль ART, мс	60	150	120
Максимальный устойчивый RPS	~ 800	~ 1 200 (≈ +50 %)	~ 2 000 (≈ +150 %)
MTBF	40 ч	60 ч	90 ч
MTTD	10 мин	2 мин	20 с

Облачно-микросервисный подход достигает наилучших результатов: благодаря автоматическому масштабированию и встроенным механизмам самовосстановления он обеспечивает минимальное время простоя, быструю реакцию на сбои и наивысшее значение SLA (99,99 %). Отказоустойчивость обеспечивается распределением нагрузки между множеством экземпляров и зон доступности, что резко снижает радиус поражения при сбое (blast radius) и сводит к минимуму негативное влияние на пользователей.

Таким образом, с усложнением архитектуры система становится значительно более надежной, отказоустойчивой и предсказуемой при высокой нагрузке, что критически важно для гостиничного бизнеса.

Касательно сложности реализации диаграмма на рис. 4 показывает, как с переходом от монолита к микросервисам и облачно-микросервисной архитектуре растёт DevOps-сложность: от простых централизованных CI/CD, мониторинга, деплоя, тестирования и безопасности в монолите до усложнённых пайплайнов и инструментов в микросервисах и максимальных требований к автоматизации безопасности, управлению конфигурациями и оркестрации обновлений в облачно-микросервисах. Этот рост сложности отражает ключевой компромисс: чем выше гибкость и отказоустойчивость, тем дороже поддержка [4].

Рис. 4. DevOps-сложность по архитектурам

На рис. 5 представлен график демонстрирующий изменение структуры совокупной стоимости владения (TCO) при переходе от монолита к микросервисам и далее к облачно-микросервисной

архитектуре. В случае монолита основную долю расходов составляет капитальная часть (CapEx) — это затраты на собственные сервера, лицензии и инфраструктуру, при этом операционные издержки (OpEx) и расходы на поддержку остаются относительно низкими. Микросервисная архитектура перераспределяет затраты: снижается CapEx за счет перехода на облачные или виртуализированные решения, но существенно возрастают постоянные расходы на ресурсы (OpEx) и команду DevOps, необходимую для поддержки распределенной системы. В облачно-микросервисной модели CapEx почти отсутствует — вся нагрузка переносится на оплату потребляемых облачных ресурсов и обеспечение непрерывной эксплуатации, включая автоматизацию, мониторинг и обеспечение отказоустойчивости [5].

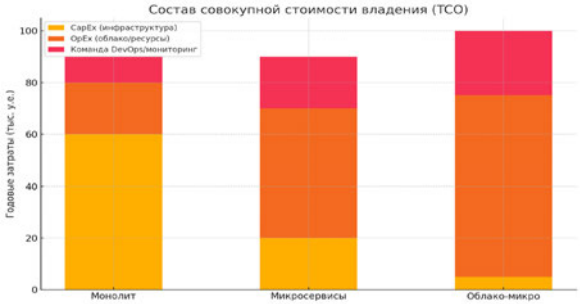


Рис. 5. Состав совокупной стоимости владения (TCO)

Таким образом, экономический профиль архитектуры сдвигается от фиксированных затрат к модели потребления, что особенно выгодно при переменной нагрузке, но требует более тщательного контроля текущих расходов.

Заключение

Сравнение трех архитектурных подходов к реализации CRS показало, что архитектура напрямую влияет на масштабируемость, надежность, производительность и затраты на эксплуатацию.

Монолит прост в разработке и сопровождении, показывает лучшие задержки при низкой нагрузке, но плохо масштабируется и уязвим к сбоям. Микросервисы обеспечивают гибкое масштабирование, локализацию сбоев и повышенный SLA, но требуют более сложной DevOps-инфраструктуры. Облачно-микросервисная архитектура демонстрирует наилучшую устойчивость и адаптивность при пиковых нагрузках, а также минимальное время простоя, однако отличается наибольшей инженерной и операционной сложностью.

Выбор архитектуры должен учитывать размеры бизнеса, предполагаемую нагрузку и зрелость инфраструктуры: от монолита для небольших объектов до облачно-микросервисной модели для крупных и динамичных гостиничных сетей.

Список литературы

1. Central Reservation System for Hotels: CRS Functionality Explained [Электронный ресурс]: AltexSoft Tech Blog. – Режим доступа: https://www.altexsoft.com/blog/central-reservation-system-hotel/

2. Alshuqayran, N. A Systematic Mapping Study in Microservice Architecture / N. Alshuqayran, N. Ali, R. Evans // 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (Macau, 4-6 Nov. 2016). – Macau, China, 2016. – P. 44-51. – Режим доступа: https://core.ac.uk/download/pdf/188256155.pdf

3. Serverless vs. Kubernetes when deploying microservices [Электронный ресурс]: ThoughtWorks Insights. – Режим доступа: https://www.thoughtworks.com/en-us/insights/blog/microservices/serverless-kubernetes-microservices

4. How to Achieve High Availability and Fault Tolerance in DevOps [Электронный ресурс]: Jeevi Academy. – Режим доступа: https://www.jeeviacademy.com/how-to-achieve-high-availability-and-fault-tolerance-in-devops/

5. Comparative Analysis of On-Premises and Cloud Hosting Solutions [Электронный ресурс]: DIVA Portal. – Режим доступа: https://www.diva-portal.org/smash/get/diva2:1887940/FULLTEXT01.pdf

Разработка web-сервиса для автоматизации работы отдела интеллектуальной собственности ФГБОУ ВО ВГУ

П. О. Филонов

Студент магистр

Е. А. Копытина

Доцент

Введение

Стремительное развитие информационных технологий дало мощный толчок к автоматизации систем. Любой процесс, поддающийся какому-либо алгоритму, способен стать частью ИТ среды. Последние десятилетия основным трендом развития любой деятельности является её перевод в цифровой вид. Таким образом, автоматизация систем, безусловно, актуальная тема, требующая владение сложными навыками и современными инструментами, которые позволяют вести разработку с обеспечением защиты информации и отсутствием ошибок для корректной работы [1-6]. От чётким соответствиям требований зависит удобство и полезность дальнейшей работы системы. Следовательно, разработка информационной системы требует ещё и глубокое понимание предметной области.

Одной из важных сфер является система образования. Применение знаний об информационных технологиях и навыков по их разработке может способствовать автоматизации внутренних процессов университета. Это даёт возможность уменьшить нагрузку на сотрудников по рутинным делам, дать возможность переключиться им на более важные задачи, а также ускорить общее время выполнения какого-либо процесса.

В рамках данной статьи, рассматривается разработка информационной системы, а именно, веб-сервиса, для автоматизации работы отдела интеллектуальной собственности ФГБОУ ВО «ВГУ». Сотрудники отдела занимаются заявками и патентами различных программ, баз данных и другими видами интеллектуальной собственности, которые разрабатываются представителями университета. При всём этом, работниками отдела необходимо собирать большое количество данных. При отсутствии специализированной

© Филонов П. О., Копытина Е. А., 2025

программы, используются либо бумажные носители, либо неподходящие для этой задачи цифровые инструменты, например, excel. При этом, совершенно неудобным являются возможности агрегирования реестров, поиск нужных записей, фильтрация и многое другое. Отсюда следует вывод о необходимости разработки специализированного для данной потребности сервиса.

1. Постановка задачи

В рамках данной статьи, разрабатываемое приложение должно иметь следующие функциональные возможности:

- Создание, редактирование, удаление и просмотр элементов реестров;
- Изменение статусов элементов реестров;
- Разделение функциональных возможностей и доступов по ролевой модели.

Реестры приложения:

- Заявок на изобретения;
- Заявок на полезные модели;
- Программ ЭВМ;
- Баз данных;
- Топологий интегральных микросхем;
- Договоров;
- Сведений, составляющих коммерческую тайну.

2. Средства реализации программного продукта

Проект реализуется с помощью следующих технологий:

- Язык программирования Java 11. Строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems (в последующем приобретённый компанией Oracle). Популярные области применения языка: разработка enterprise приложений, веб-разработка, мобильная разработка.
- Фреймворк tesler 3.0.4. Open-source фреймворк для разработки enterprise-приложений, работающий на принципе бизнес-компонент и содержащий абстрактную реализацию операций CRUD (create, read, update, delete) и некоторых дополнений для работы с бизнес-сущностями [7].
- Язык программирования TypeScript. Статически типизированный язык программирования, использующийся для создания интерактивных и динамических web-сайтов. С его помощью сайты делают интерактивными: добавляют всплывающие окна, анимацию и формы для отправки

информации. Разработан на базе языка программирования JavaScript и расширяет его возможности. Они являются обратно совместимыми, а TypeScript компилируется в JavaScript.

– Фреймворк React. JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. React помогает создать функциональные интерактивные web-интерфейсы, быстро и удобно реализовать отдельные компоненты и страницы целиком.
– Платформа Docker. Программная платформа для разработки, доставки и запуска контейнерных приложений. Позволяет создавать контейнеры, автоматизировать их запуск и развёртывание, управляет жизненным циклом. Приложение, упакованное в контейнер, изолируется от операционной системы и других приложений. Следовательно, его возможно запускать единообразно и меньше заботиться о системных зависимостях.

Для реализации проекта используется следующее программное обеспечение:

– IntelliJ IDEA 2023.3.2 (Ultimate Edition). Интегрированная среда разработки (IDE), используемая для программирования на языке Java. Она также поддерживает множество других языков программирования, таких как Kotlin, Groovy, Scala, JavaScript, TypeScript и другие. IntelliJ IDEA предлагает широкий спектр инструментов для повышения производительности разработчиков, включая рефакторинг, отладку, интеграцию с системами контроля версий, такими как Git, и поддержку различных фреймворков и технологий, включая Spring, Hibernate и Java EE. Ultimate Edition предоставляет дополнительные функции, такие как поддержка корпоративных технологий и инструментов для работы с базами данных, что делает её мощным инструментом для профессиональных разработчиков.

3. Реализация логики программного продукта

На рисунке 1 отображена архитектура разрабатываемого веб-приложения, программные модули.

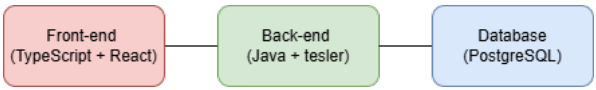


Рис. 1. Архитектура веб-приложения

– UI сторона реализована на front-end части с помощью языка программирования TypeScript и фреймворка React.
– Основная бизнес-логика реализована на back-end стороне, в так называемом «сервисном» слое, с помощью языка программирования Java и фреймворке tesler.
– База данных приложения – реляционная СУБД PostgreSQL. Используется для хранения записей созданных бизнес-сущностей.

Бизнес-логика разделена на бизнес-компоненты, каждая из которых определяет действие над конкретной бизнес-сущностью (рисунок 2):

```
public enum TeslerRestController implements EnumBcIdentifier {

  userRead(UserReadService.class),
  userEdit(UserEditService.class),

  departmentRead(DepartmentReadService.class),
  departmentEdit(DepartmentEditService.class),

  innovationRead(InnovationReadService.class),
  innovationEdit(InnovationEditService.class),

  usefulModelRead(UsefulModelReadService.class),
  usefulModelEdit(UsefulModelEditService.class),

  agreementRead(AgreementReadService.class),
  agreementEdit(AgreementEditService.class),

  programRead(ProgramReadService.class),
  programEdit(ProgramEditService.class),

  databaseRead(DatabaseReadService.class),
  databaseEdit(DatabaseEditService.class),

  integratedCircuitRead(IntegratedCircuitReadService.class),
  integratedCircuitEdit(IntegratedCircuitEditService.class),

  tradeSecretRead(TradeSecretReadService.class),
  tradeSecretEdit(TradeSecretEditService.class),
```

Рис. 2. Список бизнес-компонентов

В контроллере бизнес-компонент (БК), структуре данных – енам, производится соответствие (маппинг) между наименованием БК и сервисом, переопределяющий характерные для уровня бизнес-логики методы по созданию, обновлению, фильтрации записей, а также списку доступных действий в рамках страниц, реализованных данной БК.

Построение страниц приложения происходит с помощью следующих программных компонент:

- Screen – верхний уровень, разрез раздела меню, в рамках которого обобщаются страницы глобального типа приложения. Содержит в себе список view и ролей, которым доступен данный screen (рисунок 3).
- View – средний уровень, разрез одной страницы, например: страница с таблицей записей, страница создания/редактирования записи, информационная страница записи и др. Содержит в себе список widget и ролей, которым доступен данная view.
- Widget – низкий уровень, разрез части страницы, например: заголовок, список редактируемых полей и др. Является более гибким компонентом, позволяющим «собирать» страницу из тех полей, которые содержат widget`ы.

Рис. 3. Screen раздела реестров

4. Реализация интерфейса программного продукта

На рисунке 4 отображено меню приложения под уровнем доступа администратора. Два раздела «Администрирование» и «Реестры» являются корневыми и разделяют логически разные подразделы. В зависимости от уровня доступа роли, разделы могут быть скрыты.

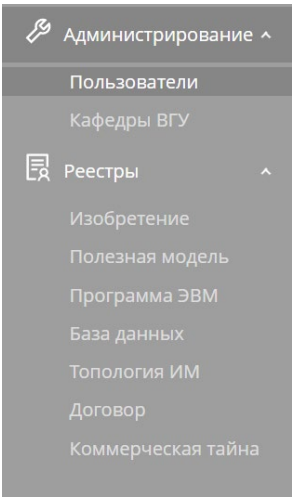


Рис. 4. Полное меню приложения

На рисунке 5 представлена часть процесса создания/редактирования элемента реестра базы данных. Сам процесс состоит из шагов, разделённых по страницам. Поля ввода представлены несколькими видами: ввод текста, выбор даты из календаря, выбор элементов из выпадающего списка и др.

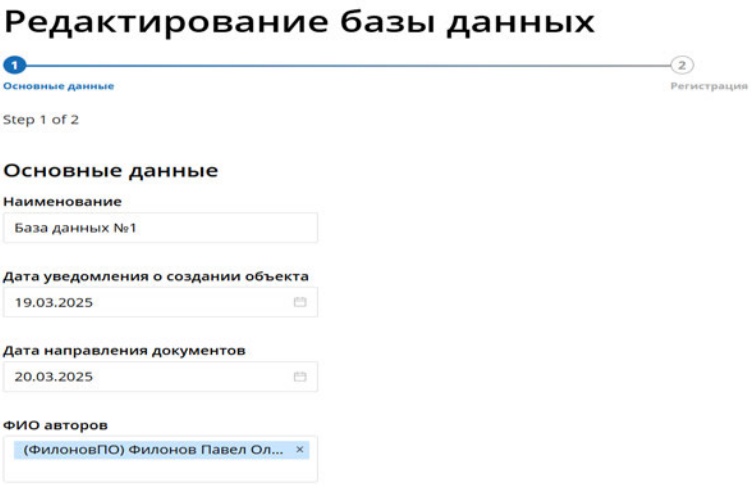


Рис. 5. Создание/редактирование элемента реестра базы данных

Вывод списка записей одного реестра реализован в табличном виде с указанием его ключевых параметров (рисунок 6). Кнопки с добавлением новой записи и открытием раздела фильтрации расположены сверху от таблицы. Первая колонка таблицы является гиперссылкой для перевода на информационную страницу, а справа от записи расположен открывающийся список доступных действий с записью.

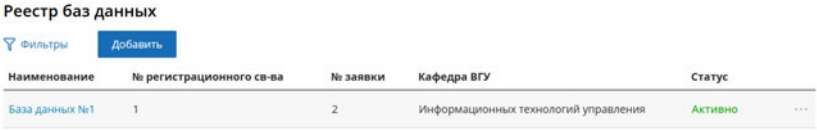


Рис. 6. Таблица реестра базы данных

Статусная модель состоит из трёх статусов: черновик, активно, неактивно. Черновик – временно сохранённая запись на стадии создания, активная запись является актуальной, а неактивная – отменённой. Из активной в неактивную и наоборот, возможен перевод по кнопкам «Деактивировать» (рисунок 7) и «Активировать» соответственно.

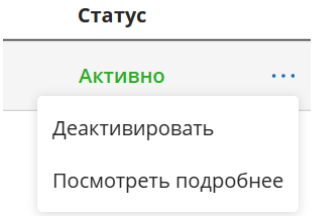


Рис. 7. Функциональные возможности для активной записи реестра

В системе действует ролевая модель, состоящая из трёх ролей: администратор, сотрудник и автор. В этом же порядке, их роли определённым способом урезаются для того, чтобы на разных уровнях поддерживать функциональные возможности и конфиденциальность. Наименьшая по доступным действиям и данным роль – «Автор», предназначена для авторов интеллектуальной собственности, не имеет доступа к каким-либо кнопкам, кроме дополнительного просмотра своих записей. Пример урезания доступа представлен на рисунке 8. На

нём отображено меню с разделами, отвечающими только за реестры заявок на интеллектуальную собственность.

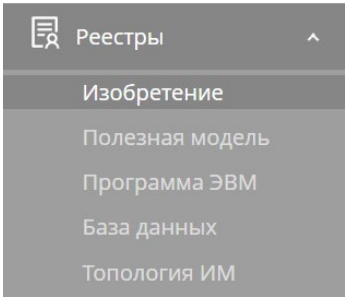


Рис. 8. Меню приложения для роли «Автор»

Заключение

В данной статье рассмотрена разработка веб-приложения для автоматизации работы отдела интеллектуальной собственности ФГБОУ ВО «ВГУ». Представлено предложенное решение по управлению реестрами, включающее собственные ролевую и статусную модели. Приложение отвечает поставленным требованиям и готово к использованию.

Список литературы

1. Васянкин, А. А. Модификация понятия «электронный документ» в системе электронного документооборота / А. А. Васянкин, Э. В. Кузьмина // Кайгородовские чтения : Материалы региональной научно-практической конференции, Краснодар, 29 апреля 2010 года / Краснодарский государственный университет культуры и исскуств. – Краснодар: Краснодарский государственный университет культуры и искусств, 2010. – С. 75-79.

2. Свидетельство о государственной регистрации программы для ЭВМ № 2023661037 Российская Федерация. Приложение для автоматизации документооборота дополнительного образования : № 2023618813 : заявл. 03.05.2023 : опубл. 25.05.2023 / А. С. Голышева, Е. А. Копытина ; заявитель Федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный технический университет». – EDN NBUVMM.

3. Шорохова, Т. А. Разработка проекта «List_of_documents» для реализации программ дополнительного образования ФГБОУ ВО «Воронежского государственного университета» / Т. А. Шорохова, Е. А. Копытина // Труды молодых учёных факультета компьютерных наук

ВГУ : Сборник статей. – Воронеж : Воронежский государственный университет, 2023. – С. 284-289. – EDN MMGXTK.

4. Претков, А. И. Разработка программного комплекса для автоматизации формирования документов основных профессиональных образовательных программ высшего образования факультета компьютерных наук Воронежского государственного университета / А. И. Претков, Е. А. Копытина // Информатика: проблемы, методы, технологии : материалы XXIII Международной научно-практической конференции им. Э.К. Алгазинова, Воронеж, 15–17 февраля 2023 года. – Воронеж: Воронежский государственный университет, 2023. – С. 907-912. – EDN ENWODO.

5. Претков, А. И. Автоматизации формирования документов основных профессиональных образовательных программ высшего образования факультета компьютерных наук Воронежского государственного университета / А. И. Претков, Е. А. Копытина // Труды молодых учёных факультета компьютерных наук ВГУ : Сборник статей. – Воронеж : Воронежский государственный университет, 2023. – С. 94-107. – EDN HKXMIS.

6. Проектная деятельность в дополнительном образовании на примере практик российских ВУЗов / Е. А. Копытина, Л. А. Гладкова, М. Е. Семенов [и др.] // Информатика: проблемы, методы, технологии : Материалы XXII Международной научно-практической конференции им. Э.К. Алгазинова, Воронеж, 10–12 февраля 2022 года / Под редакцией Д.Н. Борисова. – Воронеж: Общество с ограниченной ответственностью "Вэлборн", 2022. – С. 1342-1355. – EDN ONMOJZ.

7. Филонов, П. О. Расширение фреймворка «теслер» для разработки web-сервиса для автоматизации работы отдела интеллектуальной собственности ФГБОУ ВО «ВГУ» / П. О. Филонов, Е. А. Копытина // Информатика: проблемы, методы, технологии (IPMT-2025), Воронеж, 12–13 февраля 2025 года. – Воронеж: Воронежский государственный университет, 2025. – С. 907-912. – EDN ENWODO.

Реализация системы машинного зрения для выявления дефектов при производстве печатных плат

А. В. Фуфаева

Студент магистр

А. В. Максимов

Ст. преподаватель

Введение

В процессе производства и сборки печатных плат осуществляется ряд испытательных процедур. Эти процедуры направлены на оценку качества и долговечности печатной платы, а также на подтверждение того, что конструкция успешно пройдет этап сборки без недостатков.

В ряде компаний проверка печатных плат проходит вручную, а в условиях массового производства ручная проверка на микроскопе - очень трудоемкий процесс. При увеличении производства печатных плат трудоемкость ручной проверки возрастает пропорционально количеству производимых плат. Введение автоматизированной проверки позволит снизить трудоемкость процесса проверки.

Одной из ключевых задач систем машинного зрения является задача поиска и обнаружения дефектов на разного рода поверхностях или устройствах. Поэтому использование машинного зрения для выявления дефектов печатных плат позволит частично отказаться от ручной проверки.

1. Подготовка фотоматериалов

Первым этапом реализации системы машинного зрения является подготовка фотоматериалов и их разметка для дальнейшего обучения модели. Этот процесс включает в себя идентификацию и маркировку объектов на изображениях. Разметку можно выполнять вручную или с использованием автоматизированных инструментов.

В ходе работы было выявлено, что разметка простыми фигурами (прямоугольными областями) не приводила к эффективным результатам в данной задаче. Поэтому в качестве масок использовалась разметка по возможности максимально точно повторяющая контур выбранного элемента.

© Фуфаева А. В., Максимов А. В., 2025

Также было замечено улучшение результатов работы модели после редактирования исходных изображений. Изначально на фотоматериалах по краям присутствовали части изображений других печатных плат. Было принято решение обрезать изображения по размерам печатной платы.

На рис. 1 представлен пример изображения с неудачной разметкой. На нем элементы выделены прямоугольными областями, а само изображение не обрезано.

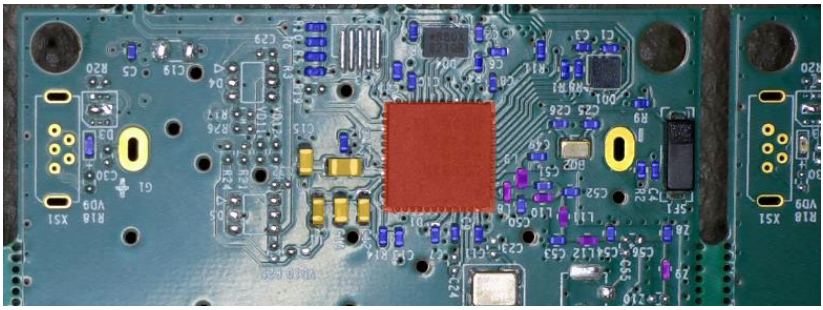


Рис. 1. Изображение с неудачной разметкой элементов

На рис. 2 представлен пример изображения с измененной разметкой. Контуры выделенных элементов совпадают с разметкой, изображение обрезано по размерам печатной платы.

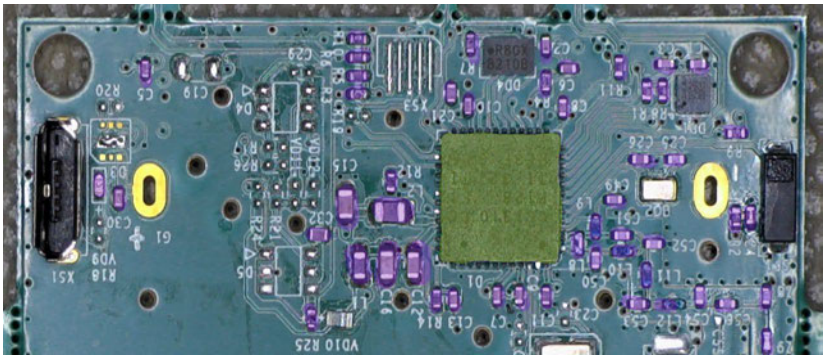


Рис. 2. Изображение с измененной разметкой

2. Выбор архитектуры модели и ее реализация

U-Net — это широко используемая архитектура глубокого обучения. Основной целью этой архитектуры было решение проблемы

ограниченных аннотированных данных в медицинской сфере. Эта сеть была разработана для эффективного использования меньшего объема данных при сохранении скорости и точности.

Архитектура U-Net уникальна тем, что состоит из сжимающегося пути и расширяющего пути. Сжимающийся путь содержит слои кодировщика, которые захватывают контекстную информацию и уменьшают пространственное разрешение ввода, в то время как расширяющий путь содержит слои декодера, которые декодируют закодированные данные и используют информацию из сжимающегося пути через пропускные соединения для генерации карты сегментации.

На рис. 3 представлено схематическое изображение архитектуры реализованной модели. Количество слоев и необходимые параметры были подобраны экспериментально.

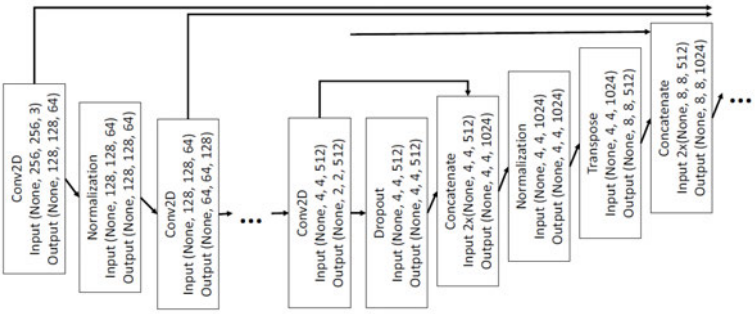


Рис. 3. Архитектура реализованной модели

На рис. 4 представлен пример работы модели.

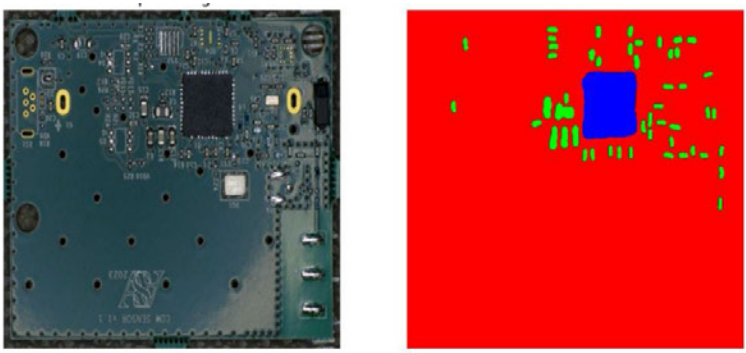


Рис. 4. Пример работы реализованной модели

3. Обучение модели распознаванию дефектов.

На данном этапе реализации системы ряд фотоматериалов был дополнен изображениями печатных плат, содержащих дефекты. На рис. 5 представлен пример изображения печатной платы с дефектом в виде смещения некоторых элементов.

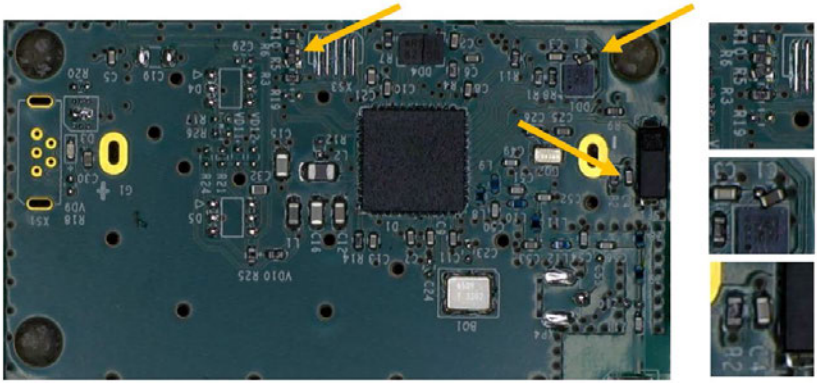


Рис. 5. Изображение печатной платы, содержащей дефекты

Для новых изображений также необходимо провести разметку. Выделяются элементы с дефектами, выделение происходит аналогично по контуру элемента.

На рис. 6 представлен пример разметки изображения печатной платы, содержащей дефекты.

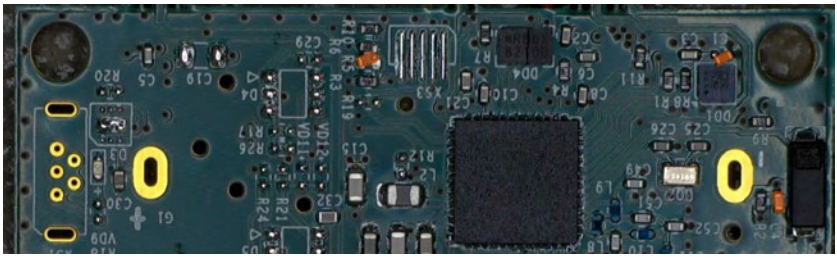


Рис. 6. Разметка изображения печатной платы, содержащей дефекты

После разметки новых изображений было проведено дообучение реализованной модели. Пример работы модели по обнаружению дефектов представлен на рис. 7

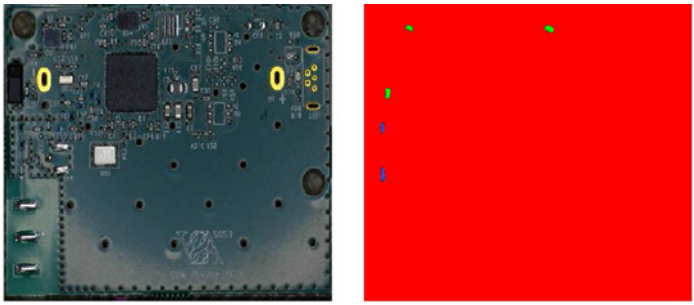


Рис. 7. Пример работы модели по обнаружению дефектов

Заключение

Реализованная модель способна обнаружить три типа дефектов печатных плат: отклонение положения компонентов, отсутствие компонентов, несоответствие цветов компонентов.

Однако модели не удается обнаружить поворот симметричного компонента, например, такого как конденсатор. Также в связи с низким качеством изображений не представляется возможным обучение модели обнаружению дефектов пайки.

Для повышения точности обнаружения дефектов необходимо улучшить качество входных изображений, что позволило бы обнаруживать даже незначительные повороты компонентов и дефекты, которые ранее детектировать не удалось.

Список литературы

1. Бергер, Е. Контроль качества печатных плат / Е. Бергер // Технологии в электронной промышленности. – 2012. – № 5(57). – С. 18-20.

2. Изосимова, Т. А. Разработка автоматизированной системы управления диагностикой печатных плат на основе машинного зрения/ Т. А. Изосимова, М. В. Максимова, О. В. Михайлова // Вестник НГИЭИ. – 2018. – №1. – С. 7-17.

3. Prabu, S. A Study on Image Segmentation Method for Image Processing / S. Prabu, J.M. Gnanasekar // Recent Trends in Intensive Computing. – 2021. – С. 419-424.

4. U-Net Architecture Explained [Электронный ресурс] : Geeksforgeeks – Режим доступа: https://www.geeksforgeeks.org/u-net-architecture-explained/

5. The original U-Net architecture [Электронный ресурс] : ResearchgGate – Режим доступа: https://www.researchgate.net/figure/The-original-U-Net-architecture

Разработка клиент-серверной системы анализа аудиоданных

С. А. Шипилов

Студент магистр

В. В. Твердохлебов

Студент магистр

А. В. Максимов

Старший преподаватель

Введение

В настоящее время анализ кашля может быть очень утомительным и длительным процессом для врачей, особенно при хроническом кашле. Это связано с тем, что кашель может быть вызван широким спектром основных состояний, от простой аллергии до более серьезных респираторных заболеваний.

У каждого человека присутствуют некие индивидуальные черты голоса. Выделяют две основные классификации кашля – сухой и влажный. В зависимости от стадии и типа заболевания можно определить, к какому классу относится кашель. Кашли для каждого индивидуальны. И даже кашли одного и того же человека сильно отличаются друг от друга в зависимости от ситуации.

Цель работы – разработка клиент-серверной системы анализа респираторной активности.

1. Функционал системы

Для хранения аудиофайлов обычно используют хранилища и взаимодействуют через FTP-протокол для их передачи. Разрабатываемая система должна выполнять следующие задачи.

- Хранить данные о пациентах;
- Хранить у каждого пациента соответствующие обследования;
- Возможность добавлять файлы на сервер;
- Реализация аутентификации и авторизации;
- Возможность создать отчет на основе данных.

Система должна иметь возможность проверять роли вошедших пользователей. Для проверки подлинности каждому пользователю (в данном случае пользователем является врач) будет выдаваться токен,

© Шипилов С. А., Твердохлебов В. В., Максимов А. В., 2025

который определяет, что именно этот пользователь с определенным идентификатором может выполнить какие-то действия в системе. Основным стеком данной системы является язык программирования Python и веб-фреймворк Django [1].

2. Основные сущности базы данных

Основными сущностями базы данных являются таблицы (рис. 1):

– Доктора (Doctor);
– Пациента (Patient);
– Обследования (Examination);
– Аудиофайлов обследования (FileOn);
– Общего отчета по обследованиям (Report).

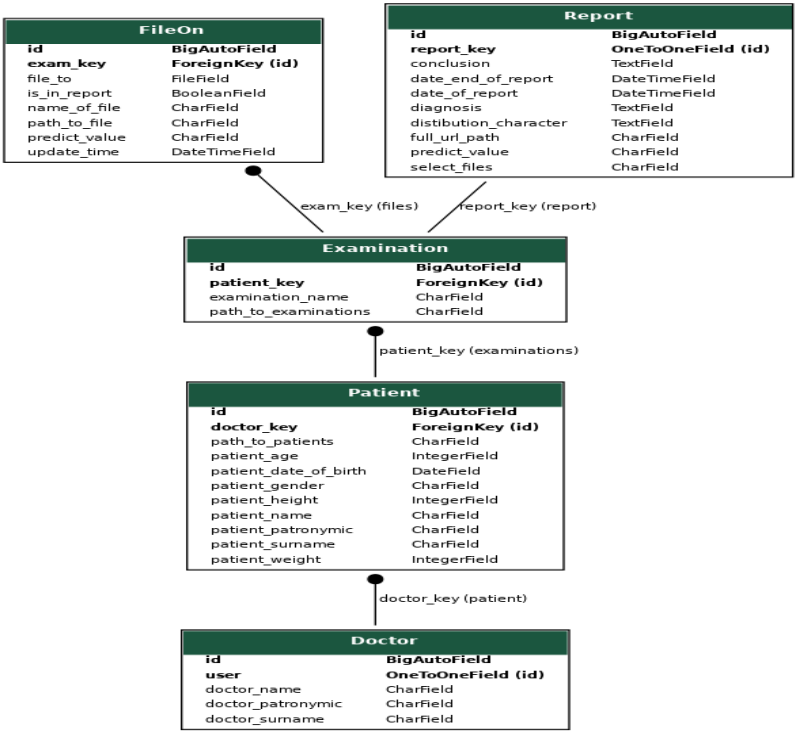


Рис. 1. Схема основных сущностей базы данных

Зашедший в систему врач должен иметь возможность просмотреть своих пациентов и их обследования. Когда в обследовании будут находиться файлы, у врача должна быть возможность просмотреть

аудиофайл и его частотные характеристики для дальнейшего анализа. На основе полученных данных врач может написать отчет о том, какие препараты или лечение необходимо назначить пациенту на основе обследования. Вывод аудиоданных пациента представлен на рис. 2.

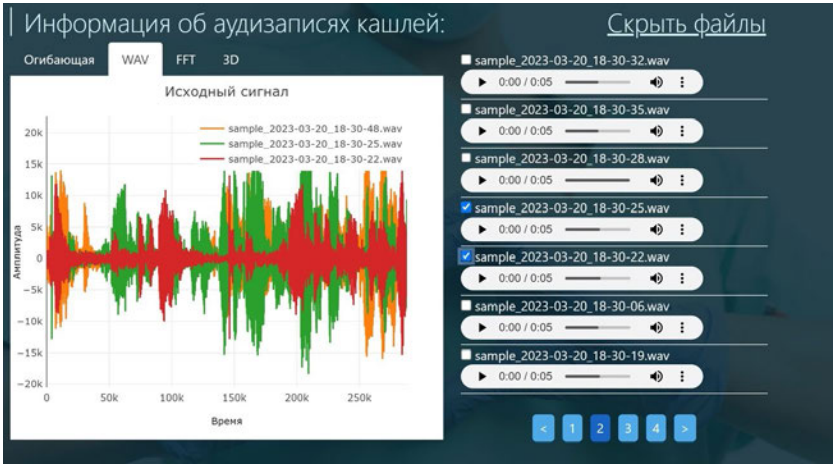


Рис. 2. Функционал для анализа аудиоданных

3. Моделирование API для внешних приложений

Одной из основных задач является отправка файлов на сервер. Пациенту дается прибор, записывающий его кашли. Нужно отправить этот файл на сервер тому пациенту, за которым закреплено обследование. Для этого необходимо создать для сервиса API, которое будет принимать файл в качестве параметра файл и дополнительные параметры для указания того, на какое обследование прикреплен этот файл. И после этого врач сможет проанализировать данные и сделать вывод. Основными API-запросами являются такие действия, как:

- Методы авторизации и аутентификации;
- Загрузка аудиофайла с мобильного приложения;
- Просмотр пациентов зашедшим врачом.

4. Фоновые задачи

Фоновые задачи полезны тогда, когда необходимо асинхронно выполнять какие-то однотипные действия. В качестве пакета для выполнения задач был выбран Python-пакет Celery [2].

Данный пакет предназначен для асинхронной и параллельной обработки большого количества задач, что обеспечивает высокую пропускную способность и повышенную масштабируемость.

Очередь задач Celery состоит из нескольких компонентов:

- Брокер сообщений;
- Задачи;
- Рабочие узлы;
- Результаты;
- Мониторинг.

Чтобы использовать Celery, нужно определить задачи, которые необходимо выполнить, и зарегистрировать их в приложении. Затем отправить задачу в очередь, и она будет выполнена асинхронно одним из рабочих узлов.

Пусть дана задача предсказать с помощью свёрточной нейронной сети, какой кашель может быть в загруженном аудиофайле. После того, как файл загружен, нужно добавить в стек определенную задачу, которая принимает выходные параметры: ID экземпляра объекта файла, путь к загруженному файлу и его название для инициализации.

Пример функции для создания фоновой задачи приведен в листинге 1.

Листинг 1

Фоновая задача на предсказание кашля у аудиофайла

```
@shared_task
def predict_cough(fileon_id, path_to_predict, filename):
    wavelet_solo(path_to_predict)
    result = predict(filename)
    fileon = FileOn.objects.get(id=fileon_id)
    return fileon
```

Данная задача создает по аудиофайлу вейвлет-скалограмму [3] и передаёт ее нейронной сети для прогнозирования значений. После успешного выполнения задачи результаты записываются в базу данных в соответствующий экземпляр, принадлежащий аудиофайлу.

Заключение

Данная статья посвящена разработке клиент-серверной системы для анализа респираторной активности, хранения и обработки аудиофайлов кашлей. В рамках данной работы были получены следующие результаты:

- Возможность проанализировать спектр файла;
- Реализована система авторизации и аутентификации;
- Создан интерфейс API для отправки файла на сервер и последующего анализа.
- Возможность вывода спектра графиков на странице.
- Встроена нейронная сеть для предсказания кашлей.

Список литературы

1. Меле, А. Django 2 в примерах / А. Меле. – М. : ДМК Пресс, 2019. – 408 с. – ISBN 978-5-97060-746-6. – Текст : электронный // Лань : электронно- 43 библиотечная система. – URL : https://e.lanbook.com/book/123711 (дата обращения : 14.05.2025). – Режим доступа: для авториз. пользователей.
2. Документация по инструменту Celery [Электронный ресурс]. – Режим доступа : https://docs.celeryq.dev/en/stable/index.html
3. Multi-Resolution Analysis [Электронный ресурс] : Wavelets – Режим доступа : https://www5.informatik.unierlangen.de/fileadmin/lectures/
4. 2012w/Lecture.2012w.IntroPR/wavelets.pdf

Обучение модели компьютерного зрения в CreateML для классификации болезней растений

Д. И. Шмойлов

Студент магистр

В. С. Тарасов

Старший преподаватель

Введение

Современное сельское хозяйство сталкивается с растущими вызовами, связанными с ранней диагностикой заболеваний растений. По данным Продовольственной и сельскохозяйственной организации AGBZ [1], ежегодные потери урожая из-за фитопатогенов достигают 20–40% от общего объёма, что угрожает продовольственной безопасности и экономике аграрных регионов.

Традиционные методы диагностики, основанные на визуальном анализе симптомов (пятна, изменение пигментации, деформация тканей), требуют высокой квалификации специалистов и часто приводят к ошибкам из-за схожести проявлений различных заболеваний.

Это создаёт критическую потребность в автоматизированных системах, способных стандартизировать процесс выявления патологий. Технологии нейросетей открывают новые возможности для решения этой задачи.

В данной работе предлагается обучение модели компьютерного зрения в CreateML [2] для выявления заболеваний растений на основе изображений. Точность модели будет протестирована на выборке изображений, включая данные из открытых интернет-источников, что позволит оценить её способность корректно классифицировать патологии в стандартизированных условиях. Результаты тестирования определят эффективность модели для дальнейшего практического применения.

1. Постановка задачи

Задача исследования заключается в разработке и валидации модели компьютерного зрения, обученной в CreateML, для автоматической диагностики заболеваний растений на основе анализа изображений. В

© Шмойлов Д. И., Тарасов В. С., 2025

рамках работы будет проведено тестирование точности модели на выборке данных из открытых интернет-источников.

Целью работы является практическое применение фреймворка CreateML для обучения модели классификации фитопатогенов, её интеграции в мобильное приложение и экспериментальное подтверждение диагностической точности на реальных данных. Результаты исследования позволят оценить потенциал использования подобных моделей в качестве инструмента для раннего выявления заболеваний растений в неконтролируемых условиях.

2. Набор данных

Для качественного обучения модели машинного обучения, дающей хорошие показатели, важно собрать качественный набор данных, именуемый также датасет. К обучающим данным предъявляются строгие требования, обусловленные необходимостью обеспечения высокой обобщающей способности модели, её устойчивости к вариациям входных сигналов и способности корректно работать в реальных условиях. Ниже представлены ключевые критерии выбора датасета:

– Репрезентативность данных – датасет должен отражать разнообразие условий, в которых модель будет применяться на практике. Это включает вариации освещения (естественное, искусственное, смешанное), ракурсы съёмки (вид сверху, боковой угол, крупный план), фоны (однородные, естественные среды) и артефакты (шум, размытие). Чем выше репрезентативность, тем меньше риск переобучения модели на идеализированных данных;

– Размер и баланс классов – объём данных должен быть достаточным для охвата внутриклассовой вариативности. Для задач с множеством классов (например, 38 заболеваний) рекомендуется не менее 1000 изображений на класс. Баланс между классами предотвращает смещение модели в сторону доминирующих категорий;

– Качество разметки – каждое изображение должно быть точно отнесено к соответствующему классу. Ошибки в разметке (ложные метки) приводят к некорректному обучению и снижению точности. Желательно, чтобы разметка выполнялась экспертами в области фитопатологии;

– Разнообразие географических и временных условий – для моделей, предназначенных для использования в различных регионах, критически важно включать данные, собранные в разных климатических зонах и сезонах. Это позволяет учесть влияние окружающей среды на визуальные проявления заболеваний;

– Наличие предобработки – предварительная обработка данных (нормализация размеров, удаление фона, коррекция яркости) сокращает время обучения и улучшает стабильность модели. Однако важно сохранить естественные артефакты, такие как тени или капли воды, чтобы избежать излишней идеализации;

– Доступность и лицензионная чистота – датасет должен быть легально доступен для исследовательских и коммерческих целей. Наличие открытой лицензии (например, CC BY-SA 4.0) упрощает интеграцию в публичные проекты.

Для решения задачи автоматической диагностики заболеваний растений был выбран датасет «New Plant Diseases Dataset» [3], размещённый на платформе Kaggle. Его структура и характеристики полностью соответствуют перечисленным критериям. На рис. 1 ниже, представлен пример данных.

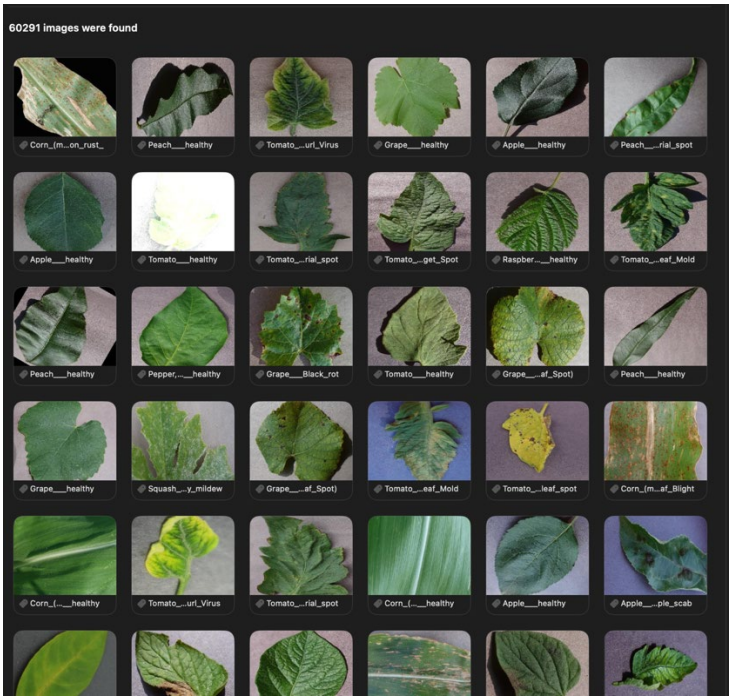


Рис. 1. Датасет «New Plant Disease Dataset»

Датасет «New Plant Diseases Dataset», доступный на платформе Kaggle, представляет собой масштабную коллекцию изображений,

разработанную для задач автоматической диагностики заболеваний растений методами компьютерного зрения. Он содержит 87 000 RGB-изображений с разрешением 256×256 пикселей, охватывающих 38 классов, включая здоровые растения и различные типы заболеваний сельскохозяйственных культур.

Каждый класс соответствует конкретной патологии, такой как фитофтороз томатов, мучнистая роса винограда или бактериальная пятнистость кукурузы, а также здоровым аналогам этих растений. Заболевания в датасете охватывают широкий спектр фитопатогенов — грибковые, бактериальные и вирусные инфекции, что обеспечивает разнообразие визуальных паттернов для обучения модели.

Данные организованы в три независимых набора: тренировочный, валидационный и тестовый. Тренировочный набор составляет 80% от общего объёма (69 600 изображений), валидационный — 20% (17 400 изображений), а тестовый включает 33 изображения для финальной проверки модели.

3. Обучение ML модели

После того, как обучающий набор был сформирован, становится возможным приступить к обучению нейронной сети. В качестве основной среды для обучения для данной задачи была выбрана CreateML Image Classifier [4], которая обеспечивает эффективную работу с изображениями и автоматизацию ключевых этапов, таких как подбор гиперпараметров и оптимизация архитектуры под мобильные устройства.

CreateML, работая с изображениями, использует архитектуру, основанную на свёрточных нейронных сетях (CNN), которые идеально подходят для задач классификации визуальных данных.

Для баланса между точностью и скоростью работы на мобильных устройствах выбраны следующие параметры:

– Feature factors для уменьшения глубины свёрточных слоёв на 30% по сравнению с базовой архитектурой для снижения вычислительной нагрузки, но с сохранением способности детектировать мелкие дефекты (например, точечные некрозы);

– Model availability для оптимизации под iOS через квантование весов и исключение избыточных операций, что сократило размер модели на 40% без потери качества;

– Iterations 500 эпох с применением ранней остановки — обучение прерывается автоматически, если точность на валидационной выборке не улучшается в течение 20 эпох.

Аугментации применены для имитации реальных условий съёмки:

– Случайные повороты (±25°), чтобы модель игнорировала ориентацию листа;

– Коррекция яркости и контраста (диапазон ±30%) для устойчивости к пересветам и теням;

– Масштабирование (от 80% до 120% исходного размера) для адаптации к разному расстоянию до объекта;

– Добавление гауссова шума (σ=0.05) — имитация помех при слабом освещении;

– Обрезка по области интереса для фокуса на центральной части изображения, где чаще расположены дефекты;

– Горизонтальное отражение для удвоения разнообразия данных без смещения статистики.

Результаты обучения модели, а именно значения функции ошибки и точности на тестовых и валидационных данных, представлены на графиках на рис. 2.

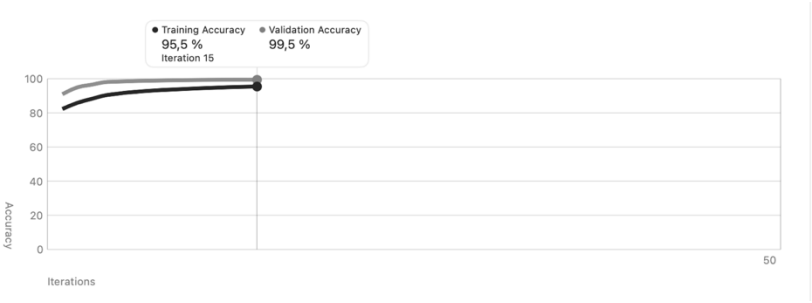


Рис. 2. Результаты обучения модели

Характеристики модели:

– Точность на тестовых данных = 0.955;

– Точность на валидационных данных = 0.995.

4. Тестирование модели

После завершения этапа обучения модели машинного обучения, был проведён комплексный анализ её производительности на специализированном тестовом наборе данных. Основной целью тестирования являлась верификация способности модели корректно идентифицировать заболевания растений в условиях, приближенных к реальным, а также оценка её устойчивости к вариациям входных данных. Для тестирования модели используется системные инструменты CoreML Model Prediction [5].

Для валидации модели использовался предварительно размеченный набор данных, включающий изображения здоровых и поражённых растений. Тестовые данные были структурированы таким образом, чтобы охватить широкий спектр условий:

– Различное освещение (естественное, искусственное);
– Вариации ракурсов (вид сверху, боковой угол);
– Наличие артефактов (блики, частичная заслонённость листьев).

Первичный фокус тестирования был направлен на диагностику фитофтороза томатов (Tomato late blight). На рис. 3 ниже представлен пример тестового изображения листа томата с некротическим поражением верхней части.

Рис. 3. Тестирование модели на листе томатов с поражением верхней части

Модель продемонстрировала 100% точность при классификации тестового изображения, соответствующего фитофторозу. Изображение-образец содержало следующие визуальные признаки:

– Морфология объекта. Лист томата, расположенный в центральной части кадра;

– Патологические изменения. Чётко выраженное почернение апикальной (верхушечной) части листовой пластинки, характерное для некротической фазы фитофтороза;

– Фоновые условия. Однородный серый фон, исключающий влияние посторонних объектов на процесс классификации.

Продолжая валидацию модели, проведено тестирование на изображении, соответствующем целевой пятнистости томатов (Tomato Target Spot). На рис. 4 ниже представлен пример тестового изображения листа томата с мелкими концентрическими пятнами на всем листе.

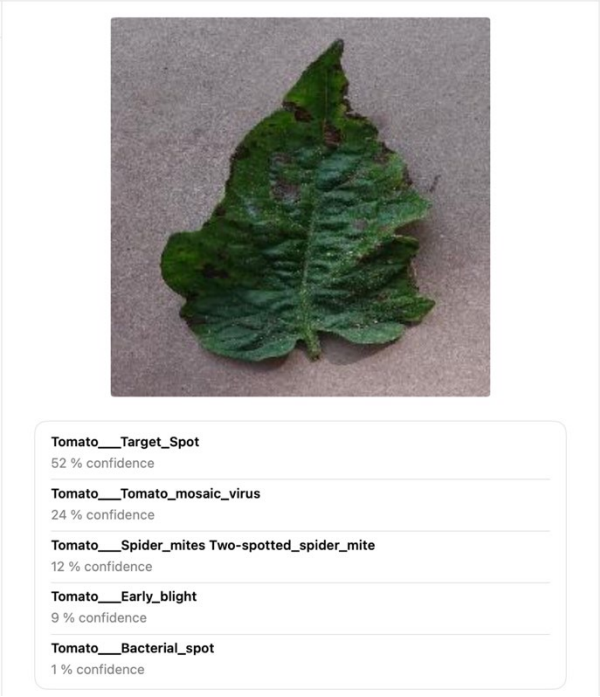


Рис. 4. Тестовый пример листа томата с целевой пятнистостью

Модель идентифицировала заболевание следующим образом:

– Целевая пятнистость (Tomato Target Spot) 52%;
– Вирус мозаики томатов (Tomato Mosaic Virus) 24%;
– Паутинный клещ (Two-Spotted Spider Mite) 12%;
– Ранняя пятнистость (Early Blight) 9%;

– Бактериальная пятнистость (Bacterial Spot) 1%.

Несмотря на низкую уверенность в 52%, модель верно определила целевую пятнистость как доминирующий класс. Это объясняется способностью архитектуры детектировать мелкие паттерны благодаря уменьшению глубины свёрточных слоёв на 30%, что сохранило чувствительность к микроскопическим дефектам.

В рамках продолжения валидации проведена оценка работы модели в условиях, имитирующих сложные полевые сценарии. На рис. 5 ниже, в этот раз тестирование выполнено на изображении здорового листа томата, снятого при экстремально ярком освещении, что привело к появлению засветов и цифровых шумов.

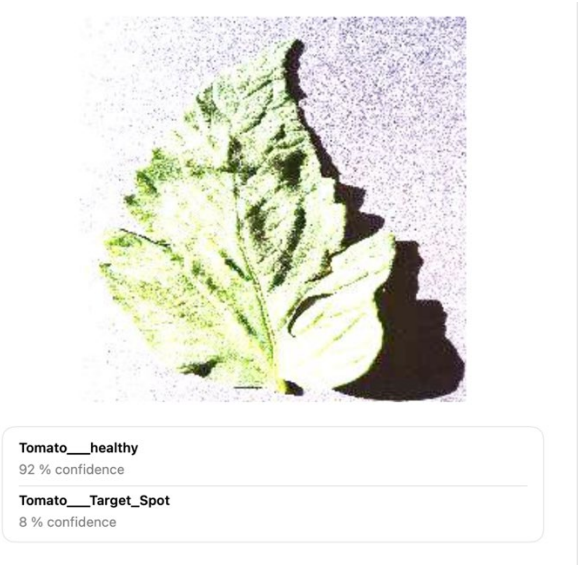


Рис. 5. Тестирование модели на здоровом листе томатов в условиях экстремально ярком освещении

Данный тест направлен на проверку устойчивости модели к артефактам съёмки, которые часто возникают при работе в солнечную погоду. Модель корректно идентифицировала объект как здоровый с уверенностью 92%. Изображение-образец содержало следующие визуальные признаки:

– Морфология объекта. Лист сохраняет естественную структуру, равномерный зелёный окрас без пятен, деформаций или иных аномалий;

– Артефакты съёмки. Изображение имеет пересвет по всей площади и искажения в теневых участках, вызванные перегрузкой сенсора камеры.

Для завершающей валидации проведён тест работы модели на изображении, снятом в естественной среде выращивания томатов. На рис. 6 ниже представлен пример тестового изображения листа томата с выраженными признаками заболевания, расположенного на фоне стеблей и незрелых плодов.

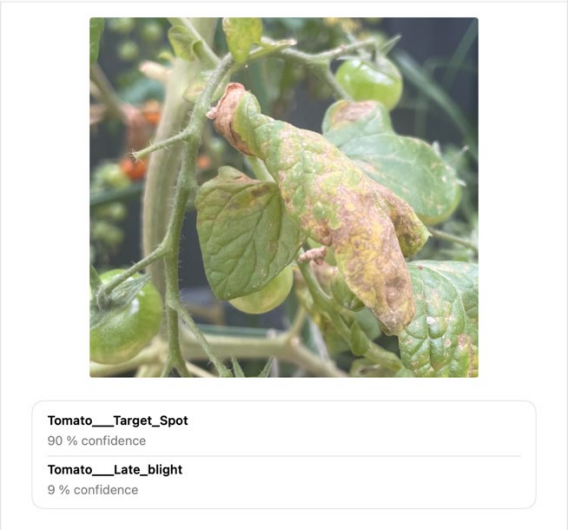


Рис. 6. Тестирование модели на листе томата с целевой пятнистостью в естественных условиях

Изображение-образец содержало следующие визуальные признаки:

– Морфология объекта. Лист томата, расположенный в центральной части кадра;

– Патологические изменения. Болезненное пятно в нижней части листовой пластинки и мелкие пятна в центральной зоне;

– Фон. Естественный дневной свет с распределением теней и гетерогенный фон с почвой и другими листьями.

Модель корректно идентифицировала целевую пятнистость томатов с уверенностью 90%, продемонстрировав высокую специфичность к характерным паттернам заболевания. Вторичный прогноз фитофтороза томатов (9%) связан с частичным визуальным

сходством некротических зон, однако модель корректно выделила доминирующий диагноз, проигнорировав фоновые артефакты.

Проведённые тесты подтверждают, что модель демонстрирует достаточную надёжность для интеграции в iOS-приложение, обеспечивая точность классификации целевых заболеваний даже в условиях сложного фона. Наличие второстепенных указывает на необходимость дополнительной калибровки модели для исключения перекрёстной реактивности между классами. Тем не менее, текущие результаты валидации позволяют считать модель готовой к внедрению.

Заключение

В данной работе реализован подход к автоматической диагностике заболеваний растений с использованием модели компьютерного зрения, обученной в CreateML. Модель продемонстрировала высокую точность на стандартизированных данных, а её интеграция в мобильное приложение подтвердила возможность локального выполнения предсказаний без зависимости от облачных сервисов.

Ключевым результатом стало успешное тестирование модели на изображениях из интернета, включая случаи с неидеальными условиями съёмки. Это подтверждает её применимость для раннего выявления патологий в неконтролируемых средах. Однако для повышения надёжности системы необходимо расширить обучающий датасет, добавив примеры с редкими заболеваниями и артефактами, характерными для пользовательского контента.

Список литературы

1. ООН: болезни и вредители виноваты в 40% потерь урожая [Электронный ресурс]. – Режим доступа: https://www.agbz.ru/news/oon-bolezni-i-vrediteli-vinovaty-v-40-poter-urozhaya/

2. CreateML documentation | Apple Developer [Электронный ресурс]. – Режим доступа: https://developer.apple.com/documentation/createml

3. New plant disease dataset | Kaggle [Электронный ресурс]. – Режим доступа: https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset

4. Creating image classifier model | Apple Developer [Электронный ресурс]. – https://developer.apple.com/documentation/createml/creating-an-image-classifier-model

5. Mlmodel testing [Электронный ресурс]. – Режим доступа: https://apple.github.io/coremltools/docs-guides/source/model-prediction.html

Разработка системы поддержки принятия решений при подборе и оценки персонала

Е. С. Юревич

Студент магистр

Е. А. Михайлюк

Доцент

Введение

Людям часто приходится сталкиваться с проблемой выбора. И принять правильное решение бывает очень сложно.

Выбор подходящего сотрудника компании, подбор персонала – не исключение. Рекрутинг персонала необходим практически во всех сферах. Например, при открытии новых точек продаж или компаний требуется быстро подобрать большое количество человек, также есть еще естественная текучка персонала. Эффективный рекрутинг необходим для поддержания конкурентоспособности бизнес-компаний.

И именно в рекрутинге проблема выбора является наиболее актуальной.

Целью данной работы является разработка системы поддержки принятия решений при подборе и оценки персонала. Необходимо выполнить следующие задачи: провести анализ существующих решений; изучить механизм решения задачи оптимальной остановки случайных процессов с целью разработки системы принятия подобных решений; разработать систему поддержки принятия решений при подборе и оценки персонала; провести анализ результатов, полученных с помощью реализованной программы.

1. Анализ существующих решений

Современный рекрутинг требует решений, позволяющих максимально автоматизировать процесс. Среди наиболее востребованных решений – системы управления наймом (ATS), которые автоматизируют размещение вакансий, сортировку резюме, планирование собеседований и составление отчетов. Практически все они направлены на работу с резюме и на отслеживание продвижения кандидата по воронке рекрутинга.

© Юревич Е. С., Михайлюк Е. А., 2025

Но что, если попробовать оптимизировать процесс отбора кандидатов именно во время самого собеседования.

Особых проблем может не возникнуть, когда можно спокойно рассмотреть все варианты, обдумать и выбрать лучший. Но, к сожалению, так бывает не всегда. Иногда приходится проводить отбор в очень сжатые сроки и в условиях появления соискателей в реальном времени. Бывает так, что при отказе в выборе определенного кандидата, мы теряем возможность вернуться к нему навсегда, его могут забрать более заинтересованные люди, компании, которые уже сделали свой выбор в пользу данного кандидата. Также может возникнуть ситуация, когда мы второпях уже приняли решение и выбрали специалиста, но потом понимаем, что выбранный вариант не является оптимальным. В таких случаях может быть сложно изменить свое решение и перейти к другому варианту [1-3].

2. Задача оптимальной остановки случайных процессов

Для создания системы поддержки принятия решений, изучив задачи оптимальной остановки случайных процессов, остановимся на задаче Мартина Гарднера. В ней идёт речь о поиске наилучшей стратегии выбора из случайной последовательности. Цель задачи – максимизировать вероятность выбора наилучшего из n объектов. В 1966 г. данный метод предложил С.М. Гусейн-Заде [4, 5]. В основе лежит метод «динамического программирования».

Метод решения заключается в том, что сначала следует пропустить первую 1/ e часть, только запоминая лучший вариант для будущего сравнения, а дальше выбирается первый элемент, который больше выбранного. При этом вероятность получить в конце концов наилучший вариант из всех n равна примерно 36,8% [6-7]. На рис. 1 представлена визуализация метода решения данной задачи.

Рис. 1. Визуализация метода решения

Реализовав данную задачу на языке Python, проведем анализ результатов данной программы. Вероятность выбрать один из 4-х лучших объектов из 1000 – ~ 60%.

3. Разработка системы поддержки принятия решений при подборе и оценки персонала

Решение данной задачи оптимальной остановки случайных процессов показывает отличные результаты в подобных условиях, поэтому мы можем использовать ее для разработки системы поддержки принятия решений при подборе и оценки персонала в условиях жесткого ограничения времени и при появлении кандидатов в реальном времени.

В качестве «уровня кандидата» будет выступать сумма баллов всех оценочных признаков – определенных требований к кандидатам, которые установит работодатель, также будет выбираться шкала оценки: 100-бальная, 10-бальная или да/нет (да = 1, нет = 0 или наоборот – пользователь сам указывает), при этом будет учитываться коэффициент важности определенных критериев (внутри программы это будет умножение полученных баллов за данный критерий на некоторое число, определяющее его приоритет). Для использования данной программы должно быть известно точное число кандидатов. Если кандидаты будут появляться в настоящем времени, то есть точное число претендентов будет не известно, то будем отталкиваться от ограничений во времени. Программа будет просить внести данные о количестве часов, которые будут затрачены на проведение всего отбора претендентов, и планируемое время собеседования каждого человека. Основываясь на этих данных, программа автоматически вычислит и определит общее количество кандидатов. Например, если все собеседования планируется провести в течение одного рабочего дня – 9 часов, а на каждого кандидата готовы потратить 15 минут, то общее число кандидатов будет равно 36.

Постепенно в процессе собеседования будет заполняться список претендентов и соответствующие им баллы. Дойдя до лучшего претендента по расчету программы, программа выдаст оповещение о том, что это может быть лучший вариант и что остановиться на нём может быть самым оптимальным решением. Но так как программа является системой поддержки принятия решения, и не предполагает выбор самого лучшего претендента со 100-процентной вероятностью, то пользователь может выбирать: остановиться на этом кандидате, если он его более чем устраивает, или же собеседовать дальше в поисках более подходящей кандидатуры, в том числе опираясь на уже просмотренных

претендентов с их баллами. На рис. 2 представлена блок-схема системы поддержки.

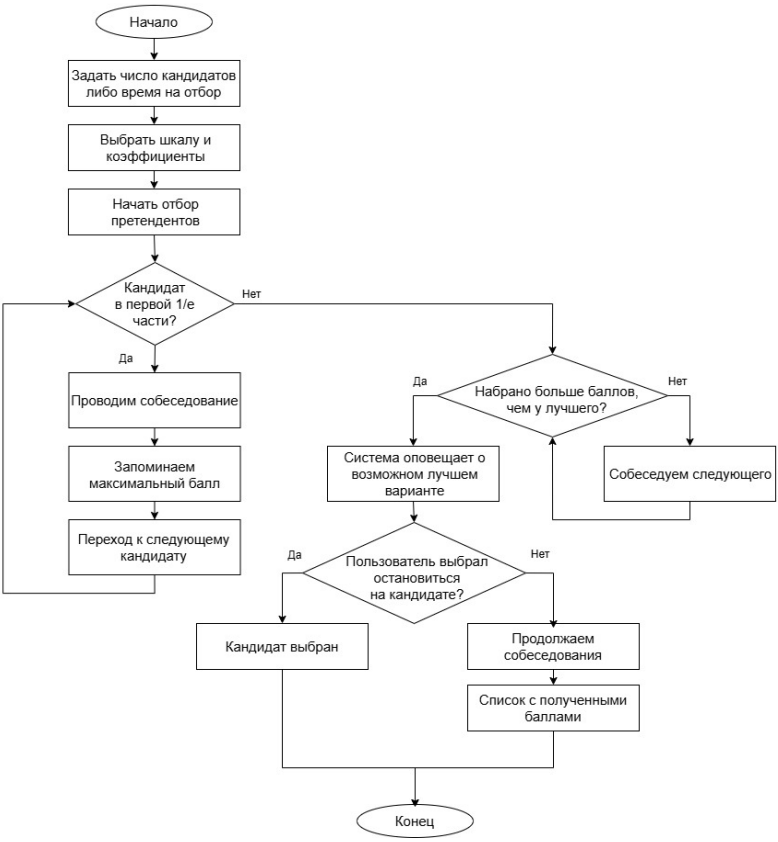


Рис. 2. Блок-схема системы поддержки

4. Анализ

Проведём анализ результатов программы:
Лучший кандидат был выбран – в 367 случаях из 1000 (36.7%)
Один из 3 лучших кандидатов – в 560 случаях (56%)
Один из 5 лучших – в 616 случаях (61.6%)
Один из 10 лучших – в 654 случаях (65.4%)
Вошёл в лучшую половину – в 819 случаях (81.9%)
На рис. 3 представлена гистограмма результатов.

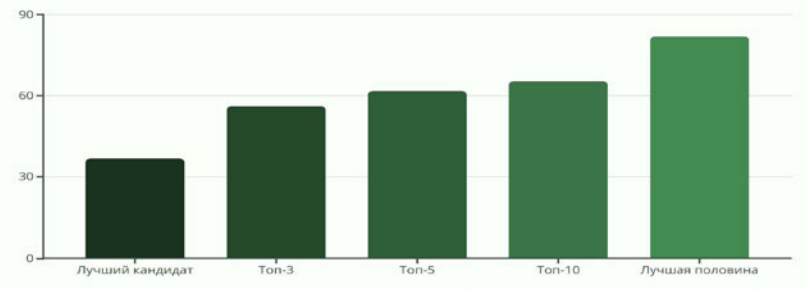


Рис. 3. Гистограмма результатов

Алгоритм находит лучшего кандидата в 36.7% случаев, а одного из топ-5 – в 61.6% случаев, входит в лучшую половину – в 81.9%.

Таким образом, алгоритм часто находит очень достойного кандидата, особенно если цель – выбрать кого-то из лучших 5–10, лучшего он выбирает примерно в трети случаев. Проведя сравнительный анализ, результаты которого представлены на рис. 4, выяснили, что для случаев, когда достаточно найти кандидата из топ-5, топ-10 лучше работает стратегия, когда отказываешься от первых 20%, а не от 1/e.

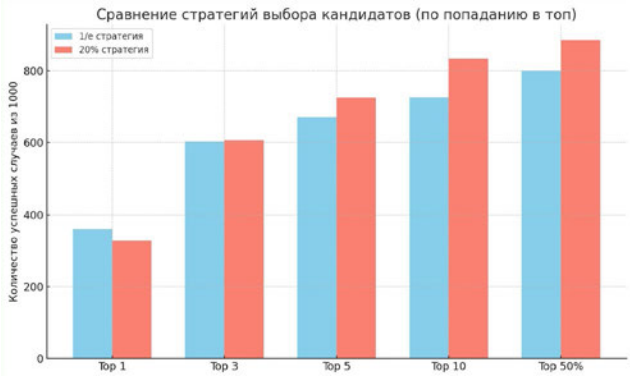


Рис. 4. Сравнительный анализ стратегий выбора кандидатов

Поэтому добавили в систему возможность указать: системе нужно стремиться к наилучшему варианту, или к топ-5, топ-10, и в зависимости от этого будет использована подходящая стратегия. Программа отлично работает как система поддержки принятия решений.

Заключение

Проведя анализ существующих решений и изучив механизм решения задачи оптимальной остановки случайных процессов, была разработана система принятия подобных решений при подборе персонала в условиях ограниченного времени, построена блок-схема и проведен анализ результатов, полученных с помощью реализованной программы, составлена статистика и предоставлены наглядные диаграммы для понимания работы системы.

Список литературы

1. Последовательный анализ / А. Вальд [и др.]; пер. с англ. П. А. Бакута; под ред. Б. А. Севастьянова. – М. : Гос. изд-во физ.- мат. лит., 1960 . – 327 с.
2. Arrow, K. J. Bayes and minimax solutions of sequential decision problems / K. J. Arrow, D. Blackwell, M. A. Girshick // Econometrica. – 1949. – Iss. 17. – P. 213–244.
3. Snell, L. J. Applications of martingale system theorems / L. J. Snell// Trans. Amer. Math.Soc. – 1952. – Iss. 73. – P. 293-312.
4. Гусейн-Заде, С. М. Разборчивая невеста / С. М. Гусейн-Заде. – М. : МЦНМО, 2003. – 24 с.
5. Гусейн-Заде, С. М. Задача выбора и оптимальное правило остановки последовательности независимых испытаний / С. М. Гусейн-Заде // Теория вероятности и её применение. – 1966. – Т. 11. – Вып. 3. – С. 534-537.
6. Chow, Y. S. Great Expectations: The Theory of Optimal Stopping / Y. S. Chow, H. Robbins, D. Siegmund. – Boston : Houghton Mifflin Company, 1971. — 534 P.
7. Ширяев, А. Н. Статистический последовательный анализ: Оптимальные правила остановки / А. Н. Ширяев. – М. : Наука, 1969 . – 230 с.

Анализ производительности сетевого взаимодействия RSocket, HTTP/2 и WebSocket

П. О. Яровитчук

Студент магистр

Н. К. Самойлов

Старший преподаватель

Введение

Протоколы сетевого взаимодействия играют важную роль в современных вычислительных системах, позволяя стандартизировать обмен данными и упрощая настройку и поддержку приложений.

Еще недавно стандарт HTTP 1.0 считался универсальным решением, однако при росте объемов передачи данных текстовые протоколы становятся узким местом многих систем. В современных реалиях для клиент-серверного и межсервисного взаимодействия требуются более оптимальные, надёжные и удобные сетевые протоколы. Современная индустрия все активнее начинает переходить на бинарные протоколы, которые исключают избыточность передачи текста, тем самым снижая размер передаваемых фрагментов и, соответственно, снижают затрачиваемое на передачу данных время.

Использование протоколов HTTP/2, RSocket или WebSocket позволяет решить многие проблемы, но для каждого протокола стоит провести анализ производительности сетевого взаимодействия [1, 2, 3].

1. Тестовый стенд

Для моделирования сценария запускаются два сервиса – Сервис-1 и Сервис-2 (рис. 1). Сервис-1 не может вернуть ответ на запрос до получения ответа от Сервиса-2. В итоге, имеется цепочка, в которой важны несколько критериев для протоколов, а именно: установление соединения между сервисами, время запроса-ответа и пропускная способность.

В качестве программного стека использован фреймворк Spring – де-факто многолетний стандарт для Java в коммерческой разработке [4]. Фреймворк предоставляет достаточный набор модулей для использования в различных сценариях и также поддерживает объекты анализа – HTTP/2, RSocket и WebSocket. За сбор метрик отвечает

© Яровитчук П. О., Самойлов Н. К., 2025

Prometheus. Физически, нагрузка генерируется на ноутбуке MacBook с процессором M1 Pro.

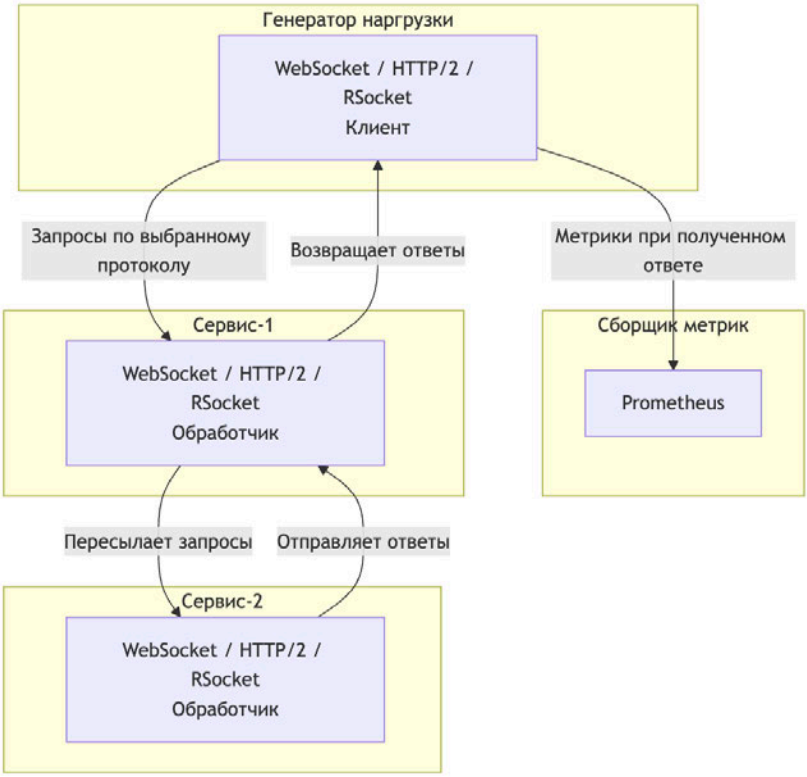


Рис. 1. Схема стенда, для тестирования сетевого взаимодействия

2. HTTP/2

Протокол, в отличие от первой версии HTTP, является бинарным и поддерживает мультиплексирование (возможность обрабатывать несколько логических потоков в рамках одного TCP-соединения). Данный протокол, является одним из основных, используемых в коммерческой разработке.

Так как был выбран фреймворк Spring, нельзя не подчеркнуть удобство перехода с HTTP 1.1 на HTTP/2, которое выражено лишь в переопределении в одной конфигурации из листинга 1 в файле application.yaml.

Листинг 1

Конфигурация включения HTTP/2

```
server.http2.enabled=true
```

При проверке производительности (параллелизм 50, перерыв между запросами 10мс), получены условные 5000 запросов в секунду (рис. 2).

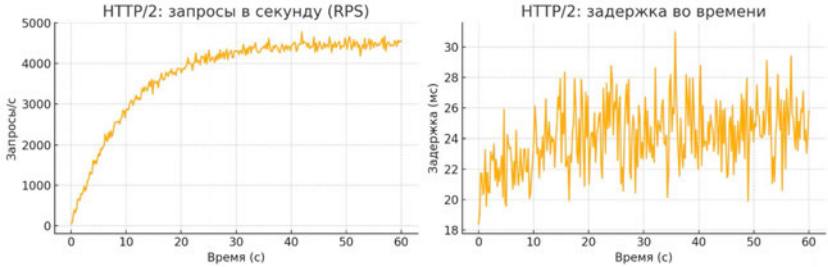


Рис. 2. График метрик протокола HTTP/2

3. WebSocket

Данный протокол, достаточно сильно отличается по семантике от HTTP, на прикладном уровне никак не контролируется получение ответа на отправленное сообщение. В коммерческой разработке данный протокол чаще используется для клиент-серверного взаимодействия, когда необходимо поддерживать длительное соединение клиента с сервером с опцией отправки некоторой информации с сервера на клиент.

Но для примера, с некоторыми доработками, получено практически схожее поведение с HTTP и возможность передачи бинарных данных.

При проверке производительности (параллелизм 50, перерыв между запросами 10мс), получены условные 2000 запросов в секунду (рис. 3). Показатель ниже, чем у HTTP/2, по причине недостаточных оптимизаций, по сравнению с HTTP/2, также стоит учитывать, что данный сценарий не является целевым для WebSocket.

4. RSocket

RSocket – самый молодой из рассматриваемых протоколов. RSocket является реализацией спецификации Reactive Streams, которая стала переосмыслением реактивного программирования из языка Java, но также и получила признание у многих сообществ других языков программирования [5].

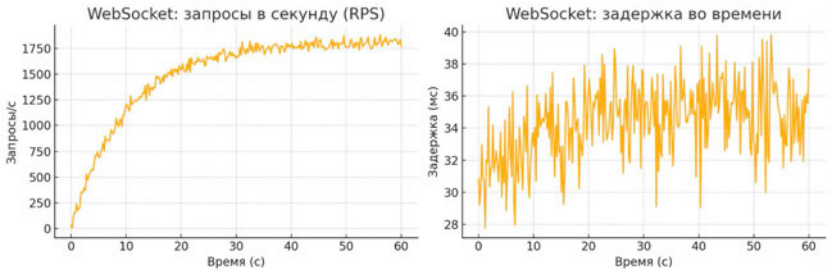


Рис. 3. График метрик протокола WebSocket

Особенностью данного протокола является набор различных сценариев использования. Например, rsocket поддерживает сценарий fire-and-forget, похожий по семантике на websocket. Сценарии request-response, request-stream, stream-stream, в целом схожи с HTTP/2. Кроме различных сценариев, RSocket поддерживает контроль нагрузки на уровне протокола, позволяя серверу регулировать объем входящих запросов.

Переход тестового стенда на RSocket – не простая задача. RSocket требует реактивного подхода, который не является обычным и простым для большинства разработчиков. Кроме этого, при переходе необходимо достаточно углубленно погружаться в набор настроек, где ошибка или неоптимальные параметры соединений могут достаточно критично сказаться на производительности.

После тестов (параллелизм 50, перерыв между запросами 10мс), получен условный пик в 1000 запросов в секунду (рис. 4), что является наименьшим показателем из уже протестированных протоколов.

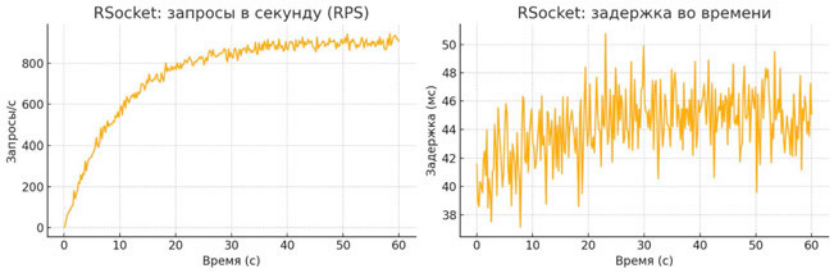


Рис. 4. График метрик протокола RSocket

Заключение

Из проведённых тестов можно с уверенностью выделить протокол HTTP/2, который показал себя достаточно эффективным и при этом простым в настройке. Также, можно убедится, что применение WebSocket в контексте межсервисного взаимодействии выглядит скорее как исключение, нежели как целевой сценарий. Говоря о RSocket, можно выделить высокую сложность в настройке, а также переходе с других протоколов, например HTTP 1.1, который, все еще используется во многих информационных системах.

Список литературы

1. RFC 7540. Hypertext Transfer Protocol Version 2 (HTTP/2) [Электронный ресурс] : Internet Engineering Task Force (IETF). RFC 7540, май 2015 – Режим доступа: https://datatracker.ietf.org/doc/html/rfc7540
2. RFC 6455. The WebSocket Protocol [Электронный ресурс] : Internet Engineering Task Force (IETF). RFC 6455, дек. 2011 – Режим доступа: https://datatracker.ietf.org/doc/html/rfc6455
3. RSocket Protocol Specification [Электронный ресурс] : RSocket.io. Официальная спецификация протокола RSocket – Режим доступа: https://rsocket.io/about/protocol
4. Spring Framework: поддержка RSocket [Электронный ресурс] : Spring Framework, раздел «RSocket» в официальной документации Spring – Режим доступа: https://docs.spring.io/spring-framework/reference/rsocket.html
5. Reactive Streams Specification [Электронный ресурс] : Reactive-Streams.org. Официальная спецификация для асинхронной обработки потоков с поддержкой backpressure – Режим доступа: https://www.reactive-streams.org/

Содержание

Статьи студентов бакалавриата

Акинин В. Ю., Акимов А. В. Тестирование безопасности веб-приложений методом черного ящика 4
Бородин Д. Н., Самойлов Н. К. Интеграция WebRTC с серверной логикой 12
Бортник Н. А., Бебнева С. В. Значимость интегрирования документооборота в бизнес-процессы посредством инструментария 1С 19
Братышев Т. Д., Воронцов Я. А. Разработка Telegram-бота для учёта личных финансов 25
Бутовецкая С. И., Копытина Е. А. Разработка методов генерации документов, сопутствующих процессу ГИА ФГБОУ ВО «Воронежского государственного университета 31
Высоцкий Д. М., Вялых С. А. Основные направления развития средств анализа уязвимостей информационных систем 39
Галимов А. Э., Самойлов Н. К. Извлечение ключевых слов из русскоязычных текстов с использованием Apache OpenNLP 47
Гончарова А. А., Храмов В. Ю. Оценка риска информационной безопасности систем обработки информации с использованием нечетких продукционных когнитивных карт 53
Джаббаров Р. Р., Копытина Е. А. Разработка серверного приложения для сбора рыночных исследований 61
Долженко М. С., Ушаков В. А. Разработка телеграм-бота для поиска данных о зарубежных авто 68
Дьякова А. С., Мальцев А. С. Анализ поведенческих характеристик пользователя для обеспечения защиты информации в информационной системе 75
Елисеев А. С., Самойлов Н. К. Реализация Exactly-Once Semantics в Kafka 83
Ермаков И. С., Илларионов И. В. Оценка целесообразности автоматизации функционального тестирования программного обеспечения на примере веб-приложения 91
Зинченко К. М., Гаршина В. В. Использование графов знаний для анализа и визуализации зависимостей между веб-сайтами 98
Кораблин Д. С., Попов М. И. Исследование явной схемы численного интегрирования бигармонического уравнения 103

Котов А. В., Акимов А. В. Применение нейронных сетей для ранжирования товаров 109
Котов Д. С., Соломатин Д. И. Система управления финансовым портфелем на базе глубокого обучения с подкреплением 118
Кравцова Н. Е., Соломатин Д. И. Сравнение характеристик моделей YOLO на примере распознавания объектов по фотографии 129
Ларина М. С., Отырба Р. Р. Разработка системы автоматизированного обнаружения дефектов и аномалий на изображениях 137
Леонов М. С., Соломатин Д. И. Недостаток публикаций об уязвимостях в программных продуктах: сравнение российских и мировых практик 145
Литвин Д. А., Мальцев А. С. Управление доступом к информационным ресурсам с использованием биометрической аутентификации 153
Лыгина П. С., Швырева А. В. Исследование устойчивости нейронной сети Inception-ResNet-v1 к атакам уклонения 161
Масыч С. Э., Борзунов С. В. Сравнение производительности реализаций языка Python в задачах ресурсоёмких вычислений 169
Медведев М. С., Проскуряков Е. Д. Реактивная реализация паттерна сага в условиях легаси архитектуры 174
Меркулов А. Е., Иванков А. Ю. Разработка программного комплекса для контроля защищенности внешнего периметра информационной системы 181
Минакова А. Ю., Минакова А. Ю., Туровский Я. А. Разработка гибридного нейроинтерфейса с имитацией P300-реакции 189
Михайлюк Е. А. Разработка картографического приложения PhotoMap 195
Новичихин В. Б., Чекмарев А. И. Модули «Курсовые работы и проектные семинары» информационной системы факультета 202
Паневина Е. С., Бебнева С. В. Оптимизация управления контрагентами в 1С: анализ проблем и пути решения 209
Пекшев А. С., Акимов А. В. Генеративная аугментация данных для задач детектирования летательных аппаратов 214
Петренко М. С., Сирота Е. А. Анализ и исследование некоторых методов прогнозирования нестационарных временных рядов 222
Платов Д. К., Проскуряков Е. Д. Разработка системы автоматизированного парсинга расписания факультета с интеграцией в Telegram-бота 230

Плотников И. А., Деревянко Е. А., Ерофеева Н. А., Максимов А. В. Информационная система 3D-сканирования раневых поверхностей .. 236
Путин П. А., Соломатин Д. И. Инструмент импорта задач из Polygon в Moodle 242
Рогачев Е. А., Соломатин Д. И. Применение SRE практик для построения отказоустойчивых систем 250
Рычков В. В., Кондрашов А. Н., Туровский Я. А. Информационная система расчёта вероятностей совпадения нуклеотидных последовательностей в задачах биоинформатики 260
Саков Е. А., Шлыков Д. Г., Соломатин Д. И. Разработка платформы для организации соревнований по программированию 268
Сапегин П. А., Соломатин Д. И. Разработка мобильной игры «Цифровой кузнечик 276
Ситникова В. В., Проскуряков Е. Д. Автоматическая суммаризация больших текстов: современные нейронные подходы и их ограничения 283
Скарга Д. Ю., Соломатин Д. И. Практики и подходы к организации баз данных в высоконагруженных системах 290
Смагина М. Б., Ушаков В. А. Разработка видеоплатформы с поддержкой транскрибации на базе мобильного приложения 297
Солопанов Н. Д., Туровский Я. А. Информационная система оценки особенностей управления оператором беспилотными летательными аппаратами 304
Танаев А. А., Маковий К. А. Использование Prometheus для обеспечения безопасности 312
Телегина А. С., Ушаков В. А. Разработка мобильного приложения для туристов 320
Третьяков Д. С., Ушаков В. А. Разработка мобильного приложения для домашних тренировок 326
Ульянов Н. С., Швырева А. В. Определение степени созревания плодов томатов в теплицах с помощью YOLOv5 332
Шепляков В. В., Тарасов В. С. Разработка и применение семейства алгоритмов K-Means для сегментации изображений рукописного текста с фоном 339
Шестопалов Д. А., Чекмарев А. И. Разработка приложения для размещения рекламы на радиостанции. Жизненный цикл 347
Шульженко Д. И., Соломатин Д. И. Анализ и применение структуры данных "Дерево палиндромов" в алгоритмах обработки строк 354
Юров Е. О., Коротков В. В. Использование (квантовых алгоритмов для решения задач дискретной оптимизации 361

Статьи студентов магистратуры

Анохин Р. Б., Махортов С. Д. Блочное хеширование для построения массива нечетких граней 369
Бозоров Р. М., Клинских А. Ф. Деградация современной компьютерной графики в фильмах и видеоиграх 378
Бондарева О. С., Митрофанова Е. Ю. Система автоматизированного принятия решения для задачи выбора метода идентификации пользователя 383
Борисенков В. Э., Тарасов В. С. Методы распознавания печатного текста в мобильных приложениях на Flutter 391
Бурнашев Д. С., Залыгаева М. Е. Реализация рекомендательной системы на основе двухстороннего вариационного автоэнкодера и GraphRag 401
Быченко Д. А., Самодуров С. А. Разработка программного обеспечения для анализа состояния приемников и проведения базового тестирования 409
Валеев В. И., Тарасов В. С. Интеграция игры на Unity с Web3 416
Горбачева И. В., Алейникова Н. А. Разработка программного обеспечения анализа устойчивости оптимального решения задачи нахождения критического пути с нечеткими параметрами 421
Грошев Н. С., Залыгаева М. Е. Исследование методов обработки естественного языка для автоматического резюмирования интервью 429
Дрюкова П. В., Матвеев М. Г. Информационные сервисы на электронной торговой площадке 436
Зюзин А. А., Швырева А. В. Дообучение модели естественного языка BERT с использованием GPU 444
Ивонин Д. А., Залыгаева М. Е. Алгоритм на основе вложения Татта для автоматической UV-развёртки 3D-фигур 452
Канюс А. С., Сирота А. А. Высокоточное определение координат объектов в стереосистеме с использованием методов глубокого обучения 460
Караваев В. Н., Сметанин И. Ю., Самойлов Н. К. Подходы к масштабированию WebSocket сервиса с сохранением состояния в микросервисной архитектуре с использованием Apache Kafka 468
Кравцов Д. Д., Лысачев П. С. Разработка (Telegram-бота для поиска и бронирования отелей 475

Кулиев Ф. Р., Сирота Е. А. Анализ и исследование методов параметрической идентификации на базе метода наименьших квадратов и рекуррентного метода наименьших квадратов 482
Лоскутова А. Е., Алейникова Н. А., Матвеев М. Г. Задача управления процессом обучения на основе нечетких цепей Маркова 491
Майорников М. В., Дрюченко М. А. Интерактивная семантическая сегментация изображений на основе моделей Thin Object Selection и RITM Interactive Segmentation 499
Малаков Л. А., Матвеев А. В., Максимов А. В. Аппаратная платформа системы биологической обратной связи в игровой форме 512
Мамонов Д. В., Самойлов Н. К. Сравнение алгоритмов сборщиков мусора на примере платформ Go и Java 518
Матюхин А. Е., Махно А. А., Киселев Е. А., Ушаков С. Н. Алгоритмы кратномасштабного анализа на основе базисных сплайнов 527
Мистрюков В. С., Коваль А. С. Классификация трафика при помощи глубокого анализа пакетов 535
Мишин П. С., Чекмарёв А. И. Разработка ядра для буферизованной обработки аудиофайлов 543
Обухова Э. В., Швырева А. В. Анализ лингвистического корпуса профилей студентов в социальных сетях 550
Павлов Я. Э., Попов М. И. Повышение точности разностных схем для бигармонического уравнения методом экстраполяции Ричардсона 561
Перова С. В., Акимов А. В., Гречишкин А. В. Классификация неисправностей пассивных радиочастотных устройств по S-параметрам при помощи нейронных сетей 567
Полев А. А., Вахтин А. А. Алгоритм оценки тематической близости на основе взвешенного сравнения тегов 576
Потапов А. Г., Романов А. В. Новый подход к организации программной архитектуры гибридного вычислительного кластера ФКН ВГУ 581
Ролдугин А. В., Маковий К. А. Реализация универсального парсинга XML-документов в сфере госзакупок 586
Сапелкин А. Г., Самойлов Н. К. Сравнительный анализ колоночных и реляционных СУБД при выполнении агрегатных запросов 594

Сафонова Ю. В., Атанов А. В. Модификации метода искусственных иммунных систем для решения задач негладкой оптимизации 603
Синяков И. В., Тарасов В. С. Искусственные иммунные системы в информационной безопасности 611
Сливкин Д. В., Иванков А. Ю. Генерация изображений цветов по текстовому описанию с помощью условных GAN и постобработка изображений с помощью Real-ESRGAN 621
Смирных П. В., Гавшин А. Л. Исследование технологии Kotlin Multiplatform при разработке web-приложений 634
Смирных П. В., Гавшин А. Л. Исследования возможностей GeoServer в контексте разработки геоинформационных систем 640
Соколов В. А., Савинков А. Ю. Анализ и сравнение встраиваемых операционных систем реального времени на базе STM32 645
Токарев Р. А., Туровский Я. А. Разработка и анализ эффективности электромиографического интерфейса в рамках управления дроном ... 651
Ушакова А. А., Гаршина В. В. Разработка приложения для поиска документов и генерации ответов с расширенным поиском 656
Федоров А. Н., Лысачев П. С. Сравнительный анализ архитектурных подходов к реализации CRS систем для гостиничного бизнеса 663
Филонов П. О., Копытина Е. А. Разработка web-сервиса для автоматизации работы отдела интеллектуальной собственности ФГБОУ ВО ВГУ 673
Фуфаева А. В., Максимов А. В. Реализация системы машинного зрения для выявления дефектов при производстве печатных плат 682
Шипилов С. А., Твердохлебов В. В., Максимов А. В. Разработка клиент-серверной системы анализа аудиоданных 687
Шмойлов Д. И., Тарасов В. С. Обучение модели компьютерного зрения в CreateML для классификации болезней растений 692
Юревич Е. С., Михайлюк Е. А. Разработка системы поддержки принятия решений при подборе и оценки персонала 702
Яровитчук П. О., Самойлов Н. К. Анализ производительности сетевого взаимодействия RSocket, HTTP/2 и WebSocket 708

Научное издание

ТРУДЫ МОЛОДЫХ УЧЁНЫХ
ФАКУЛЬТЕТА КОМПЬЮТЕРНЫХ НАУК ВГУ

В ы п у с к 5

Под общей редакцией *Д. Н. Борисова*

Издано в авторской редакции

Редакционная коллегия

Главный редактор:

Борисов Дмитрий Николаевич

Члены редакционной коллегии:

Кафедра информационных систем	Попова Елена Вячеславовна
Кафедра технологий обработки и защиты информации	Швырева Анна Владимировна
Кафедра цифровых технологий	Попова Александра Евгеньевна
Кафедра программирования и информационных технологий	Нужных Алексей Васильевич
Кафедра информационных систем управления	Коротков Владислав Владимирович