ТРУДЫ МОЛОДЫХ УЧЕНЫХ ФАКУЛЬТЕТА КОМПЬЮТЕРНЫХ НАУК ВГУ

Выпуск 2

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 2

Под редакцией кандидата технических наук Д. Н. Борисова

Воронеж
ООО «ВЭЛБОРН»
2022

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 2 / под ред. Д. Н. Борисова ; Воронежский государственный университет. – Воронеж : ООО «ВЭЛБОРН», 2022. – 397 с.

ISBN 978-5-6045486-5-3

В сборник включены научные работы студентов бакалавриата и магистратуры факультета компьютерных наук ВГУ, выполненные в 2021-2022 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

ISBN 978-5-6045486-5-3

Статьи студентов бакалавриата

Разработка математического аппарата для сервиса параметрического подбора недвижимости

В. А. Васильчиков
Студент бакалавриата
М. Г. Матвеев
Профессор
Н. А. Алейникова
Доцент

Введение

В последние несколько лет рынок недвижимости проходит стадию активной цифровизации. Представители рынка производят внедрения разноплановых сервисов, которые автоматизируют весь цикл покупки недвижимости в режиме онлайн. Тем не менее, подбор оптимального объекта недвижимости под заданные пользователем требования остается актуальной проблемой на сегодняшний день. Согласно исследованию Mail.ru Group [1], покупатели недвижимости в интернете тратят на выбор подходящего объекта от 1-го месяца до года активного поиска, что связано с обширным числом параметров играющих роль при выборе, а также большим числом предложений на рынке. Так, согласно вышеупомянутому исследованию, у пользователей имеется по меньшей мере одиннадцать параметров, которые играют важную роль при подборе недвижимости. Причем каждый из параметров имеет разную важность для пользователя. Так, например, цена квартиры, как правило, значительно важнее этажа, на котором она находится. Помимо этого, стоит учитывать расплывчатые требования и предпочтения покупателя при выборе недвижимости. Например, покупатель готов приобрести объект недвижимости и за 6 млн.р. и за 7 млн.р., однако более дешевый вариант будет предпочтительнее. В настоящей работе предлагается описание математического аппарата сервиса параметрического подбора недвижимости, который позволяет учитывать все вышеуказанные особенности, связанные с заданием значимости характеристик относительно друг друга и вводом требований и пожеланий к объекту в виде диапазона значений с предпочтением.

1. Формализация постановки задачи исследования

Пусть имеется некоторый объект недвижимости, представленный совокупностью своих взаимозаменяемых типов, различающихся значениями характеристических параметров. Объект недвижимости будет описан двумя типами характеристических параметров: техническими, например, количество комнат, общая площадь и т.д. и коммерческими, например, цена, тип сделки, год сдачи дома. Оба типа параметров будут представлены в виде одного вектора: $q = (q^1; ...; q^n)$, каждая n-я компонента которого принимает значения либо на количественной, либо на качественной шкале [2].

В свою очередь, каждый $k-й$ покупатель, $k = 1; 2; ...; K$ хочет приобрести объект недвижимости с желаемыми характеристическими параметрами. Скорее всего желания покупателя будут носить расплывчатый характер, что обуславливается взаимозаменяемостью вариантов объектов недвижимости. Таким образом, покупателю необходимо формализовать свои желания и возможности в виде вектора спроса с нечеткими характеристиками – $\tilde{q}_k = (\tilde{q}_k^1; \tilde{q}_k^2; ...; \tilde{q}_k^n)$. Компонентами вектора могут являться четкие, нечеткие или лингвистические переменные. Каждая переменная имеет кусочно-линейные функции принадлежности $f_{q_k^i}(x) \in [0;1]$, носители которых - $x_{\min} \le x \le x_{\max}$ отражают возможности покупателя, а значения – его предпочтения (желания). Для дискретных значений носителя функция принадлежности будет иметь табличный вид.

Из-за простоты программной реализации для задания $f_{q_k^i}(x)$ используется треугольная функция принадлежности [3]:

$$f_{q_k^i}(x) = \begin{cases} \dfrac{x-a}{m-a}, & x \in [a;m] \\ \dfrac{b-x}{b-m}, & x \in [m;b] \\ 0, & \text{в остальных случаях.} \end{cases}, \qquad (1)$$

где a и b – это левая и правая границы диапазона требований к параметру объекта недвижимости, а m – это желаемое значение.

Каждый j-й продавец, $j = 1; 2; ...; J$ аналогично формирует свое предложение в виде вектора с теми же компонентами, что и у покупателя, однако с четкими значениями параметров. Таким образом,

предложение продавца представляет собой совокупность таких параметров и имеет следующий вид: $q_j = (q_j^1;...;q_j^n)$.

2. Нахождение совокупного спроса

Для дальнейшего рассмотрения поставленной задачи необходимо формализовать определение локального соответствия [4]. Локальным соответствием называется соответствие предложения продавца покупательскому спросу по одной из характеристик объекта недвижимости. Мера локального соответствия пары «предложение $j-го$ продавца $k-му$ покупательскому спросу» по $i-му$ $(i=\overline{1,N})$ параметру вычисляется путем подстановки значений $x = q_j^i$ в соответствующую функцию $f_{q_k^i}(x)$. Обозначим такое соответствие через μ_{jk}^i.

При выборе объекта недвижимости также важно принять во внимание, что параметры объекта являются неравнозначными для покупателя. Например, цена для покупателя важнее, чем этаж, на котором находится квартира. В этом случае необходимо дать возможность *k*-му покупателю самому назначить веса соответствий для каждой характеристики, которые обозначаются как $\phi_k^i, i=\overline{1,n}$.

Следующим шагом в решении поставленной задачи является агрегирование полученных степеней соответствия с учетом неравнозначности каждого из параметров: $\mu_{jk} = agr\{\mu_{jk}^i\}$, где agr – оператор агрегирования, μ_{jk} – степени агрегированного соответствия.

В качестве оператора агрегирования для решения поставленной задачи был выбран интеграл Шоке по нечеткой мере, поскольку он удовлетворяет всем необходимым требованиям.

Вычисление интеграла Шоке основано на λ-нечеткой мере Сугено, которая выражает субъективный вес или значимость каждого подмножества критериев и определяется следующим образом:

$$\phi(\{q_j, j \in M^1\}) = \frac{[\prod_{j\in M}(1+\lambda\phi_j)-1]}{\lambda},\ M^1 \subseteq M, \qquad (2)$$

где ϕ_j – коэффициенты важности отдельно взятых частных показателей эффективности при построении обобщенного показателя, в нашем случае – это веса, назначенные экспертами ϕ_k^i, M – множество индексов $\overline{1,n}$, параметр λ можно найти из уравнения:

$$\lambda + 1 - \prod_{j \in M} (1 + \lambda \phi_j) = 0;\ \lambda > -1,\ \lambda \neq 0\,. \qquad (3)$$

Тогда интеграл Шоке для нахождения агрегированного соответствия μ_{jk} рассчитывается по формуле:

$$\mu_{jk} = agr(\mu_{jk}^1, ..., \mu_{jk}^n) = \sum_{l=1}^{n} (\mu_{jk}^l - \mu_{jk}^{(l-1)}) \phi(x / f_{\tilde{q}}(x) \geq \mu_{jk}^l)\,, \qquad (4)$$

где $l = \overline{1,n}$, $\mu_{jk}^1, \mu_{jk}^2, ..., \mu_{jk}^N$ – перестановка элементов $\mu_{jk}^1, \mu_{jk}^2, ..., \mu_{jk}^n$ такая, что $\mu_{jk}^1 \leq \mu_{jk}^2 \leq ... \leq \mu_{jk}^n$, $\mu_{jk}^0 = 0$, x – это подмножество компонент вектора q_j. При этом, если хотя бы одно локальное соответствие равно 0, то считаем, что $\mu_{jk} = 0$.

Таким образом, получаем матрицу агрегированных соответствий предложений продавцов с требованиями и предпочтениями покупателей.

3. Пример решения задачи

Рассмотрим пример расчета агрегированного соответствия предложений продавцов квартир покупательскому спросу. Пусть на рынке имеется 4 объекта недвижимости со значениями параметров, представленных в табл. 1.

Таблица 1

Объекты недвижимости, имеющиеся на рынке

	Цена, млн. руб	Кол-во комнат	Общая площадь, кв.м.	Этаж	Общая площадь кухни,	Кол-во Балкон/ Лоджия	Район	Санузел
1 объект	6	2	57	11	15	1	Коминтер-новский	Разд.
2 объект	5.5	2	50	10	10	1	Коминтер-новский	Разд.
3 объект	5.7	2	55	7	12	2	Левый берег	Разд.
4 объект	4	1	45	11	12	0	Советский	Совм.

Покупатель задает свои требования и пожелания к объекту недвижимости в виде совокупности четких и нечетких переменных (табл. 2-4).

Таблица 2

Требования к объекту недвижимости в виде нечетких переменных

	Цена	Общая площадь, кв.м.	Этаж	Общая площадь кухни, кв.м.
a	5 000 000	45	5	7
m	5 500 000	60	10	15
b	6 500 000	65	16	17

Таблица 3

Требования к объекту недвижимости в виде четких переменных

Количество комнат	Количество балкон/Лоджия
2	1

Таблица 4

Требования к объекту недвижимости в виде дискретных нечетких переменных

Район			Санузел	
Коминтерновский	Советский	Левый берег	Раздельный	Совместный
0.8	0.5	0	1	0

На следующем этапе покупатель задает веса соответствий для каждого из параметров объекта недвижимости (табл. 5):

Таблица 5

Веса соответствий для параметров объекта

Цена	0,9
Количество комнат	0,5
Общая площадь, кв.м.	0,8
Этаж	0,1
Общая площадь кухни, кв.м.	0,5
Количество Балкон/Лоджия	0,5
Район	0,7
Санузел	0,5

После того, как покупателем сформированы требования к объекту недвижимости, а продавцами сформированы предложения, можно приступить непосредственно к расчету агрегированных соответствий.

Первоначально необходимо рассчитать локальные соответствия для каждого параметра по формуле (1).

Так, таблица локальных соответствий объектов недвижимости с требованиями и предпочтениями покупателя выглядит следующими образом (табл. 6):

Следующий этап – это агрегирование полученных локальных соответствий. Первоначально необходимо проранжировать полученные значения локальных соответствий.

Ранжированные локальные соответствия для каждого из объектов выглядят следующим образом (табл. 7).

Таблица 6

Локальные соответствия

	Цена	Кол-во комнат	Общая площадь, кв.м.	Этаж	Общая площадь кухни, кв.м.	Кол-во Балкон/ Лоджия	Район	Санузел
1 объект	0,5	1	0,8	0,833	1	1	0,8	1
2 объект	1	1	0,333	0,833	0,375	1	0,8	1
3 объект	0,8	1	0,667	1	0,625	0	0	1
4 объект	0	0	0	0,4	0,625	0	0,5	0

Таблица 7

Ранжированные локальные соответствия

1 Объект								
Номер параметра	1	3	7	4	2	5	6	8
Значения соответствий	0,5	0,8	0,8	0,833	1	1	1	1
2 Объект								
Номер параметра	3	5	7	4	1	2	6	8
Значения соответствий	0,333	0,375	0,8	0,833	1	1	1	1
3 Объект								
Номер параметра	6	7	5	3	1	2	4	8
Значения соответствий	0	0	0,625	0,667	0,8	1	1	1

Продолжение таблицы 7

4 Объект								
Номер параметра	1	2	3	6	8	4	7	5
Значения соответствий	0	0	0	0	0	0,4	0,5	0,625

Затем необходимо рассчитать значение λ-меры Сугено. Для этого необходимо изначально вычислить само значение λ по формуле (3):

$$\lambda + 1 - (1 + 0,9\lambda) \bullet (1 + 0,5\lambda) \bullet (1 + 0,8\lambda) \bullet (1 + 0,1\lambda) \bullet (1 + 0,5\lambda) \bullet$$

$$(1 + 0,5\lambda) \bullet (1 + 0,7\lambda) \bullet (1 + 0,5\lambda) = 0.$$

Полученное значение λ = -0.9997.

Далее необходимо рассчитать саму λ-меру Сугено по формуле (2) (табл. 8):

Таблица 8

Полученные значения λ-меры

1-й объект	2-й объект
$\phi(1,3,7,4,2,5,6,8) = 0,99996$	$\phi(3,5,7,4,1,2,6,8) = 0,99996$
$\phi(3,7,4,2,5,6,8) = 0,99992$	$\phi(5,7,4,1,2,6,8) = 0,9986$
$\phi(7,4,2,5,6,8) = 0,983$	$\phi(7,4,1,2,6,8) = 0,997$
$\phi(4,2,5,6,8) = 0,943$	$\phi(4,1,2,6,8) = 0,989$
$\phi(2,5,6,8) = 0,937$	$\phi(1,2,6,8) = 0,987$
$\phi(5,6,8) = 0,875$	$\phi(2,6,8) = 0,875$
$\phi(6,8) = 0,75$	$\phi(6,8) = 0,75$
$\phi(8) = 0,5$	$\phi(8) = 0,5$

Так как в 3-ем и 4-ом объекте недвижимости имеются локальные соответствия равные 0, то их мы не учитываем.

Последний этап – это непосредственно вычисление самого интеграла Шоке, т.е. агрегирование совокупного соответствия покупательских требований и предпочтений с предложениями продавцов по формуле (4).

Таким образом, результаты проведенных вычислений представлены в табл. 9.

Таблица 9

Результат

Номер объекта	Агрегированное соответствие
1	0,998
2	0,996
3	0
4	0

Как видно из представленной выше таблицы, 1-й объект недвижимости является наиболее подходящим под заданные пользователем требования.

Заключение

В данной работе был рассмотрен математический аппарат сервиса, который позволяет пользователям в интернете подобрать оптимальные объекты недвижимости под заданные нечеткие требования. Работоспособность предложенной математической модели была доказана в ходе вычислений, проведенных в данной работе. Использование рассматриваемого сервиса при подборе недвижимости позволит пользователю более быстро и качественно произвести выбор объекта недвижимости онлайн.

Список лиитературы

1. Исследование Mail.ru Group: 79% пользователей рунета уверены, что VR-туры могут облегчить поиск квартиры [Электронный ресурс]: интернет-издание. – Режим доступа: https://vk.company/ru/press/releases/10470/

2. Матвеев М.Г. Информационные технологии формирования сервисов на электронной торговой площадке / Матвеев М. Г Шмелев М. А., Алейникова Н.А. // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2021. – № 1. – С. 63–73.

3. Костенчук М.И. Построение нечетких лингвистических переменных с использованием методов кластерного анализа данных / М.Т. Костенчук, Н.А. Дрожжин, Р.Л. Белоусов // Прикладная информатика. – 2015. – №1 – С.98-104.

4. A Game Model for Selecting a Seller’s Offer on the Marketplace / Mikhail Matveev; Natalia Alejnikova; Semen Podvalny; Svetlana Beletskaya // 2021 3rd International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA), 2021, Page(s):294 – 298.

Создание модуля генерации unit-тестов для Spring приложений на основе метода анализа AST-деревьев

А. А. Антонов

Студент бакалавриата

Н. К. Самойлов

Старший преподаватель

Введение

Важным и неотъемлемым этапом жизненного цикла разработки программного обеспечения является тестирование системы. Когда речь заходит о разработке API-интерфейсов особое внимание обращается на модульное тестирование конечных точек. Только эти элементы кода отвечают за доступ объектов к службам сервера, по этому их корректная работа жизненно необходима для адекватного функционирования системы. Ведется ли разработка по TDD либо команда придерживается другой философии – не важно, хороший разработчик обязан протестировать конечные точки еще до перехода к интеграционным тестам. Таким образом модульное тестирование уже давно стало частью повседневной жизни любого разработчика. При этом именно эта часть забирает значительную часть усилий, а значит и драгоценного времени. В связи с чем автоматизация тестирования программного обеспечения уже долгое время рассматривается как средство снижения затрат на разработку за счет устранения или сокращения этапа ручного тестирования. В связи с чем автоматизация тестирования программного обеспечения уже долгое время рассматривается как средство снижения затрат на разработку за счет устранения или сокращения этапа ручного тестирования. Данная работа посвящена разработке решения, позволяющего автоматизировать процесс написания модульных тестов.

1. Постановка задачи

При автогенерации тестов на основе кода необходимо убедиться в том, что их генерация не покроет валидными тестами не валидный код, а что еще более важно – не заменит корректные не валидные тесты на некорректные валидные.

При работе со Spring приложениями, и при их тестировании необходимо правильно обрабатывать объекты, управляемые IoC

контейнером, что делает этот процесс отличающимся от тестирования других приложений.

Использование AST-дерева позволяет получить доступ к логике программы, что означает, что анализируя код можно сгенерировать наборы тестов и данных к ним, обеспечивающие максимальное покрытие кода тестами.

Таким образом задача состоит в том, чтобы разработать решения, позволяющего генерировать модульные тесты для Spring приложений, отвечающее следующим требованиям:

– получение и анализ AST дерева для класса, обработка которого производится приложением;

– проверка существования сгенерированных тестов и генерация новых с учетом уже существующих;

– генерация модульных тестов, адекватных приложениям использующим фреймворк Spring и объекты, управляемые Spring IoC контейнером;

– генерация наборов данных, обеспечивающих максимальное покрытие кода.

2. Анализ AST-дерева

При рассмотрении среднестатистического Unit-теста (рис. 1) и сопоставлении его с любым небольшим фрагментом AST дерева (рис. 2), построенного инструментом JavaParser, становится понятно, что анализ существующего кода будет происходить в несколько этапов.

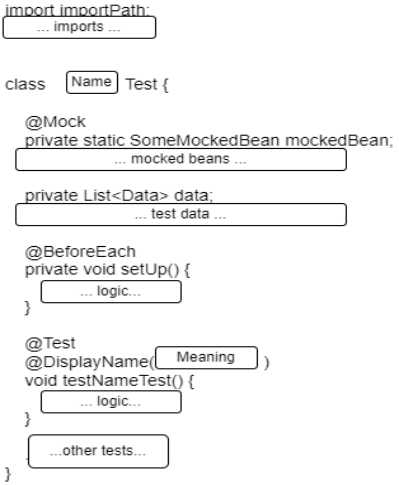
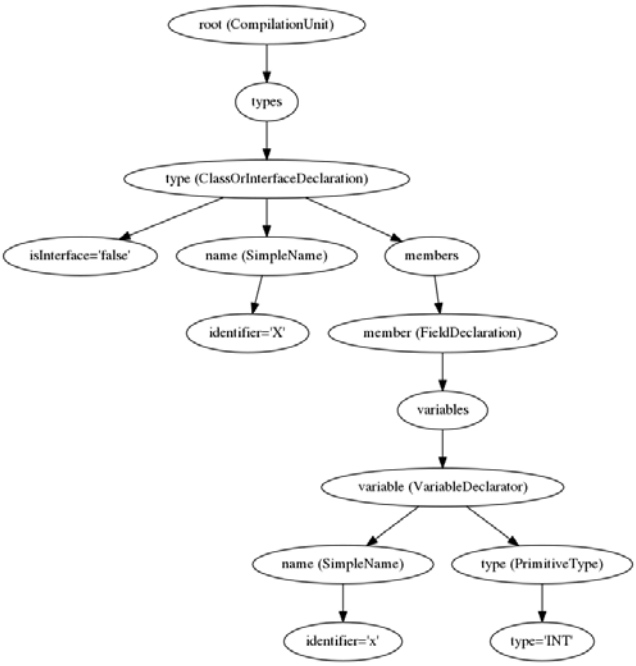


Рис. 1. Шаблон unit-теста



Класс с декларированным полем

Рис. 2. Пример фрагмента AST дерева

Этапы анализа:

– Выявление существующих классов (в дереве отображаются как узлы ClassOrInterfaceDeclaration) с помощью обхода в глубину через корневой узел CompilationUnit [1, с 10].

– Сбор сведений об импортах на основе поля imports узла CompilationUnit.

– Сбор на уровне класса сведений об использовании аннотаций по отношению к полям и самому классу с помощью узла MarkerAnnotationExpr.

– Сбор на уровне класса информации о всех существующих методах (узлы MethodDeclaration[1, с 7]), соответствующих выбранной глубине построения unit-тестов (имеется в виду опциональное покрытие private и protected методов).

– Для каждого метода в зависимости от передаваемых параметров генерируется набор соответствующих тестов и данных.

3. Средства реализации

В качестве средств реализации разрабатываемой системы были выбран высокоуровневый язык java, что обусловлено разработкой для Spring-api, а также следующие инструменты:

– JavaParser – позволяет строить и модифицировать AST деревья, также использование совместно с ним библиотеки javasymbolsolver – позволяет в ходе анализа определить тип переменной по ее имени, а также уровень объявления переменной;

– Mockito – фреймворк, позволяющий предоставить тестируемому элементу экземпляры классов, которыми он должен пользоваться при работе [2];

– JUnit5 – среда тестирования для приложений Java [3].

4. Этапы обработки

В результате анализа предметной области и требований было принято решение реализовать программу, выполняющую поэтапную обработку сущностей, передавая их от одного класса-обработчика к другому (рис. 3).

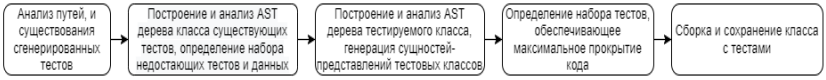


Рис. 3. Этапы работы программы

5. Сущности и конструкторы

В качестве сущностей, передающих информацию от анализатора к генератору, были выбраны следующие классы: UnitTest и TestClass (рис. 4). Именно они становятся накопителями результатов анализа AST дерева.

6. Анализ дерева, генерация сущностей классов

Для выполнения построения AST-дерева, генерации сущностей и их заполнения в ходе анализа был выполнен набор утилит, представленный на диаграмме классов (рис. 5).

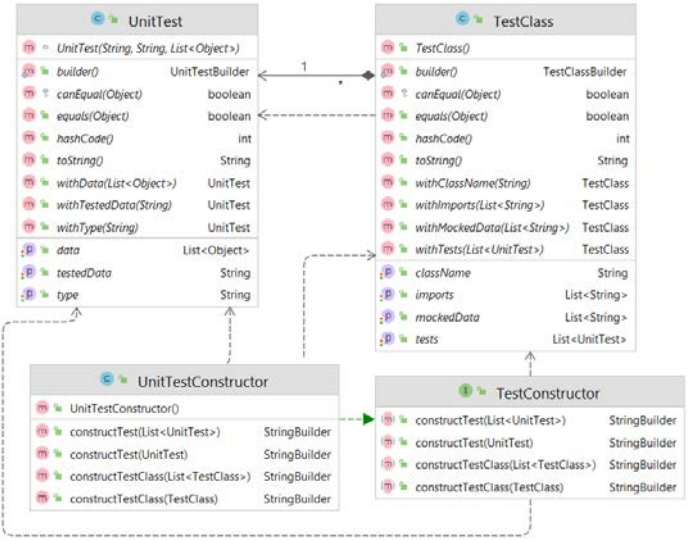
Начало выполнения инициализирует ASTGenerator – фасад системы.

FileManager – утилита необходимая для анализа существующих тестов и передачи информации о них анализатору.

ASTTreeManager – отвечает за построение AST дерева и передачу корневого узла анализатору.

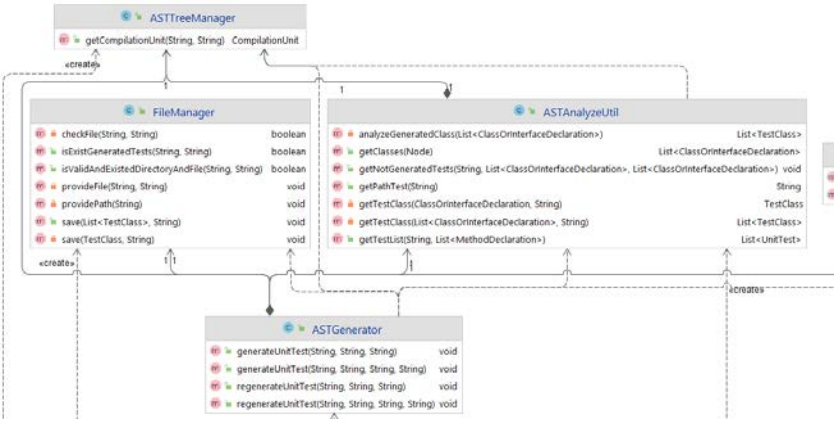
После чего на основании поиска в глубину ASTAnalyzeUtil определяет классы, декларированные в целевом файле, не имеющие сгенерированных тестов, а также создает объект сущности TestClass, в

который помещаются аннотированные поля, а также сами классы, макеты которых необходимо будет воспроизвести для вызова при тестировании [2]. Также в ходе анализа формируется и пул импортов.



Сущности и конструкторы тестов

Рис. 4. Фрагмент диаграммы классов



Утилиты анализа, генерации сущностей и AST дерева

Рис. 5. Диаграмма классов

7. Генерация набора тестов и данных

Поскольку определение необходимых для метода тестов исходит из его параметров, выбор такого набора происходит в соответствии со схемой (рис. 6).

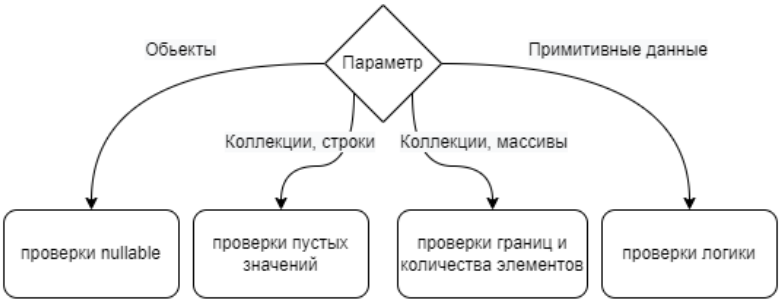


Рис. 6. Вариативность обработки параметров

Таким образом различно будут обрабатываться:

- массивы;
- коллекции;
- объекты классов;
- примитивные типы данных.

В коде данную обработку выполняет класс UnitTestGenerator путем выбора из словаря соответствующего типу параметра обработчика, являющегося реализацией интерфейса TestGeneratorByParam с дальнейшим делегированием обработки ему (рис. 7).

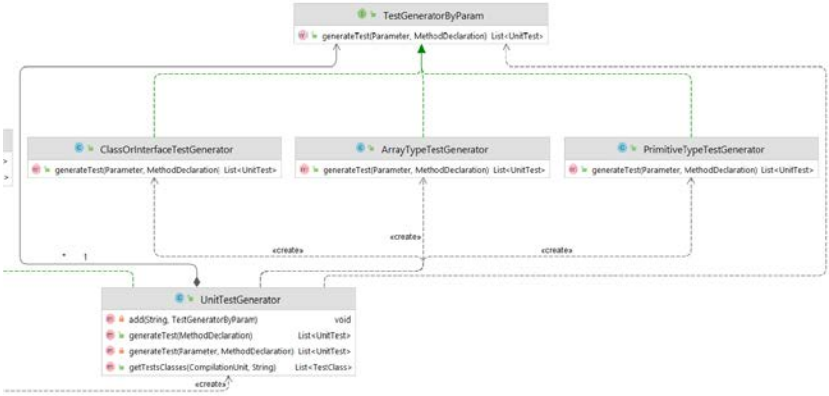


Рис. 7. Диаграмма классов, обработчиков генерации тестов.

8. Результат работы программы

Результатом выполнения полученной программы является класс (рис. 8). Жирным шрифтом на рисунке выделены сгененрированные блоки данных

```
class FilteringServiceImplAutogeneratedTest {

    @Mock
    private static FilteringService filteringService;

    private List<ProductDto> unfilteredProducts;
    private List<ProductDto> titleFilterProducts;
    private List<ProductDto> fullTitleFilterProducts;


    @BeforeEach
    private void init() {
    filteringService = new FilteringServiceImpl();
        ProductDto product1 = new ProductDto()
                .withTitle("title1")
                .withMaxPrice(20.0)
                .withMinPrice(10.0)
                .withSearchFrequency(18L);
        ProductDto product2 = new ProductDto()
                .withTitle("title2")
                .withMaxPrice(25.0)
                .withMinPrice(15.0)
                .withSearchFrequency(14L);
    …
    }


    @Test
    @DisplayName("priceFilter null products testing")
    void priceFilterNullTest() {
        assertNull(filteringService.priceFilter(null, 0, 1));

}
```

Рис. 8. Фрагмент сгенерированного класса

Тестирование программы проводилось на основе сервисов REST-приложения, при запуске сгенерированного файла ошибок на этапах компиляции и рантайма не выявлено, unit-тесты выполняются успешно

Заключение

Данная статья посвящена разработке и реализации решения, позволяющего автоматически генерировать набор unit-тестов для Spring приложений. В ходе нее был произведен анализ способов получения и обработки AST деревьев, а также воспроизведен генератор, защищенный от перекрытия невалидного кода. В дальнейшем планируется добавить обработку бизнес-логики, включенной в stream chains, что обеспечит лучшее покрытие кода, а также добавление возможности выбора необходимого уровня покрытия.

Взгляд на unit-тестирование с точки зрения анализа AST-дерева позволил получить достаточные данные для генерации тестов,

контролирующих поведение программы при не предоставленных или незаполненных данных так и для тестирования бизнес-логики.

Несмотря на сложность реализации данного решения в сравнении с проведением ручного тестирования, выбранный подход раскрывает большие возможности по значительному сокращению времени проведения модульного тестирования Spring приложений.

Список литературы

1. Smith, N. JavaParser: Visited Analyse, transform and generate your Java code base : Leanpub book / N. Smith, D. van Bruggen, F. Tomassetti. – Leanpub, 2019. – 59p.

2. Mockito и как его готовить, тестирование IT-систем Java [Электронный ресурс] : статья. – Режим доступа : https://habr.com/ru/post/444982/

3. Туториал по JUnit 5 – Введение [Электронный ресурс] : статья. – Режим доступа : https://habr.com/ru/post/590607/

Разработка системы управления вентиляцией воздуха на базе микроконтроллера Arduino

Д. И. Белашков

Студент бакалавриата

П. С. Лысачев

Старший преподаватель

Введение

Современное здание, вне зависимости от его функции – административное, жилое, производственное – состоит из множества систем, каждая из которых отвечает за тот или иной процесс, происходящий в строении. В ходе функционирования здания, эти подсистемы решают различные задачи, многие из которых чрезвычайно важны и требуют большое количество времени на управление и обслуживание. При усложнении общей системы, растет число подсистем и их сложность, по этой причине управление ими требует все больших расходов, идущих на обслуживающий персонал, ремонт и эксплуатации. Данные проблемы впервые стали заметны на больших производственных комплексах и привели к внедрению автоматизации практически на всех этапах производства.

Несмотря на то, что обычный жилой дом не так сложен, как, например, металлургический комбинат, в нем в последнее время появилось огромное количество автоматизированной техники. Например, это роботы-пылесосы, чайники и стиральные машинки, которые можно запустить из приложения на смартфоне, «умные» телевизоры и прочая мелкая и крупная бытовая техника.

Для упрощения работы с различной техникой, зачастую требующей отдельных пультов, приложений или иных контроллеров, разумно использовать системы, которые могут объединить все в одну сеть. Для демонстрации принципов работы такой системы, можно взять вентилятор или кондиционер и, на базе доступного программируемого микропроцессора Arduino Uno, сделать пульт для него, управляемый с персонального компьютера (тем самым, получив возможность для дистанционного выставления таймера, режимов работы и так далее).

1. Принципы работы сети ZigBee

ZigBee – это открытый стандарт беспроводной связи для различных систем автоматизации. Технология ZigBee позволяет создавать самоорганизующиеся и самовосстанавливающиеся беспроводные сети с автоматической ретрансляцией сообщений. Сети ZigBee при относительно небольших скоростях передачи данных обеспечивают гарантированную доставку пакетов и защиту передаваемой информации. [1]

Стандарты IEEE 802.15.4 и ZigBee 2007 Specification определяют стандартизированные протоколы, которые обеспечивают сетевую инфраструктуру, необходимую для беспроводных сетевых приложений с большим числом датчиков и исполнительных механизмов. IEEE 802.15.4 определяет физический и MAC уровни, а ZigBee определяет сетевой уровень и уровень приложений (рис. 1).

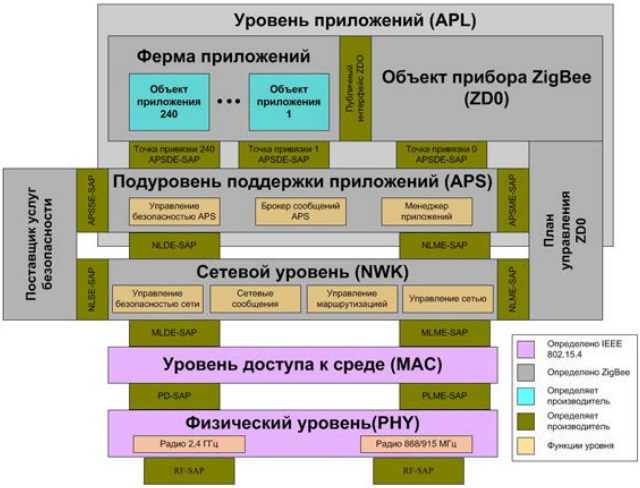


Рис. 1. Стек протоколов ZigBee

Спецификация ZigBee регламентирует стек протоколов взаимодействия узлов сети, в котором протоколы верхних уровней используют сервисы, предоставляемые протоколами нижележащих уровней.

Уровень приложений (APL) состоит из фермы приложений (Application Framework), объекта устройства ZigBee (ZDO) и подуровня поддержки приложений (APS).

Сети ZigBee включают в себя несколько типов устройств – координаторы (ZC), роутеры (ZR) и конечные устройства (ZED). Координатор запускает сеть и управляет ей, являясь центром сети. Роутеры расширяют зону покрытия сети, восстанавливают маршруты и, в основном, передают маршрутные пакеты. Как правило, они подключены к стационарному источнику питания. Конечные устройства выполняют основные функции – например, умные лампочки или розетки. Тем не менее, большинство девайсов на основе ZigBee умеют выполнять роль как роутера, так и конечного устройства [2]. Пример сети показан на рис. 2.

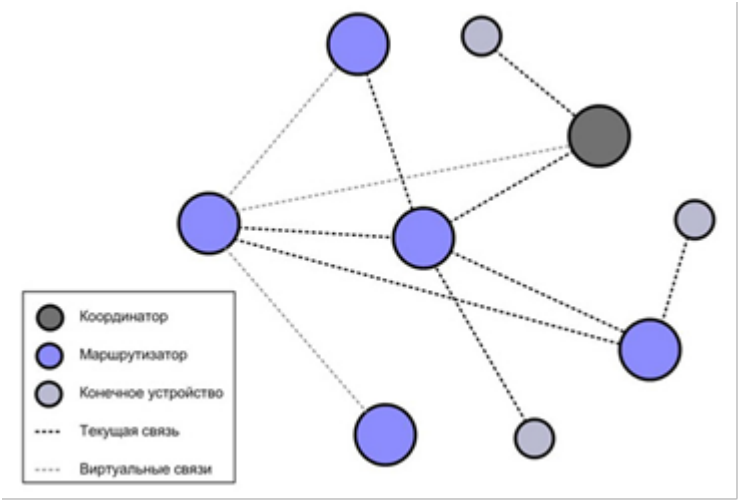


Рис. 2. Ячеистая сеть ZigBee

2. ZigBee и Arduino

Для системы управления вентиляцией воздуха необходимо реализовать следующую сеть: ZED (модуль с инфракрасным светодиодом, который подает команды вентилятору) → ZR (роутер, например умная лампочка) → ZC (координатор, который связан с сервером или ПК). Чтобы упростить реализацию, имеет смысл исключить из сети роутер и не реализовывать включение и отключение питания для вентилятора (при необходимости, для этого может использоваться умная розетка или реле, установленная в кабель питания).

Модули XBee используют протоколы ZigBee и являются радиомодемами для защищенной передачи данных с минимальным

энергопотреблением и небольшой скоростью. Реализация данной системы на основе Wi-Fi или Bluetooth так же возможна, но они не обеспечивают достаточное шифрование.

Несмотря на то, что для управления вентилятором не так важна защищенность данных, как, например, для системы жизнеобеспечения, стоит заранее задуматься о возможных последствиях небрежного отношения к безопасности передачи информации. Например, возможен сценарий случайной ошибки, когда команды, посылаемые одному устройству, принимаются и исполняются другим.

На рис. 3 изображена схема подключения модуля XBee к Arduino Uno. Данная схема является ZED (конечным устройством), которое будет обмениваться информацией с другим модулем XBee, установленным в персональном компьютере через USB-интерфейс. Это конечное устройство, в зависимости от установленных на него датчиков, сенсоров и модулей, а также от прошивки, может совмещать в себе несколько функций – например, сбор информации о влажности и температуре воздуха.

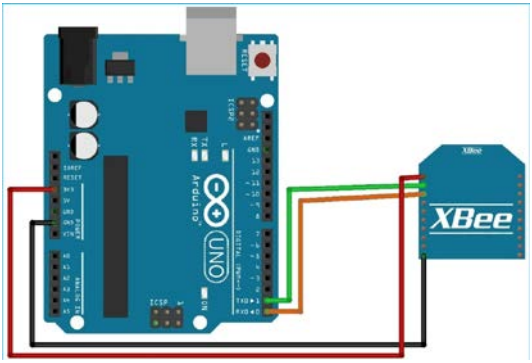


Рис. 3. Схема подключения модуля XBee к Arduino Uno

3. Реализация

Кроме сборки «пульта» необходимо решить проблему с передачей сигнала вентилятору. Поскольку каждый производитель техники используют свой протокол, к каждому устройству необходим свой подход. Эту проблему решают готовые базы данных по основным производителям и моделям, но для упрощения системы возьмем подобранные коды для вентилятора Vitek.

Передатчик, при получении определенной команды от пользователя, посылает пакет, который эмулирует нажатие клавиш пульта. Например, в пакете из восьми байтов нас будут интересовать

только те байты, где передается информация ON/OFF, mode (1,2,3), quiet (тихий режим), timer. Посылая различные команды, подбираем те, на которые вентилятор будет реагировать.

Приемник, присоединившись к сети координатора, получает от него пакеты с командами для вентилятора и, при помощи инфракрасного светодиода, подает их на вентилятор.

Таким образом, наша система логическим образом делится на несколько модулей: модуль-передатчик, модуль-приемник и управляющая программа.

Прошивка Arduino и модулей XBee осуществляется при помощи программного обеспечения XCTU. Иллюстрацию того, как выглядят используемые компоненты, можно увидеть на рис. 4.

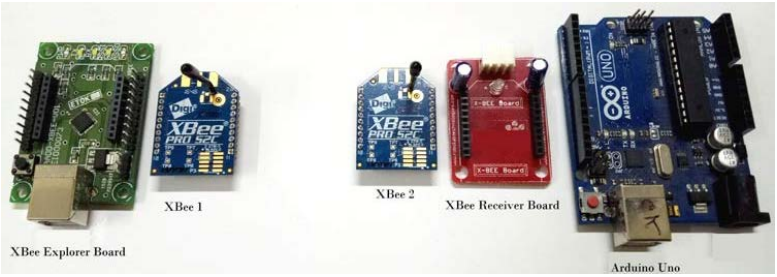


Рис. 4. Пример компонентов для системы

Управляющую программу, в целях упрощения реализации и дальнейшего развития проекта, разумно сделать в виде консольного приложения, которое будет посылать на передатчик уже готовые пакеты с информацией.

При необходимости, можно расширить сеть до внушительных размеров – каждый роутер поддерживает до 36 устройств. В таком случае, имеет смысл сделать координатор стационарным и управлять им или через удаленный доступ к компьютеру, или через облако. [3]

Если конечное устройство не оснащено пультом дистанционного управления, но имеет «кнопочный» интерфейс, можно установить внутрь устройства реле, которое при получении команд будет перемыкать в нужном месте контакт, имитируя нажатие кнопок. Однако, данное решение не подходит компактным девайсам, внутри корпуса которых нет места для установки реле. Также необходимо заранее продумать модель сети и понимать, каким образом устройства взаимодействуют между собой, так как конечное устройство может подсоединится не только к координатору, но еще и к роутеру. На рис. 5

можно видеть, как координатор связывает два роутера, причем сеть является устойчивой к потере связи – если один роутер перестает принимать пакеты напрямую от координатора, то сети перестраивается и он связывается с роутером.

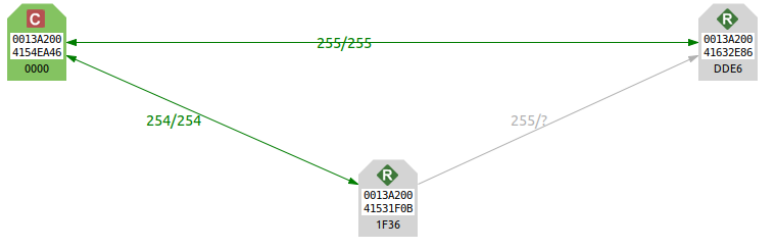


Рис. 5. Пример сети с двумя роутерами

Заключение

Данная статья посвящена разработке и реализации системы управления вентиляцией воздуха. Описаны работа сети ZigBee, основные принципы, протоколы и стандарты, технические решения для самостоятельных решений в области «умного дома», позволяющие при помощи доступных модулей и не сложной кодовой базы автоматизировать практически любую бытовую технику на дистанционном управлении.

Был проведен анализ пакетов, получаемых устройством, и их расшифровка, в ходе которой были вычислены биты, отвечающие за те или иные нажатия клавиш. В итоге данной разработке была реализована рабочая модель управления вентилятором. В дальнейшем, предполагается расширение системы и ее автоматизация, основанная на сборе данных, полученных с датчиков влажности и температуры воздуха, а также сенсоров движения, добавление в сеть еще нескольких устройств и масштабирование ее на несколько комнат.

Список литературы

1. Developing with ZBOSS for Zigbee [Электронный ресурс]. – Режим доступа : https://developer.nordicsemi.com/nRF_Connect_SDK/doc
2. Беспроводные сети ZigBee и IEEE 802.15.4 [Электронный ресурс]. – Режим доступа : http://book.itep.ru/4/41/zigbee.htm
3. Shahin Farahani ZigBee Wireless Networks and Transceivers / Shahin Farahani // Newnes; PAP/COM edition. – 2008. – P. 136-168.

Разработка шагающего робота в среде моделирования Webots

Е. П. Булавина

Студент бакалавриата

Д. И. Соломатин

Старший преподаватель

Введение

В настоящее время область робототехники стремительно развивается и внедряется как в промышленности (механизированные сборочные линии), так и в быту (роботы-пылесосы, квадрокоптеры). Роботы позволяют совершать работы, недоступные человеку, находящиеся в опасных или труднодоступных местах, автоматизировать процесс, тем самым компенсируя затраты на время. С их появлением в повседневной жизни, все больше людей начинают интересоваться робототехникой, в особенности к шагающим роботам, зачастую созданным по подобию животных, или даже самого человека. Их конструкция позволяет им передвигаться по неровным поверхностям, ступенькам, преодолевать препятствия на пути.

Но, к сожалению, создание робота достаточно затратный процесс. Стоимость материалов и деталей не позволяет свободно экспериментировать с конструкцией и алгоритмами поведения. Ошибки при сборке или проектировании могут поставить крест на всем проекте, из-за чего множество людей, которые хотели бы заниматься робототехникой, даже не приступают к ней. Поэтому, для обучения и прототипирования используются специально разработанные среды моделирования, которые позволяют в доступной форме создать и запрограммировать своего робота, не оглядываясь на недоступность деталей и свободно экспериментируя с различными формами и алгоритмами. Использование среды позволяет не только обучать робототехнике, но и сэкономить при сборке робота, из-за полноценной симуляции физики и возможности переноса программной части из виртуальной среды в реальную.

Чтобы продемонстрировать возможности сред моделирования, было принято решение создать и запрограммировать с нуля шагающего робота-гексапода в симуляторе Webots.

1. Создание модели робота в Webots

Webots – это свободно распространяемый симулятор роботов, который позволяет полностью моделировать их конструкцию, поведение, окружающую среду и взаимодействие с ней. [1]

За основу модели гексапода была выбрана форма, напоминающая строение шестиногого паука.

Моделирование в Webots представляет собой «сборку» робота из простейших частей, отвечающих за геометрию, физику, движение, внешний вид и т.д. Законченный робот представляет собой иерархическую структуру (рис. 1), в основе которой стоит класс Robot, позволяющий использовать контроллер и управлять всеми дочерними элементами из него, такими как вращательный двигатель, считывать данные с различных сенсоров. Все части модели настраиваются, так возможно изменить форму детали, настроить параметры поворота, наложить ограничения на максимально возможные углы, силу мотора, вес робота, отдельных деталей, их плотность, силу трения для различных материалов.

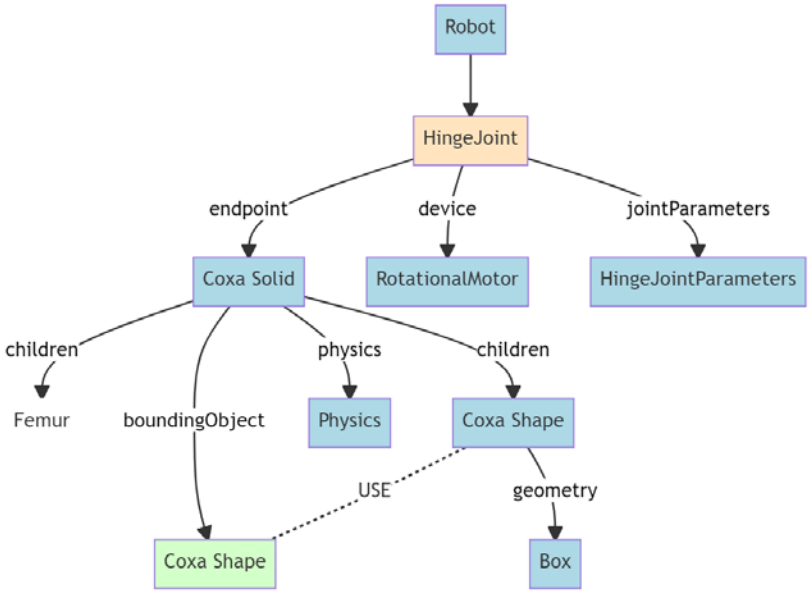
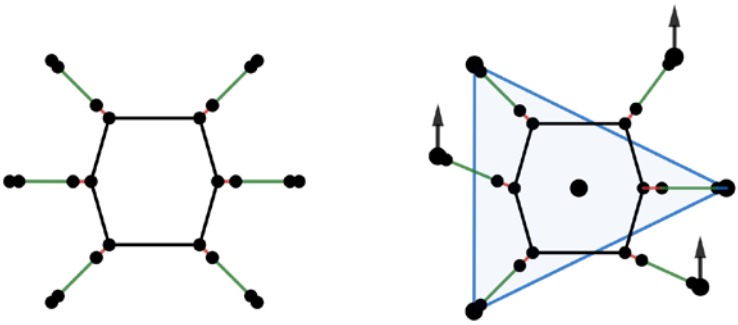


Рис. 1. Структура детали «Соха» (тазик)

2. Планирование движения

Существует несколько основных видов движения для шестиногих роботов, которые представляют собой циклическое повторение определенного шаблона для каждой из ног. Для данного проекта была выбрана модель движения «Tripod», или же «тренога», при которой для обеспечения стабильности три конечности остаются на поверхности (две на одной стороне робота – спереди и сзади, и одна по центру – на другой), выполняя роль опоры и помогая двигать робота вперед, а другие три – поднимаются и перемещаются в следующую точку. Таким образом, три конечности всегда остаются на поверхности, а из-за расположения ног центр тяжести робота всегда остается внутри площади, составляемой точками опор (рис. 2). [2]

Для того, чтобы робот не «скользил» при движении, следует сохранять дистанцию между ногами при движении и делать так, чтобы точки опор всегда оставались на одном месте. Поэтому, стоит выбирать следующее место для постановки ноги так, чтобы они двигались ровно в одном направлении при движении вперед. Или же, для поворота, перемещение ног должно осуществляться по концентрическим окружностям, описанным вокруг центра робота.



а б

а – положение спокойствия, б – движение с опорой в трех точках и сохранением положения центра тяжести

Рис. 2. Модель движения гексапода

3. Кинематика

Для перемещения конечностей к заданным позициям необходимо так рассчитать углы между частями каждой из ног, чтобы конец оказался

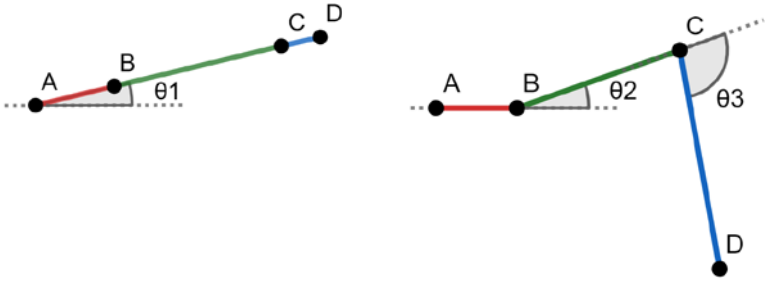
в требуемом положении. Есть два подхода для достижения данной цели – эмпирический, где углы рассчитываются экспериментально, либо же способ, использующий инверсную кинематику, который позволяет вычислить необходимые углы конструкции исходя из её размеров и конечного положения. Данный подход позволяет свободно изменять размеры частей конечности, так как вычисления останутся теми же. Таким образом, углы рассчитываются в реальном времени и положение ног не закреплено для определенного движения.

Для гексапода необходимо вычислить три угла для каждой из ног:

– θ_1 – угол, определяющий положение «Соха» (тазика), находящийся в плоскости XY (рис. 3а);

– θ_2 – угол, определяющий положение «Femur» (бедра), находящийся в плоскости XZ (рис. 3б);

– θ_3 – угол, определяющий положение «Tibia» (голени), находящийся в плоскости XZ (рис. 3б).



а б

Рис. 3. Схематическое представление конечности гексапода: а – вид сверху, б – вид сбоку

Для вычисления значений данных углов можно воспользоваться сведениями о том, что известны точка назначения (X, Y), в координатах относительно основания, высота основания конечности Z и длины каждой из частей ноги AB, BC, CD.

Следует найти проекцию BD на плоскость XY и расстояние BD, которое необходимо для вычисления вспомогательных углов:

$$L = \sqrt{X^2 + Y^2} - AB \quad (1)$$

$$BD = \sqrt{L^2 + Z^2} \quad (2)$$

Из конечных координат вычисляется угол θ_1:

$$\theta_1 = \arctan\left(\frac{Y}{X}\right) \tag{3}$$

Вспомогательные углы α, β, γ, которые отвечают соответственно за угол между диагональю BD и плоскостью XY, углом между BC и BD, а также углом между BC и CD:

$$\alpha = \arccos\left(\frac{L}{BD}\right) \tag{4}$$

$$\beta = \arccos\left(\frac{BC^2 + BD^2 - CD^2}{2 \cdot BC \cdot BD}\right) \tag{5}$$

$$\gamma = \arccos\left(\frac{BC^2 + CD^2 - BD^2}{2 \cdot BC \cdot CD}\right) \tag{6}$$

Отсюда, значения оставшихся углов θ_2 и θ_3: [3]

$$\theta_2 = \alpha - \beta \tag{7}$$

$$\theta_3 = \pi - \gamma \tag{8}$$

После их вычисления остается только передать значения на вращательные двигатели.

4. Программная реализация

Для написания контроллера в данной среде можно использовать различные языки программирования, такие как C, C++, Python, Java или Matlab. Но для данного проекта был выбран Python, как один из наиболее удобных для написания.

Исходя из вышеперечисленного, было принято решение разделить вычисления на два модуля – MotionPlanner и IKModule, первый из которых отвечает за вычисление конечной координаты, куда следует переместиться, второй же вычисляет требуемые углы для моторов. Эти два модуля объединяет Animator, который по переданным командам передает необходимые параметры для других модулей (рис. 4).

Webots обладает доступным API, который позволяет легко получать информацию с устройств и передавать значения. Так, задание угла вращательного двигателя выгладит как (листинг 1):

Листинг 1

Задание позиции вращательного двигателя

```
robot = Robot()
…
motors = […, robot.getDevice('RFC') ,…]
…
      Motors[i].setPosition(pos)
```

Робот получает команды управления с клавиатуры (листинг 2), и исходя из предыдущего положения и точки назначения, куда переместится робот, с помощью аниматора изменяет параметры, к которым должно привести движение за текущий шаг.

Листинг 2

Получение ввода с клавиатуры

```
key = keyboard.getKey()
while key > 0:
      state = getState(key)
      addMovementState(state)
      key = keyboard.getKey()
```

При этом, данные модули не являются жестко привязанными к размерностям робота и получают их из файла конфигурации, с помощью которого данный алгоритм движения можно применить к другим моделям гексаподов.

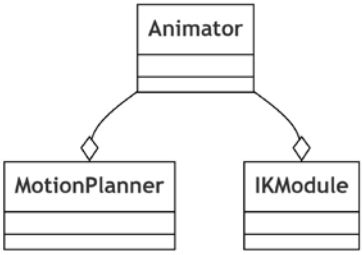
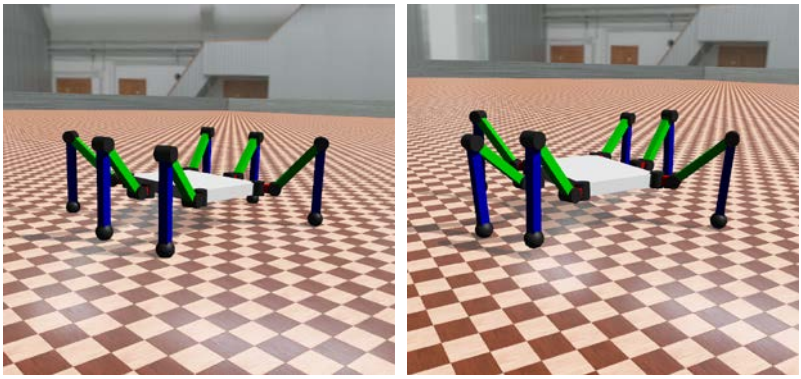


Рис. 4. Диаграмма классов, отвечающих за вычисление положения конечностей в определенный момент времени

5. Результаты

При объединении написанного контроллера и модели робота получается рабочая симуляция, которая способна передвигаться по местности с помощью команд, подаваемых с устройства ввода (рис. 5).



а б

а – начальная положение, б – в движении

Рис. 5. Симуляция гексапода

Заключение

Данная статья посвящена разработке рабочей модели гексапода (шестиногого робота) в среде моделирования Webots и написанию контроллера движения для него. Проведен анализ различных способов движения. Были приведены математические расчеты для получения требуемых значений для перемещения робота в заданную точку. Был реализован и протестирован алгоритм движения гексапода, совместимый со средой Webots, который в последующем можно применить для реальной модели.

Список литературы

1. Webots Reference Manual [Электронный ресурс]. – Режим доступа : https://cyberbotics.com/doc/reference/index

2. Chen, Z. Time-optimal trajectory planning method for six-legged robots under actuator constraints / Z. Chen, F. Gao // Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science. – 2019. – Vol. 233. – P. 4990-5002.

3. Faigl, J. Adaptive locomotion control of hexapod walking robot for traversing rough terrains with position feedback only / J. Faigl, P. Čížek // Robotics and Autonomous Systems. – 2019. – Vol. 116. – Iss. 14. – P. 136-147.

Перенос обучения при тренировке глубокой сверточной нейронной сети на основе регионов на небольшой обучающей выборке

К. И. Ганигин

Студент бакалавриата

А. В. Акимов

Ст. преподаватель

Введение

Задача детектирования на изображениях широко распространена в наше время, начиная от тривиальных задач, вроде классификации зерен на изображении, и заканчивая серьезными алгоритмами биометрической идентификации личности. Самый идеальный классификатор в таких задачах — это глаз человека. Очевидно, что такой подход будет медленным, его сложно масштабировать. Но с появлением нейронных сетей, который, по сути, эмулируют мышление человека, была разработана специальная архитектура, подобная зрению человека – Сверточные Нейронные Сети (CNN). Человек не пытается проанализировать каждый квант света, который поступает на сетчатку. На деле мы обобщаем полученную картину на интересующие нас регионы и дальше уже апеллируем ими. Такой же подход используется в CNN – изображение многократно «сворачивается», находятся некоторые характерные черты и в итоге получаем векторное представление изображения, которое в дальнейшем классифицируется.

Основная проблема использования CNN лежит не в написании архитектуры или разработке алгоритма, а в обучении детектора. Обучающая выборка сильно влияет на результативный детектор. Но бывают ситуации, когда задача подразумевает недостаток датасета. В этом случае уже стоит вопрос как максимально эффективно использовать имеющиеся данные.

В данной работе рассматривается процесс обучения детектора на изображениях при малой обучающей выборке и методы улучшения результата в таких условиях.

1. Обзор моделей семейства R-CNN

Для задачи детектирования лиц на изображениях была выбрана модель семейства R-CNN (Сверточная нейронная сеть на основе регионов).

Стоит описать, как реализуются те или иные модели детекторов R CNN в рамках нейронных сетей. По сути, модель оборачивается вокруг готовой CNN (сверточной нейронной сети). Алгоритмы модели преобразуют исходное изображение в подходящий для нее вид. Рассмотрим этот процесс на примере первой итерации моделей R-CNN.

Основой каждой модели R-CNN является алгоритм Selective Search (алгоритм выборочного поиска).

Исходное изображение делится на множество регионов (гипотез). Выделение объектов происходит на основании контрастности, перепадов яркости и т.п. (более подробное описание алгоритма выборочного поиска можно найти в [1]).

Собственно, изображения, заключенные в полученные рамки, подаются на вход CNN (рис. 1). Предварительно изображения растягивают (или сжимают) до размера входного слоя сети.

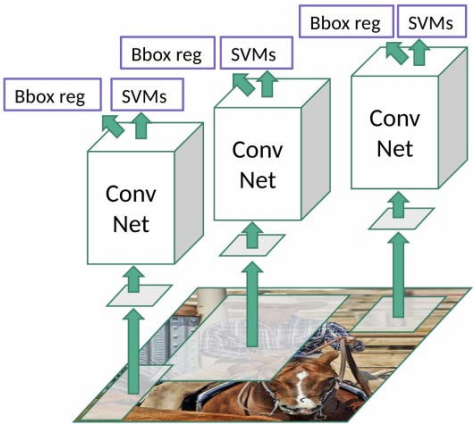


Рис. 1. Демонстрация работы R-CNN из [1].

Далее, с помощью CNN гипотеза представляется в виде вектора признаков. Их классификация является тривиальной задачей, которая решается с помощью алгоритма SVM (Метод опорных векторов).

После всех операций, на выходе мы получаем координаты рамок объектов и их принадлежность к определенному классу.

Очевидно, так подход будет слишком времязатратным. Каждый объект по отдельности будет проходить классификацию. Эта особенность ограничивает данную модель при использовании, к примеру, в задаче детектировании на видео.

Данную проблему решили в модели Fast R-CNN [3]. Теперь вместо того, чтобы каждую гипотезу передавать на вход CNN, на вход передается целое изображение (рис. 2). Достигается эта оптимизация благодаря дополнительным ROI-слоям (слой регионов интереса).

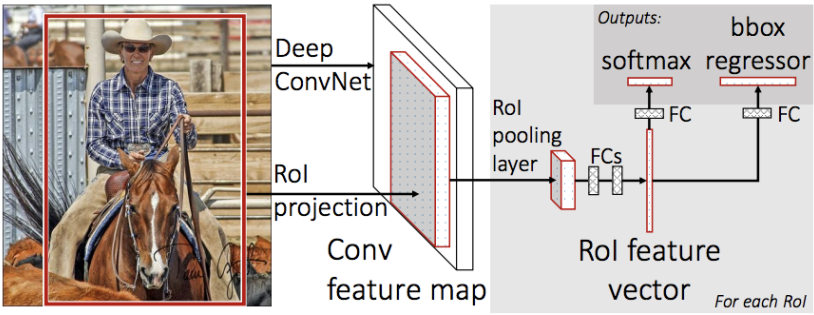


Рис. 2. Архитектура Fast R-CNN из [3].

На выходе ROI-слоя мы получаем карту признаков (feature map). По сути, алгоритм делит изображения на элементы одинакового размера и вычисляет для каждого из них значения признаков. В результате получается следующая матрица (рис. 3).

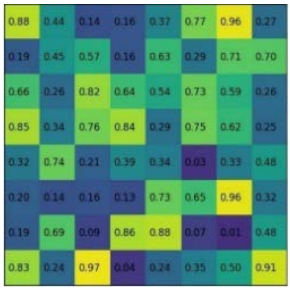


Рис. 3. Карта признаков из [4].

Эта операция заменяет собой многократное использование сверточной нейронной сети для каждой гипотезы.

Далее модель сопоставляет полученную карту признаков и координаты гипотез для дальнейшей классификации.

2. Обзор архитектур сверточных нейронных сетей. Перенос обучения.

Важнейшей частью любого детектора является сверточная нейронная сеть, которая преобразует тензор цветного изображения размером HxWx3 (H – высота изображения, W – ширина изображения в пикселях) в вектор признаков гораздо меньшего размера, пригодного для классификации.

Выбор архитектуры CNN зависит от типа задачи. Для относительно тривиальных задач (вроде классификации зерна или цифр) может подойти сеть с небольшим числом параметров. Но для задач вроде детектирования или распознавания лиц потребуется сеть больших размеров.

Иными словами, размер модели влияет на то, насколько пристально детектор будет «вглядываться» в объекты, подмечать характерные признаки и т.п.

В данной работе, для задачи детектирования лиц на изображении, были выбраны архитектуры VGG16 (138 миллионов параметров) и AlexNet (62 миллиона параметров).

В классической задаче обучения сверточной нейронной сети слоям задается случайный вес, который в процессе меняется методом обратного распространения ошибки. Но как показывает практика, при обучении детектора, веса слоев схожи, т.е. почти все детекторы пытаются найти на изображениях одни и те же признаки и процесс обучения сверточных слоев является избыточным.

Для решения этой проблемы существует понятие переноса обучения. Для этого используют предобученные сверточные нейронные сети, вроде VGG16 и AlexNet. Изначально они были обучены на множестве классов объектов, поэтому имеют подходящие для многих задач детектирования веса. За пользователем остается только модифицировать ее под свои задачи.

Модификации сети происходит довольно просто: требуется убрать последние классифицирующее слои и заменить их на подходящие для конкретной задачи (выставить необходимое число классов).

Т.о. мы получим сеть, которая знает «как» искать, но не знает «что» искать. В данной работе производилось обучение именно последних слоев.

3. Формирование обучающей выборки.

Для задачи детектирования лиц на изображениях использовалась выборка фотографий групп людей [5]. Обучение детектора подразумевает разметку каждого класса на изображении, т.е. помимо

самих изображений, требуется список, содержащий координаты рамок лиц для каждого изображения.

Очевидным решением является ручная разметка каждого изображения с помощью специального инструмента. Но для упрощения процесса, в данной работе был следующий подход.

Предварительно, на всей выборке использовался детектор MTCNN (многозадачная каскадная сверточная нейронная сеть) [6], предназначенный специально для детектирования лиц. Полученные рамки формировались в специальный объект, состоящий из списка изображений, списка рамок для каждого изображения и наименования классов этих рамок (в данной случае каждая рамка классифицировалась как лицо).

Предварительная разметка корректируется вручную. Так же добавляются рамки лиц, которые не были обнаружены MTCNN.

В результате получаем список рамок вокруг каждого лица для каждого изображения (рис. 4).

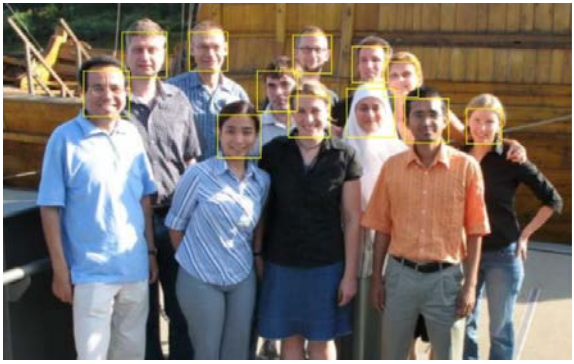


Рис. 4. Пример изображения из обучающей выборки.

Вышеописанный процесс проводился также для тестовой и валидационной выборок.

4. Анализ процесса обучения. Тестирование детекторов.

Рассмотрим детектор на основе Alexnet, обученный на выборке из 200 изображений (около 1500 лиц).

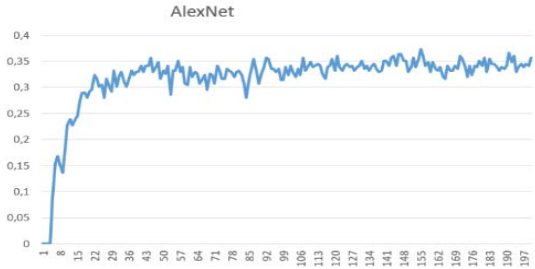


Рис. 5. Анализ работы детектора Alexnet, 200 изображений.

Здесь и далее, на подобных графиках по оси X размещается количество эпох, в течении которых обучался детектор, по оси Y – полнота.

Полнота – отношение числа лиц, которые были обнаружены верно, к общему количеству лиц.

Как видно на графике, значение параметра полноты, даже после 200 эпох, не превышает 0,4. Это является неудовлетворительным результатом – меньше половины лиц было обнаружено.

Первое и самое «дешевое» решение для улучшения результатов – аугментировать выборку.

В процессе обучения на каждое изображение применялись два вида аугментации – аффинные и цветовые (рис. 6).

Аффинные аугментации заключались в том, что изображение случайным образом поворачивалось по или против часовой стрелки в диапозоне от 0° до 30°. Так же с 50% вероятностью изображение отражалось по горизонтали.

Цветовые аугментации – менялась контрастность, яркость и т.п.

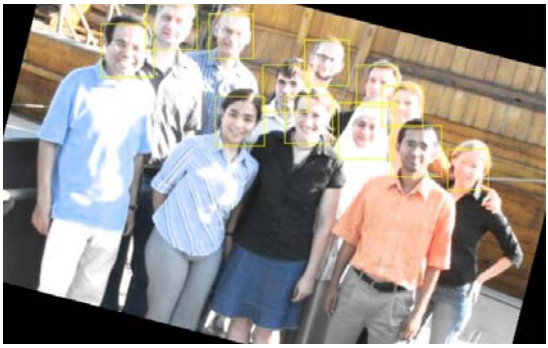
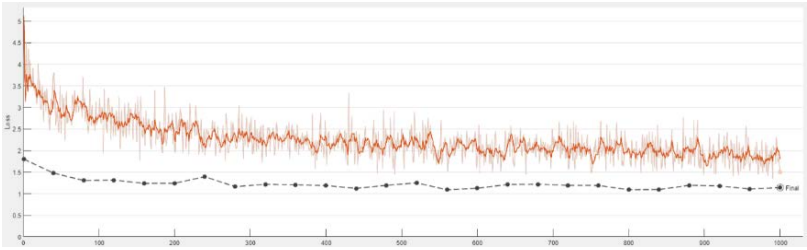
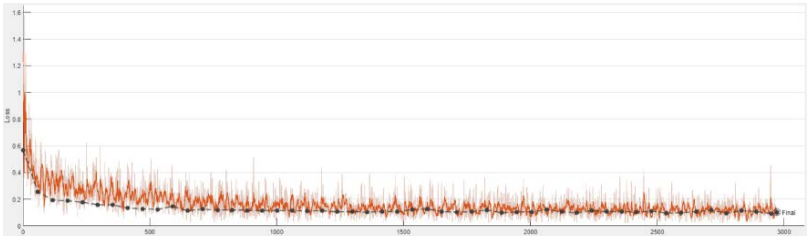


Рис. 6. Пример преобразованного изображения.

Рассмотрим процесс обучения сетей на основе VGG16 и AlexNet с аугментированной выборкой (рис. 7).



а



б

Рис. 7. График зависимости loss от количества итераций обучения AlexNet (а) и VGG16 (б)

Одним из основных параметров, на который стоит обращать внимание при обучении, является loss (значение функции потерь). Этот параметр отражает суммарную ошибку классификации объектов.

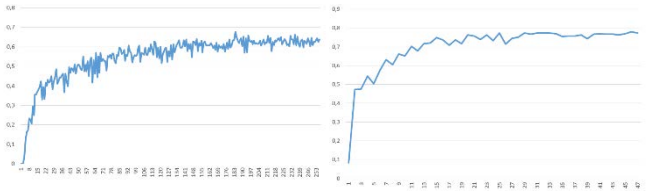
В идеальном случае, loss должен не убывать в процессе обучения. Наличие обратной ситуации говорит о проблемах (к примеру, переобучение).

Как видно на графиках, обучение проходило успешно.

Рассмотрим результаты детектирования после каждой эпохи (рис. 8).

Использование аугментаций заметно улучшило результаты. Значение показателя полноты сети AlexNet увеличилось с 0,35 до 0,67 (рис. 8а).

Замена же архитектуры на VGG16 увеличила полноту до 0,77, что не далеко от приемлемого результата (рис. 8 б).

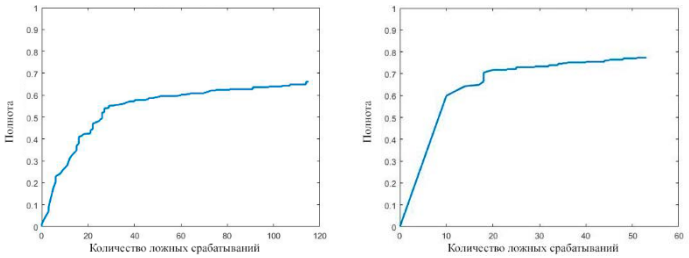


а б

Рис. 8. Анализ работы детектора AlexNet (а) и VGG16 (б), 200 изображений с использованием аугментаций

Данный способ оценки (только по параметру полноты) не является избыточным, т.к. не учитывает ложные срабатывания детектора. Использовался он лишь для поверхностного наблюдения.

Поэтому, при получении приемлемых результатов, лучшие версии детекторов анализировались с помощью ROC – кривой (рис. 9).



а б

Рис. 9. ROC кривая Alexnet (а) и VGG16 (б)

На данных графиках можно увидеть, что при одинаковом значение параметра полноты, количество ложных срабатываний детектора на основе AlexNet значительно выше, чем у детектора на основе VGG16.

Вывод

В ходе работы были проведены эксперименты по обучению Fast R-CNN детектора на малой обучающей выборке. Также, был рассмотрен процесс формирования выборки, ее аугментация.

Экспериментально была найдена зависимость выбора архитектуры CNN от результатов полученного после обучения детектора (выбор архитектуры VGG16 вместо AlexNet увеличил параметр полноты на 0,1, при этом уменьшив количество ложных срабатываний). Анализ результатов производился, как в контексте всего процесса обучения

(рис. 8), так и по его итогам при оценке финальных показателей работы полученных детекторов (рис. 9).

Были получены детекторы лиц на изображениях на архитектурах AlexNet и VGG16 с процентом верных обнаружений 67% и 77% соответственно.

Список литературы

1. Towards Data Science [Электронный ресурс]. – Режим доступа: towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e/

2. Rich feature hierarchies for accurate object detection and semantic segmentation / Ross Girshick [et al.] // CVPR. – 2014. – DOI: 10.48550/arXiv.1311.2524.

3. Girshick, R. Fast R-CNN / R. Girshick // ICCV. – 2015. – DOI: 10.48550/arXiv.1504.08083.

4. Deepsence.ai [Электронный ресурс]. – Режим доступа: deepsense.ai/region-of-interest-pooling-explained/

5. Cornell University School of Electrical and Computer Engineering [Электронный ресурс]. – Режим доступа: chenlab.ece.cornell.edu/people/Andy/ImagesOfGroups.html

6. GitHub [Электронный ресурс]. – Режим доступа: github.com/matlab-deep-learning/mtcnn-face-detection

Особенности применения технологий в области реактивного программирования

В. А. Горбушин

Студент бакалавриата

Н. К. Самойлов

Ст. преподаватель

Введение

При разработке систем считается стандартной практикой организовать процесс работы потока следующим образом – запрос к ресурсу, ожидание ответа и затем дальнейшее выполнение, при этом во время ожидания поток простаивает. Очень многое зависит от нагрузки на систему, и обычно систему масштабируют путём увеличения количества потоков. Однако, расчёт оптимального значения потоков трудновыполним из-за неравномерной нагрузки, при этом избежать простоев не получится.

Более эффективно занять ресурсы системы работой, управлять масштабированием, а также повысить отзывчивость приложения позволит концепция реактивного программирования.

Данная парадигма не является чем-то совсем абстрактным и отдалённым от мира, в чём её большой плюс. Она является логическим продолжением уже знакомых понятий, как многопоточность (свойство платформы, состоящее в том, что процесс может состоять из нескольких потоков), асинхронность (характеристика не совпадающих во времени процессов). Это значит, что этой парадигмой довольно легко овладеть разработчикам, знакомым с этими понятиями, а также она ориентирована на удовлетворение уже сложившихся потребностей.

Данная работа посвящена реактивной парадигме и особенностям реализации реактивности в рамках различных библиотек, фреймворков, а также языков программирования.

1. Предпосылки появления реактивного программирования

Некоторое время назад разработка приложения означала, что впоследствии его запуск позволит себе большее количество устройств, не только за счёт оптимизации кода, но и за счёт увеличения частоты

процессора новых устройств. В настоящий момент произошли изменения, и развитие происходит а несколько ином направлении.

На первом рисунке представлены зависимости, полученные Карлом Руппом в процессе исследования различных систем. Здесь SpecINT – это единица измерения вычислительной мощности процессора для целочисленных вычислений, предложенная корпорацией SPEC, логические ядра – это количество физических ядер, умноженное на количество потоков, которые может выполнять каждое ядро.

Как видно из графика (рис. 1), частота выпускаемых процессоров перестала расти. Произошло это потому, что техпроцессы попросту уткнулись в физический потолок.

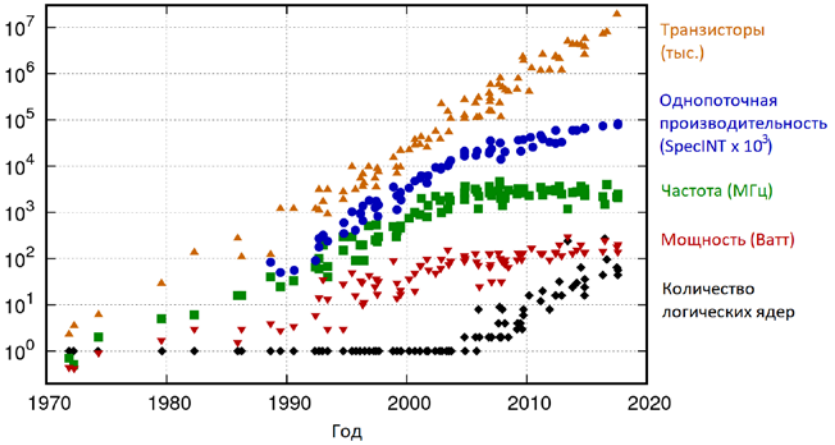


Рис. 1. Данные о тенденциях за последние 42 года

Однако это не значит, что увеличение мощности процессоров невозможно. Появились многоядерные процессоры, количество ядер можно отследить на графике, они названы «Number of Logical Cores».

Ядро – это самостоятельный вычислительный блок в архитектуре процессора, способный выполнять линейную последовательность задач за определенный период времени. Если нагрузить одно ядро несколькими последовательностями задач, то оно будет попеременно переключаться между ними, обрабатывая по одной задаче из каждого потока. В масштабах системы это приводит к замедлению работы программ и сервисов.

Поток – это выделенная область в физическом ядре процессора. Такая виртуальная реализация позволяет разделять ресурсы ядра и работать параллельно с двумя разными последовательностями команд.

Таким образом операционная система воспринимает поток, как отдельный вычислительный центр, следовательно, ресурс ядра используется более рационально, и скорость вычислений увеличивается.

Для управления возможностями ядер и потоков были выделены методы, как многопоточгость и асинхронность.

2. Многопоточность и асинхронность

Многопоточность (multithreading) – свойство платформы (например, операционной системы, виртуальной машины и т. д.) или приложения, состоящее в том, что процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть без предписанного порядка во времени. При выполнении некоторых задач такое разделение может достичь более эффективного использования ресурсов вычислительной машины.

Однако, многопоточность имеет сложности в реализации. Требование синхронизировать потоки, возникновение состояния гонки, затруднённая отладка – данные примеры демонстрируют усложнение процесса разработки многопоточной системы, при этом прирост производительности не гарантируется, это зависит от спроектированного решения.

Асинхронность – концепция программирования, которая заключается в том, что результат выполнения функции доступен не сразу же, а через некоторое время в виде некоторого, нарушающего обычный порядок выполнения, вызова. Использование кода асинхронного программирования позволяет освободить поток выполнения, из которого он был запущен, что приводит к экономии ресурсов, а также предоставляет возможность параллельных вычислений.

Технологии на основе данных понятий разрабатываются и успешно применяются. Однако, они имеют свои недостатки, которые в некоторых случаях являются критичными. Например, в многопоточной системе может возникнуть ситуация, когда потоков для обработки информации временно станет недостаточно, появятся потери. Или другой пример, когда поток будет заблокирован и находится в состоянии ожидания. В таком случае оптимальной была бы система, не ожидающая ввода, а реагирующая на события. Таким образом было образовано понятие реактивности.

3. Реактивность

Реактивное программирование – программирование с асинхронными потоками данных. Поток – это основа парадигмы,

являющаяся последовательностью событий, упорядоченных по времени. При этом в виде потока может быть представлено что угодно: переменные, пользовательский ввод, свойства, кэш, структуры данных и т.п.

Reactive Streams – это совместная инициатива выработки стандарта асинхронной обработки потоков данных. Она определяет минимальный набор интерфейсов, методов и протоколов, описывающих необходимые операции и сущности для достижения цели — асинхронной обработки данных в реальном времени с неблокирующим обратным давлением (back pressure). Допускает различные реализации, использующие разные языки программирования.

4. Технологии реактивного программирования

ReactiveX – API для Java, JS, C#, Scala, Clojure, Swift и др., мультиплатформенная реализация реактивного программирования. Отличительной особенностью является то, что работа идёт с дискретными значениями, которые образуются с течением времени.

Важной частью работы с асинхронными потоками являются реактивные операторы – функции, которые принимают один наблюдаемый (Observable) объект в качестве первого аргумента и возвращают другой наблюдаемый объект. Для каждого элемента, который испускает наблюдаемый источник, они применяет функцию, а затем испускает его в возвращённом наблюдаемом объекте. Они могут даже испускать другой Observable в наблюдаемом месте назначения. Это называется внутренней наблюдаемой.

Observer - это потребитель данных. У него есть методы, которые вызываются в зависимости от поступившего события от Observable: onNext() - вызывается для каждого элемента из потока данных, onComplete() - вызывается, если поток завершён и данные больше не будут поступать, onError() - вызывается, если произошла ошибка [1].

Также существуют множество и других типов, источников данных, например ConnectableObservable, Flowable, Single, Maybe и т.д., а также операторов, например empty(), never(), map(), buffer(), debounce() и т.д.

Для некоторых языков программирования может быть представлено несколько решений в формате реактивного программирования, например в Java асинхронный поток может быть представлен как Observable – в уже рассмотренном API ReactiveX для Java, или RxJava, Spring Reactor предоставляет Flux, в самой же спецификации Java 9 есть Publisher. Примеры возвращаемых значений можно увидеть на рис. 2.

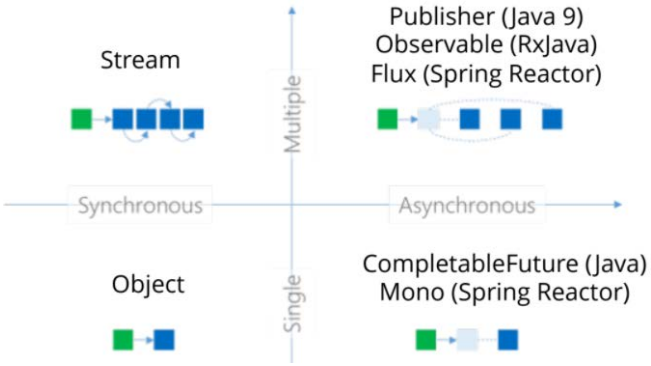


Рис. 2. Типы возвращаемых значений, с учётом синхронности и количества потоков

Node.js – платформа, которая позволяет JavaScript стать языком общего назначения, но примечательно здесь то, что в основе Node лежит событийно-ориентированное и реактивное программирование.

Есть компоненты, которые присущи реактивности – ReadableStream, WritableStream, TransformStream, то есть читаемый и записываемый поток, последний же предназначен для обработки данных.

Node.js очень быстро стал популярным и в какой-то момент вырвался вперед в нагруженных системах с большим объемом ввода-вывода. Однако подходит он не для любых систем. JavaScript – однопоточный язык (но это не значит, что там ничего нельзя делать в других потоках), вычислительные задачи для Node.js обычно очень маленькие [2].

Netty – это серверная среда ввода-вывода. Главной её особенностью является то, что она неблокирующая (NIO) и асинхронная, а значит её удобно использовать с Node.js или Spring Web Flux.

Основой является Channel, который представляет собой открытое соединение, способное выполнять операции ввода-вывода, такие как чтение и запись.

Netty использует парадигму приложений, управляемых событиями, поэтому конвейер обработки данных представляет собой цепочку событий, проходящих через обработчики. Базовым интерфейсом для обработчиков событий канала являются ChannelHandler.

ObservableComputations – это библиотека C#, которая позволяет производить вычисления над INotifyPropertyChanged и

INotifyColectionChanged объектами (вычисления включают в себя те же вычисления, что и в LINQ). Ближайшими аналогами являются следующие библиотеки: Obtics, OLinq, NFM.Expressions, BindableLinq, ContinuousLinq.

Akka Streams – реализация Reactive Streams, позволяющая избежать программирования низкого уровня, некоторых трудностей с акторами, связана с Akka.NET и C#.

В рамках реактивности есть такая аббревиатура, как FRP, которая расшифровывается, как Functional Reactive Programming. Считается, что функциональная парадигма считается наиболее естественным базисом, т.к. в данной парадигме зачастую нет переменных в привычном понимании, вместо этого необходимо оперировать функциями (в математическом представлении).

Яркими примерами FRP являются язык Elm и фреймворк Reflex языка Haskell.

Elm – это функциональный язык программирования для создания реактивных веб-приложений. Приложения на Elm обязаны следовать The Elm Architecture (TEA) – простому паттерну проектирования, который подразумевает разделение кода на три части: model, view и update: Model – это данные, которые используются в приложении, а также данные, описывающие сообщения, необходимые для любой интерактивности. View – это функция, преобразующая model в пользовательский интерфейс. Update – это функция, ответственная за обновление состояния [3].

Reflex – это фреймворк, позволяющий создавать реактивные веб-приложения на Haskell. В отличие от Elm, Reflex не накладывает строгих ограничений на архитектуру приложения. Фреймворк предоставляет абстракции для управления состоянием, а программист волен использовать их, комбинируя произвольным образом. Основных абстракций три, это Event, Behavior и Dynamic.

Есть решения в стиле FRP, такие как Mrr и Bacon (связаны с JavaScript), reactive-banana (Haskell).

Заключение

Данная статья описывает задачи и идеи, которые впоследствии привели к появлению реактивной парадигмы программирования.

В ходе статьи были также рассмотрены различные технологии, с указанием их достоинств и недостатков.

Каждая из технологий, при правильном применении может дать заметный прирост производительности разрабатываемой системы.

Список литературы

1. Morgillo, I. Grokking ReactiveX / I. Morgillo, F. Chignoli, S. Sekulic. – New York: Manning Publications, 2007. – 30 p.

2. Дейли, Б. Разработка веб-приложений с помощью Node.js, MongoDB и Angular: исчерпывающее руководство по использованию стека MEAN / Б. Дейли, Б. Дейли, К. Дейли. – Санкт-Петербург: «Диалектика-Вильямс», 2020. – 656 с.

3. Seven More Languages in Seven Weeks: Languages That Are Shaping the Future / B. Tate [et al.]. – Raleigh: Pragmatic Bookshelf, 2014. – 291 p.

Прототипирование осциллографа на базе микроконтроллера

Ф. А. Десятириков
Е.А.Саков
В. Г. Деревянко

Студенты бакалавры

С. А. Зуев

Доцент

Введение

Техника осциллографирования получила чрезвычайно широкое распространение при измерениях, настройке и испытаниях во многих областях науки и техники. Осциллограф во многих случаях является основным прибором, без которого невозможны ремонт и настройка радиоаппаратуры. Он позволяет непосредственно на экране наблюдать форму исследуемого сигнала и производить необходимую оценку его параметров. Для одновременного исследования нескольких электрических сигналов применяют многолучевые или многоканальные осциллографы. Осциллографы широко используют для исследования не только электрических, но и неэлектрических величин, предварительно преобразованных в пропорциональные им электрические сигналы.

Основным достоинством осциллографа, обеспечившим ему широкое распространение, является наглядность изображения на экране исследуемого процесса, особенно полезная при исследованиях и измерениях быстропеременных, импульсных и других кратковременных процессов.

Таким образом, при исследованиях в области телевидения, радиолокации, радиоастрономии и ряде других областей радиотехники и радиоэлектроники осциллограф является незаменимым прибором. Однако, профессиональный осциллограф является достаточно дорогим для многих любителей электроники, в связи с этим постоянно предпринимаются попытки создать данную систему своими руками. Наиболее перспективным в этом отношении является прибор, созданный на базе платы Arduino. В данной статье рассматриваются особенности создания такого осциллографа.

1. Постановка задачи

Осциллограф — регистрирующий прибор, предназначенный для исследования (наблюдения, записи, измерения) амплитудных и временных параметров электрического сигнала, подаваемого на его вход, и наглядно отображаемого непосредственно на экране, регистрируемого на фотоленту либо записываемого в память.

Сфера применения осциллографов достаточно велика. С помощью этого прибора проводят измерения напряжения, тока, мощности, полного сопротивления, параметров катушек индуктивности, фазы и частоты, модуляции и её видов, а также исследование импульсов, электронных приборов, частотных и фазовых характеристик приемников и усилителей, проводятся магнитные исследования. Кроме того, осциллографы применяются для исследования и измерения неэлектрических величин, например, давления и коротких промежутков времени. При помощи осциллографа можно контролировать качество поверхности и форму деталей и изделий. В настоящее время широко используются электронно-лучевые (аналоговые) и цифровые осциллографы.

Периодические сигналы произвольной формы эффективно исследуются электронно-лучевыми осциллографами, параметры которых за последнее время достигли высокого уровня. Блок-схема простейшего электроннолучевого осциллографа включает: электронно-лучевую трубку, блок питания, блок временной развертки, который предназначен для перемещения светящегося пятна по экрану трубки с определенной (регулируемой) скоростью. Это перемещение обычно производится по горизонтальной оси, которая носит название «оси времени» или оси X. Если одновременно с этим заставить пятно перемещаться перпендикулярно к оси времени так, чтобы это перемещение было пропорционально измеряемой величине, то прочерчиваемая светящимся пятном на экране трубки кривая (ее называют осциллограммой) позволит судить не только о самой измеряемой величине, но и о законе ее изменения во времени. Ось, в направлении которой перемещается пятно за счет исследуемой величины, носит название «оси явления» или оси Y. Можно несколько изменить способ получения осциллограммы, заставив пятно перемещаться по оси X не в зависимости от времени, а в зависимости от какого-либо другого параметра. Кроме перечисленных трех блоков применяются также усилители напряжения в канале X и в канале Y, а также делители, обеспечивающие снижение напряжения. Современные осциллографы снабжаются рядом дополнительных устройств, предназначенных главным образом для обеспечения удобств в

эксплуатации прибора и повышения точности измерений. Управление процессом регистрации электрического сигнала посредством данного типа осциллографа производится в ручном режиме. На рис. 1 приведены органы управления разверткой (блок HORIZONTAL), синхронизации (блок TRIGGER) и тракта вертикального отклонения (блок VERTICAL).

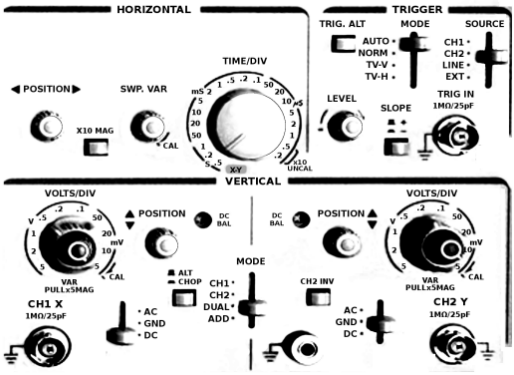


Рис. 1. Органы управления передней панели осциллографа

Ещё один тип осциллографа — цифровой, в котором аналоговый исследуемый сигнал сразу во входном блоке оцифровывается с помощью аналогово-цифрового преобразователя (заменяется последовательностью цифровых эквивалентов) и запоминается в дискретной памяти. Зафиксированный в памяти исследуемый процесс может быть затем использован любым образом, в том числе и для отображения формы процесса. Запоминание цифровых эквивалентов мгновенных значений исследуемого сигнала намного упрощает задачу измерения и обработки параметров сигнала.

Получение и хранение информации в цифровом виде позволило в цифровых осциллографах (ЦО) перейти к новому типу индикаторов – плоским матричным экранам. Дискретность экрана при этом естественным образом согласуется с дискретной формой представленной информации [1]. Вместе с этим цифровой осциллограф допускает возможность полной автоматизации систем управления в аспекте реализации функции контроля и диагностики [2, 3].

Указанные несомненные преимущества данного типа осциллографа – возможность записи сигнала, математической обработки, многоканальная регистрация и др. — способствуют постепенному вытеснению аналоговых осциллографов цифровыми.

Таким образом, целью нашей работы было – разработать проект портативного цифрового осциллографа с учетом минимизации его перспективной стоимости, а также произвести прототипирование найденных решений.

Объектом исследования являлась цифровая платформа Arduino. Предметом исследования являлось изучение возможности реализации портативного осциллографа. Методы исследования — системный анализ, инструментальный, модели и методы программирования.

2. Сервисы платформы Arduino

Arduino — это платформа для быстрой разработки электронных устройств и экспериментов в области электроники, робототехники, автоматики и автоматизации процессов. Платформа удобна, имеет простотой язык программирования, а также открытую архитектуру и программный код, благодаря чему пользуется огромной популярностью во всем мире. Устройство программируется без использования USB программаторов.

Микроконтроллер программируется при помощи языка Arduino и среды разработки Arduino. Язык программирования представляет собой язык C++ с фреймворком Wiring, он имеет отличия в написании кода с некоторыми особенностями. Облегчает написание работающей программы, имеющийся комплект библиотек Arduino, включающий в себя объекты и функции.

Программирование целиком ведется через собственную программную оболочку Arduino IDE. В этой оболочке имеется текстовый редактор, менеджер проектов, препроцессор, компилятор и инструменты для загрузки программы в микроконтроллер. Оболочка написана на основе проекта Processing на Java, работает под Windows, Mac OS X и Linux.

Основанные на Arduino проекты устройств могут работать самостоятельно, или взаимодействовать с программным обеспечением на компьютере. Платы могут быть куплены в сборе или собраны пользователем самостоятельно. Программная часть состоит из бесплатной программной оболочки (IDE) для написания программ, их компиляции и программирования аппаратуры. Аппаратная часть представляет собой набор смонтированных печатных плат, которые продаются официальным производителем, либо сторонними производителями.

На базе Arduino устройства могут получать информацию посредством различных датчиков об окружающей среде, а также могут управлять различными исполнительными устройствами.

3. Разработка осциллографа на плате Arduino

В настоящее время актуальной является задача импортозамещения в электронике и программной инженерии. С этой целью была предпринята попытка отработки заместительной программной реализации для Arduino UNO R3 при решении задачи для создания портативного осциллографа. Таким образом, объектом исследования являлось автоматизированное электронное измерительное устройство. Предметом исследования являлось улучшение технической компоновки устройства и обеспечение его программным кодом. Реализация прототипа устройства состояла из нескольких этапов.

На первом этапе был произведён выбор функциональности будущего устройства. Основное внимание было уделено функциям просмотра формы сигнала в реальном времени и возможности изменения времени заполнения буфера прорисовки.

На втором этапе для реализации выбранного на первом этапе функционала были сформулированы требования к органам управления и устройству вывода. Эти требования сводились к следующему:

1. Переключение режимов буфера.
2. Сброс текущего показателя буфера.
3. Прорисовка данных буфера.

На третьем этапе работы анализ различных вариантов решения показал, что оптимальная архитектура соответствует представленной на рис. 2.

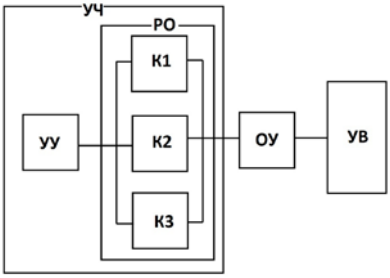


Рис. 2. Архитектура устройства

Здесь: УУ — устройство управления. УЧ — управляющая часть, РО — регулирующий орган, ОУ — объект управления (измерительная система), УВ — устройство вывода, S — измеряемый сигнал.

Управляющая часть УЧ содержит 3 кнопки: масштабирование по времени в сторону увеличения (К1), то же — в сторону уменьшения (К2), сброс (К3). Устройство вывода информации о сигнале S

представляет собой OLED экран с диагональю 0.97 дюйма. ОУ реализована на плате Arduino UNO R3 на базе микроконтроллера ATmega328p с тактовой частотой 16 МГц.

Четвёртый этап состоял в написании кода для оцифровки сигнала в измерительной части системы, а также для вывода результатов на дисплей.

Средой разработки выбрана Arduino IDE на основании следующего: открытость, простота, интуитивно понятная поддержка выбранного микроконтроллера. Программный код написан на языке C++. Использован фреймворк Arduino. Задача прорисовки данных на дисплей решена с помощью библиотеки Adafruit_SSD1306. Для низких значений времени прорисовки кадра используются регистры прескейлера микроконтроллера. Их значения устанавливаются согласно нормативным параметрам.

Ниже представлен фрагмент программного кода, реализующий настройку прескейлера на платформе Arduino для заданного режима в процессе управления.

```
 if (mode < 7) {
  // if mode < 7 we using hardware ADC prescaler
  if (mode & 1)
  sbi(ADCSRA, ADPS0);
  else
  cbi(ADCSRA, ADPS0);
  if (mode & 2)
  sbi(ADCSRA, ADPS1);
  else
 cbi(ADCSRA, ADPS1);
  if (mode & 4)
  sbi(ADCSRA, ADPS2);
  else
  cbi(ADCSRA, ADPS2);
 } else {
  sbi(ADCSRA, ADPS0);
  sbi(ADCSRA, ADPS1);
  sbi(ADCSRA, ADPS2);
 }
```

Данный программный продукт является структурной составляющей технической реализации цифрового осциллографа.

4. Прототипирование цифрового осциллографа

Реализация найденных решений доведена нами до уровня прототипа цифрового осциллографа.

В результате разработки осциллографа нами проведено прототипирование полученного устройства по методике, представленной в работе [4]. Корпус был создан в программе Siemens

NX 10. Программа использует ядро геометрического моделирования Parasolid. Зная размеры аккумулятора, и имея в свободном доступе чертеж платы Arduino UNO, сначала был создан чертеж самого корпуса. Далее в программе была выбрана плоскость для создания в ней чертежа прямоугольника, вытягиванием получен параллелепипед. В плоскости, перпендикулярной начальной, были сделаны отверстия под разъёмы. В начальной плоскости созданы опоры, на которых должна держаться плата. После этого была загружена 3D-модель платы и в «сборке» прикреплена к корпусу, выполнены неровности под плату. После этого файл сохранен в формате .stl и распечатан на 3D-принтере.

На рис. 3 представлен общий вид разработанного изделия в корпусе и показано внутреннее устройство портативного осциллографа (для иллюстрации масштаба приложена бытовая спичка).

Рис. 3. Прототип цифрового осциллографа

На входе осциллографа предусмотрен разъём BNC (показан справа), который предназначен для подключения измерительного щупа с делителем.

Для питания осциллографа в корпус установлен литий-ионный аккумулятор со стабилизатором напряжения.

Заключение

Статья посвящена разработке портативного осциллографа, реализованного на основе Arduino UNO R3 на базе микроконтроллера ATmega328p с тактовой частотой 16 МГц. Наш осциллограф не

использует внешний аналогово-цифровой преобразователь, а только плату Arduino.

В результате предложено техническое решение задачи по оцифровке сигнала и выводу его на OLED экран в реальном времени.

Написан программный код (на языке C++ в среде разработки Arduino IDE), реализующий заданный функционал.

Данный осциллограф не является профессиональным инструментом, однако он хорошо подойдет для образовательных целей и использования в учебной лаборатории при изучении на 3 курсе основ электроники студентами ФКН.

В дальнейшем предполагается усовершенствование разработанного осциллографа с заменой используемого экрана на жидко-кристаллический экран 4,7 дюйма фирмы Taidacent.

Список литературы

1. Дьяконов, В. П. Современная осциллография и осциллографы / В.П. Дьяконов. — М.: Солон-Пресс, 2005. — 320 с.

2. Akimov, V. I. Development of Statistical Models of Systems and Automation Tools for Modeling and Forecasting Technological Processes / V. I. Akimov, A. V. Polukazakov, S. A. Zuev, F. A. Desyatirikov // Proc. 2022 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering(2022ElConRus) (St. Petersburg and Moscow, Russia, 25-28 Jan. 2022). – IEEE, 2022. – P. 82-86.

3. Кийко, В.В. Программная оболочка PSpice control shell, программа моделирования Pspice и программный осциллоскоп probe / В.В. Кийко. — Екатеринбург: УПИ-УГТУ, 2001. — 36 с.

4. Desyatirikov, F.A. Metalens Concept for Focusing Wi-Fi Radiation / F.A. Desyatirikov, Yu. V. Khripunov // Proc. 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (2020 ElConRus) (St. Petersburg and Moscow, Russia, 27-30 Jan. 2020). – IEEE, 2020. – P. 105-107.

Разработка математического и программного обеспечения для автоматизации процесса распознавания текста на сложном фоне

Н. В. Дешин

Студент бакалавриата

М. А. Дрюченко

Доцент

Введение

Одной из проблем в современном мире, является скорость получения информации. Одна из причин этого - недоступность старых данных в цифровом формате. Для ввода данных в электронную таблицу или цифровой документ требуется ручной труд. Если данные имеют определенную структуру, к примеру, заданы в табличном формате, на это может уходить десятки минут, что в рамках тысяч документов превращается в месяцы работы. Изображение документа занимает гораздо больше места для хранения, чем тот же документ в цифровом формате. Широкое распространение цифровых камер, средств оцифровки и сканирования привело к активному развитию методов детектирования и распознавания объектов на изображениях. Текстовая информация по-прежнему является наиболее надежным признаком для сопоставления, классификации и описания изображений. Тонны архивных бумаг, чеков и счетов проходят сканирование и оцифровку во многих отраслях: в розничной торговле, логистике, банковских услугах и т.п. Компании получают конкурентное преимущество, если быстро оцифровывают и находят нужную информацию. Оцифровка позволила нам автоматически заполнять юридические документы и заявки на услуги, а также открыла доступ к аналитике фискальных чеков, отслеживанию динамики цен и суммарных трат.

В настоящее время наиболее успешные системы оптического распознавания символов (OCR-системы) распознают тексты на изображениях, составленных из стандартных шрифтов, не подвергнутых шуму и искажениям, с достаточно высокой точностью.

Основная задача OCR-систем заключается в назначении фрагменту изображения текста соответствующей символьной информации.

Основные требования к OCR-системам включают возможности сохранения форматирования исходных документов: шрифтов, абзацев, таблиц, графиков и изображений.

1. Типовая схема обработки изображений и распознавания текста

Общий алгоритм работы OCR-систем на произвольных изображениях приведен на рисунке 1. Первым шагом является отделение объектов переднего плана от фона. Далее происходит определение типа выделенных областей. Если найденная область является текстом, то можно приступать к ее сегментации и распознаванию.

Верхнеуровневая последовательность обработки изображения стандартного документа выглядит так:

1. Поиск области первого плана
2. Предварительная обработка и фильтрация изображения
3. Выявление структуры найденного блока текста, определение порядка чтения.
4. Сегментация текста на строки/слова и символы.
5. Получение признакового описания каждого символа.
6. Распознавание отдельных символов.
7. Словарная проверка.

Первым этапом является поиск областей текста, в результате которого на исходном изображении выделяются текстовые области.

Алгоритмы улучшения качества предполагают предварительное сглаживание изображения, определение типов присутствующих шумов и их устранение.

Сегментация заключается в разбиении текста на строки, слова и символы. Поиск строк, как правило, основывается на периодичности и регулярности текстовых областей и осуществляется на основе метода Хафа, метода связных компонент, анализа горизонтальных, вертикальных и диагональных гистограмм.

Алгоритмы получения признакового описания символов разделяются на 2 класса: анализирующие исходное изображение символа в качестве признака и анализирующие вычисляемые признаки, такие как длина хорды, аппроксимированные контуры, остовы символов. Для снижения размерности пространства признаков часто применяется метод главных компонент или линейный дискриминантный анализ.

Для распознавания символов реализуются различные классификаторы, основанные на нейронных сетях и наиболее распространенных на сегодняшний день свёрточных нейронных сетях.

Словарная проверка осуществляется на основе стандартных или динамически созданных языковых словарей, N-грамм, реализованных в виде списков, деревьев или графов.

Рис. 1. Схема обработки изображений и распознавания текста

2. Предварительная растровая обработка

Процесс обработки изображений состоит из ряда этапов, среди которых одним из наиболее важных является предварительная обработка изображений, которая представляет собой самостоятельный процесс. Предварительная обработка изображений отвечает за фильтрацию шумов, помех и искажений на изображении. К задачам предварительной обработки изображений относятся:

- подавление шумов;
- изменение диапазона яркостей;
- изменение резкости;
- коррекция цветов;
- коррекция геометрических искажений.

Задачей предварительной обработки изображения является улучшение качества изображения. Методы предварительной обработки

изображения могут существенно различаться в зависимости от того, каким путем изображение было получено — синтезировано системой машинной графики, либо путем оцифровки черно-белой или цветной фотографии.

3. Бинаризация изображения

Сегментации можно разделить на две большие группы:

1. С глобальным порогом
2. Метод Оцу
3. С адаптивным порогами

Методы с глобальным порогом достаточно просты в реализации и быстро решают поставленную задачу, но их минус состоит в том, что для определения области объекта выбирается один (глобальный) порог, потому, в идеале, текст должен быть написан или напечатан с использованием цвета на контрастном фоне. Другими словами, если текст более темный по цвету, предполагается, что фон будет более светлого цвета (например, черная доска с написанием мелом) и наоборот (например, текст на белой бумаге). Но, в реалиях, изображение может иметь неравномерное освещение и шум, что негативно сказывается на качестве распознавания, потому стоит обратиться к методам с локальными (адаптивными) порогами – в данных методах изображение разделяется на подобласти, для каждой из которых вычисляется свой локальный порог.

Метод Оцу, названный в честь Нобуюки Оцу, используется для выполнения автоматической пороговой обработки изображений. Он автоматически определяет оптимальный порог для изображения. Этот порог определяется путем минимизации дисперсии интенсивности внутри класса или, что эквивалентно, путем максимизации дисперсии между классами. Это

эквивалентно глобально оптимальному алгоритму k-средних, выполняемому на гистограмме интенсивности. Этот метод использует как простой, так и адаптивный порог, имея единое пороговое значение для всего изображения и удаляя нежелательный фон, который может возникнуть в адаптивном пороге.

Его применение зависит от входного изображения. Мы должны использовать адаптивное определение порога или пороговое значение Оцу, в зависимости от того, что лучше подходит для текущего изображения.

Примеры работы бинаризации изображения можно рассмотреть на рисунках 2, 3.

Рис. 2. Оригинальное изображение

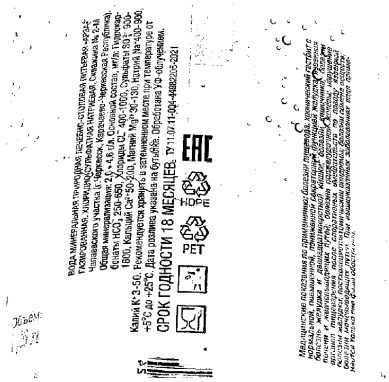


Рис. 3. Сегментация с глобальным порогом результата изоляции тёмных регионов

4. Распознавание символов

Для сегментации можем использовать различные алгоритмы сегментации:

– С использованием метрик;
– Обучаемые алгоритмы распознавания

В случае использования метрик у нас есть словарь, где описаны характеристики символов, мы идём по тексту и сверяем метрики. Данный способ относительно простой, но ввиду того, что встречаются похожие символы или буквы с симметрией, данный способ может выдавать плохие результаты, поэтому предпочтительно использовать обучаемые алгоритмы распознавания, в данном случаи используются нейронные сети, происходит предварительное обучение на выборках.

5. Словарная проверка

Есть вероятность, что при работе модель может ошибаться. Нужно исключить такие моменты или заранее предупредить пользователя о том, что распознавание не получилось, и попросить переснять документ. В этом нам помогает простая процедура постобработки, которая может проверять на предсказание только ограниченного словаря для конкретного поля. Например, для числовых полей выдавать только число.

Другим примером постобработки являются поля с ограниченным набором значений, которые подбираются по словарю на основе редакторского расстояния. Проверка на допустимость значений: в поле даты рождения не могут быть даты 18 века.

Заключение

Данная статья посвящена разработке и реализации программного обеспечения для автоматизации процесса распознавания текста на сложном фоне. Описаны алгоритмы предварительной обработки изображения, позволяющие избавиться от шумов и сегментировать текст. Использование данного решения позволяет определять текст на

Список литературы

1. Болотова, Ю. А Обзор алгоритмов детектирования текстовых областей на изображениях и видеозаписях / Ю. А. Болотова, В. Г. Спицын, П. М. Осина // Компьютерная оптика. – 2017. – Т. 41, № 3. – С. 441-452.

2. О локализации плоских объектов на изображениях со сложной структурой проективных искажений / Д. В. Тропин [и др.] // Информационные процессы. – 2019. – Т. 19, № 2. – С. 208-229.

3. Фисенко, В. Т. Компьютерная обработка и распознавание изображений : учебное пособие / В. Т. Фисенко, Т. Ю. Фисенко ; СПбГУ ИТМО, 2008. – 192с.

Применение генеративно-состязательных нейронных сетей в задаче гибридного сжатия данных

А. С. Железной

Студент бакалавриата

М. А. Дрюченко

Доцент

Введение

В век современных технологий, в связи с широким распространением систем передачи данных и их использования для решения различных прикладных задач возникает необходимость разработки новых и повышения эффективности известных алгоритмов стеганографии и гибридного сжатия данных, посредством генеративно-состязательных нейронных сетей. По сей день происходит усовершенствование техники, повышаются характеристики фото- и видеокамер, изображения получаются более плавными, а качество картинки беспрестанно растёт. Но, как известно, с повышением качества картинки происходит рост массы файлов, которые хотелось бы передавать как можно быстрее, при этом не сильно теряя их качество.

Многие современные приложения используют алгоритмы сжатия изображения с потерями для их эффективного хранения. Ключевым моментом в данных системах гибридного сжатия данных является сохранение изображения с потерями и с минимальными визуальными изменения в оригинале. Существует достаточно методов и алгоритмов, позволяющих осуществлять операции такого рода, но сжатие с потерями на основе глубокого обучения, где нейронная сеть напрямую оптимизирована для компромисса между скоростью и изображением, приводит к новым современным методам, вызывающих большой интерес.

В данной работе я проанализирую различные методы и подходы к применению генеративно-состязательных нейронных сетей в задачах стеганографии и гибридного сжатия данных, рассмотрю преимущества и недостатки и познакомлю с совершенно новыми методами.

1. Алгоритм RLE для сжатия данных

Этот алгоритм давно известен и весьма распространён. Идея алгоритма: группа подряд идущих пикселей одного цвета кодируется парой (цвет: количество пикселей). Пример работы алгоритма представлен на рис. 1.

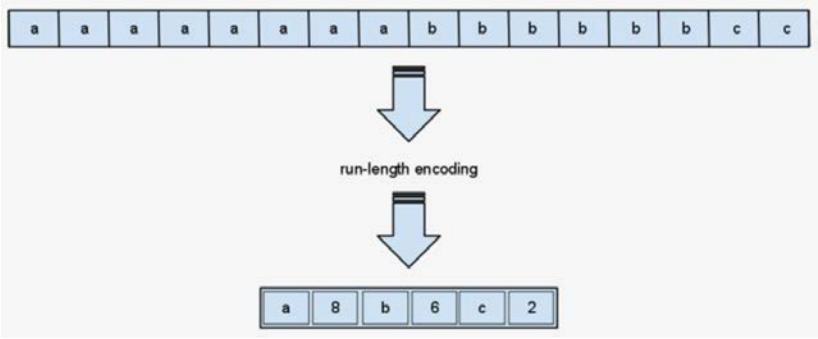


Рис. 1. Алгоритм RLE (Run-Length Encoding)

Особенности реализации: 3 байта отводится для хранения цвета. Для хранения количества подряд идущих пикселей такого цвета может быть отведено 1 или 2 байта. Если количество подряд идущих пикселей одного цвета меньше, чем 27, то используется только 1 байт, первый бит которого равен 0. В противном случае используются оба байта, причём первый бит первого из этих байтов равен 1, а второй байт является младшим. Таким образом, максимальное число пикселей в группе, кодируемой одной парой (цвет : количество пикселей) составляет 215 – 1 [1].

Преимущества:

1. довольно высокий коэффициент сжатия для изображений;
2. работает очень быстро, так как в процессе кодирования / декодирования не используются никакие дополнительные структуры данных.

Недостатки: если при прямом обходе пикселей изображения часто чередуются цвета, причём даже в том случае, когда набор цветов ограничен (например, текст), эффективность сжатия резко падает.

2. Сдвиговый алгоритм для сжатия данных

Идея алгоритма: если при прямом обходе пикселей изображения незадолго до текущего пикселя встречался пиксель такого же цвета, то 3 байта, кодирующие цвет пикселя, можно заменить на 1- или 2-байтовую ссылку на пиксель с таким же цветом, а точнее – указать, на сколько

байтов нужно сдвинуться назад относительно текущего байта, чтобы получить нужный цвет [1].

Для ускорения работы алгоритма как при кодировании, так и при 27 декодировании используется хэш-таблица. При кодировании ключом хэш-таблицы является цвет, а значением – номер байта последнего просмотренного пикселя с таким цветом. Если цвет встретился впервые или количество байтов до предыдущего пикселя с таким же цветом превышает 28 – 1, то байт ссылки = 0, а за этим байтом следуют 3 байта, хранящие значение цвета, иначе указывается только ссылка [1].

При декодировании ключом хэш-таблицы является номер начального байта последнего просмотренного пикселя с таким цветом, а значением – цвет.

Преимущества: получаемый при кодировании формат хорошо поддаётся дальнейшему кодированию другими алгоритмами.

Недостатки:

1. даже в наилучшем случае степень сжатия невелика;
2. медленно работает из-за обращения на чтение, а затем на запись к дополнительным структурам данных для каждого пикселя.

3. Гибридный алгоритм для сжатия данных

Данный алгоритм обладает высокой трудоёмкостью, демонстрирует высокую степень сжатия изображений, является органическим соединением алгоритмов RLE и сдвигового алгоритма. 2 формата представления закодированной информации объединяются в один следующим образом: (ссылка : цвет : количество пикселей). При этом количество байтов, отведённых для хранения ссылки, цвета и количества пикселей остаётся таким же, как и в вышеописанных алгоритмах. В процессе кодирования и декодирования производятся все те же действия, что и в предыдущих двух алгоритмах вместе взятых [2].

Гибридный алгоритм обладает всеми преимуществами RLE. При его использовании достигается дополнительное сжатие за счёт применения идей сдвигового алгоритма. Гибридный алгоритм не наследует недостатки сдвигового алгоритма. Недостаток (1) устраняется за счёт сжатия алгоритмом RLE. А недостаток (2) присутствует в значительно меньшей степени, так как обращение к дополнительным структурам данных происходит не для каждого пикселя, а для каждой группы. За счёт применения идей сдвигового алгоритма также устранён недостаток алгоритма RLE – в случае частого чередования цветов при прямом обходе пикселей изображения в условиях ограниченного количества этих цветов степень сжатия значительно выше, чем при сжатии RLE.

4. Принцип работы генеративно – состязательных сетей

Генеративно-состязательная нейросеть – архитектура, состоящая из генератора и дискриминатора.

Дискриминационные алгоритмы – классифицируют входные данные. Учитывая особенности полученных данных, они стараются определить категорию, к которой они относятся [3].

Генеративные алгоритмы заняты обратным. Вместо того, чтобы предсказывать категорию по имеющимся образам, они пытаются подобрать образы в данной категории [3].

Предположим, мы пытаемся сгенерировать рукописные цифры, подобные тем, что имеются в наборе данных. Цель дискриминатора – распознать подлинные экземпляры из набора.

Генератор получает рандомное число и возвращает изображение. Это сгенерированное изображение подается в дискриминатор наряду с потоком изображений, взятых из фактического набора данных. Дискриминатор принимает как реальные, так и поддельные изображения и возвращает вероятности, числа от 0 до 1, причем 1 представляет собой подлинное изображение и 0 представляет фальшивое.

Пример работы GAN с изображениями представлен на рис. 2.

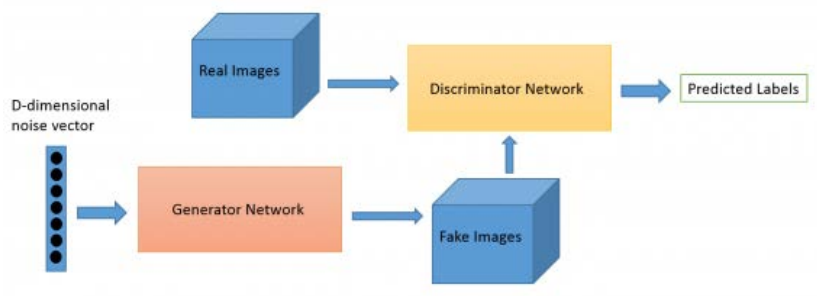


Рис. 2. Работа GAN с изображениями

5. Генеративно-состязательные сети в задачах гибридного сжатия данных. Метод HiFiC

Исследователи из Google Research проэкспериментировали с архитектурами GAN для сжатия изображений. Разработчики сравнили виды нормализации, стратегии обучения, функции потерь и архитектуры генератора и дискриминатора. Предложенная отобранная модель, по результатам сравнения, более предпочтительна даже в

случае, если прошлые подходы используют битрейт в два раза выше. Подход можно применять для изображений в высоком разрешении [4].

На рис. 3 представлена структура модели, которая состоит из четырех компонентов:

- кодировщик,
- генератор,
- вероятностная модель,
- дискриминатор.

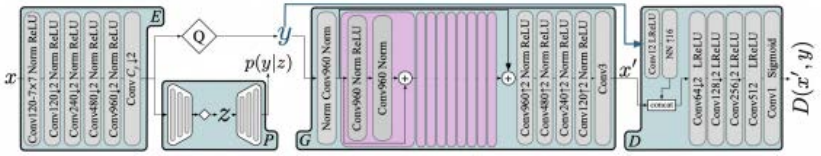


Рис. 3. Структура модели HiFiC

На рис. 4 представлено сравнение метода:

Рис. 4. Сравнение метода HiFiC с оригиналом

Мы можем видеть, что модель GAN производит высокоточную реконструкцию, которая очень близка к входу, в то время как популярный формат BPG демонстрирует блокирующие артефакты, которые все еще присутствуют при удвоенном битрейте.

Также, было произведено сравнение модели с современными методами и оценка её качественным и количественным способами,

представленными на рис. 5. Количественная оценка проводилась с помощью метрик FID, KID, NIQE, LPIPS, PSNR, MS-SSIM.

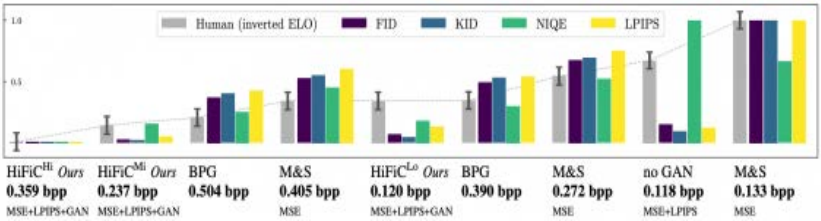


Рис. 5. Результаты оценок модели

6. Практическое применение

Применение генеративно-состязательных сетей в задачах гибридного сжатия данных может иметь множество направлений развития. Возможные применения:

- популярные мессенджеры (более сильное и качественное сжатие данных по сравнению с существующими алгоритмами);
- создание защищённого превью, с помощью которого можно будет обмениваться информацией и картинками максимально безопасно;
- создание полноценного мобильного приложения.

Заключение

Данная статья посвящена рассмотрению и обозреванию современных алгоритмов сжатия данных, а также выделил их достоинства и недостатки. В современном мире, генеративно-состязательные нейронные сети стали популярным инструментом для обучения нейронных сетей, разработки современных методов, что я и попытался рассказать в последней главе. В дальнейшем предполагается создание программы по сжатию изображений и созданию по ним защищённых превью-изображений, в ходе которой будет проведён детальный анализ более нового алгоритма сжатия данных, который будет применим в современных реалиях.

Список литературы

1. Дружинин, Д. В. Гибридный алгоритм сжатия изображения. Сравнение алгоритмов сжатия изображений. [Текст] / Д. В. Дружинин // Информационные технологии и математическое моделирование: Материалы VI Международной научно-практической конференции (9 – 10 ноября 2007 г.). – Томск: Изд-во Том. ун-та, 2007. – С. 70–73.

2. Сэломон, Д. Сжатие данных, изображения и звука. [Текст] / Сэломон Д. ; М.: Техносфера, 2004. – 368 с.

3. Генеративно-состязательная нейронная сеть [Электронный ресурс]. – Режим доступа: https://neurohive.io/ru/osnovy-data-science/gan-rukovodstvo-dlja-novichkov/

4. Fabian Mentzer, George Toderici, Michael Tschannen, Eirikur Agustsson High-Fidelity Generative Image Compression [Электронный ресурс]. – Режим доступа: https://hific.github.io/

Рециркулятор воздуха своими руками

Н. И. Иващенко

Студент бакалавриата

С. А. Зуев

Доцент

Введение

В последние несколько лет по миру гуляет эпидемия COVID-19 и, не взирая на все предпринятые меры, статистика заражений совсем не стремится к нулевой. Причиной этому служит множество факторов: эксплуатация одноразовой медицинской маски дольше положенного срока, пренебрежение дезинфицирующими средствами, контакт слизистых лица с грязными руками. Здесь перечислены нарушения личной гигиены, но как же быть с вдыхаемым воздухом? Ведь многие вирусы, бактерии и даже споры некоторых видов плесени настолько малы, что их не увидишь невооруженным глазом. Что же, теперь не дышать? Вовсе нет, ультрафиолетовыйрециркулятор воздуха позволит вам не думать о том, что вы вдыхаете. В зависимости от мощности, он способен обеззараживать воздух практически в промышленных масштабах.

Основным достоинством ультрафиолетовогорециркулятора, обеспечившим ему широкое распространение, является простота изготовления и доступность комплектующих.В связи с этим,рециркуляторыполучили широкое распространение в быту.

Помимо этого,при исследованиях в области бактериологии, вирусологиии заболеваний, передающихся воздушно-капельным путем ультрафиолетовыйрециркуляторвоздуха,является незаменимым прибором.Однако, стоить такой аппарат при розничной покупке может до нескольких десятков тысячрублей,в связи с этим, постоянно предпринимаются попытки создать его своими руками. Наиболее перспективным в этом отношении является прибор, созданный на базе бактерицидных безозоновыхлампфирмы Philips. В данной статье рассматриваются особенности создания такого ультрафиолетового рециркулятора.

1. Постановка задачи

Ультрафиолетовый рециркулятор—уникальное устройство, которое помогает обеззаразить и очистить воздух. Устройство нейтрализует вредные микроорганизмы, бактерии, вирусы и тем самым защищает от различных опасных болезней. Изначально приборы этого вида применялись в больницах, клиниках, разнообразных медучреждениях, но в связи с нестабильной эпидемиологической ситуацией, сейчас их стали использовать в школах, детских садах, различных общественных местах. Часто бактерицидные рециркуляторы приобретают для дома и квартиры.

Сфера применения Ультрафиолетовый-рециркулятора достаточно велика. В медицине выделяют целый комплекс мер по профилактике и предупреждению инфекций. И одним из приоритетных направлений в профилактике является обеззараживание воздуха, а также медицинского инвентаря бактерицидными установками (рециркуляторами) — источниками ультрафиолетового излучения. Эффективность ультрафиолетовых бактерицидных рециркуляторов обусловлена спецификой воздействия ультрафиолета на живые организмы. Болезнетворные микробы уничтожаются на молекулярном уровне в результате нарушения оболочки клетки и повреждения молекулы ДНК. В результате патогенный микроорганизм погибает.

В пищевой отрасли, к которой можно отнести не только точки общепита, но и, к примеру, хлебопекарни, находящиеся в зоне повышенного риска, появления и распространения очагов инфекционных заболеваний. Одним из ведущих факторов, который может привести к появлению очага— является порча сырья и готовой продукции.

Порче способствует появление плесневых грибков. Продукты их жизнедеятельности зачастую являются токсичными для живых организмов. Заплесневелые продукты способны вызвать отравление и аллергию у человека. Кроме того, заплесневевшая, например, буханка хлеба в хлебопекарне может стать очагом распространения грибка по всему цеху. Поэтому так важно не только бороться с ней, но главное — вовремя предупредить ее появление

Ультрафиолетовые рециркуляторы подразделяются на облучатели открытого типа, закрытые рециркуляторы воздуха и комбинированные. Первые являются источником открытого ультрафиолета, поэтому их можно использовать только при отсутствии человека, животных и растений. Закрытые рециркуляторы, наоборот, не являются источниками жёсткого ультрафиолетового излучения, поэтому их можно использовать в присутствии людей. Комбинированные сочетают в себе преимущества

облучателей и рециркуляторов, т.к. они могут быть как источником открытого, так и закрытого ультрафиолета, в зависимости от режима работы.

Также их применяют для профилактики простуды и гриппа в школах, офиса и домах.

В настоящее время широко используются именно рециркуляторы. Можно выделить три их основных вида: настенные, передвижные, напольные. Выбирать конфигурацию следует исходя из соображений удобства и безопасности. Если помещение предполагает частое или постоянное присутствие детей, пожилых людей, домашних животных, которые могут уронить и разбить прибор, лучше всего выбрать настенный вариант. Напольный рециркулятор имеет смысл приобрести, если в помещении есть безопасное место с хорошей вентиляцией. Передвижная модель подойдет тем, кто планирует использовать один рециркулятор в разных помещениях.

Рис. 1. Виды ультрафиолетовых рециркуляторов

Таким образом, целью нашей работы является разработка проекта ультрафиолетового рециркулятора с учетом минимизации его

перспективной стоимости, а также производство и прототипирование найденных решений.

2. Выбор комплектующих

Основа нашего и любого другого ультрафиолетового рециркулятора - бактерицидная лампа (не путайте с кварцевой). В отличие от кварцевой, такая лампа не выделяет большое количество ядовитого озона, являющегося сильнейших окислителем. Поскольку мы собираемся бороться с вирусами, которые передаются воздушно-капельным путем, нас следует разобраться как они взаимодействуют с ультрафиолетом.

На первом этапе был произведён выбор функциональности будущего устройства. Основное внимание было уделено выбору необходимоймощности излучения лампы для борьбы с наиболее часто встречающимися вирусами и бактериями.

Для удобства понимания, разделим вирусы на четыре группы в соответствии с типами нуклеиновых кислот:

1. Могут содержать в себе РНК с одной цепочкой;
2. ДНК с одной цепочкой;
3. РНК с двумя цепочками;
4. ДНК с двумя цепочками.

Чун-ЧиеЦенг и Чжи-Шан Ли — Тайваньские ученые, провели исследование, в котором определили необходимую [4] дозу облучения ультрафиолетом для каждой из групп вирусов. Эксперименты проводились на специально подготовленной, содержащей вирусы, биоаэрозоли. Таким образом они имитировали ситуацию, в которой человек начал чихать или кашлять. Оказалось, что вирусы, содержащиеся в микроскопических капельках, полностью теряют свою активность при определенной дозе облучения.

Наиболее чувствительными к ультрафиолету оказались вирусы, содержащие одну цепочку РНК (таблица 1, к ним также относится коронавирус). Исходя из полученных данных можно определить, что эффективная доза облучения должна быть не менее 1000 мкВт(сек/см2), а при дозе в 2000 мкВт(сек/см2) погибнут даже самые стойкие вирусы. 90% микобактерий туберкулеза погибает при дозе 576-1080 мкВт(сек/см2)[5].При такой дозе не выживут 50%бактерий и спор плесени, но из-за фотохимической реакции оставшиеся 50% замедлят свой рост до нескольких суток.

Таблица 1

Необходимая доза облучения каждой из групп вирусов

	Одна цепочка РНК	Одна цепочка ДНК	Две цепочки РНК	Две цепочки ДНК
90%	339-423 $\mu Вт \frac{сек}{см^2}$	444-494 $\mu Вт \frac{сек}{см^2}$	662-863 $\mu Вт \frac{сек}{см^2}$	910-1196 $\mu Вт \frac{сек}{см^2}$
99%	803-909 $\mu Вт \frac{сек}{см^2}$	974-1031 $\mu Вт \frac{сек}{см^2}$	1388-1771 $\mu Вт \frac{сек}{см^2}$	1906-2005 $\mu Вт \frac{сек}{см^2}$

На втором этапе, для реализации выбранного на первом этапе функционала, были сформулированы требования к выбору длины волны ультрафиолетового излучения.

В качестве источника ульрафиолета чаще всего используют лампы низкого давления с длиной волны 254 нанометра, именно они лучше всего разрушают ДНК и РНК. Также в спектр таких ламп попадает значительная часть ультрафиолета с длиной волны 185 нанометров. Такой ультрафиолет сильней всего воздействует на кислород содержащийся в воздухе и образует из него озон, который имеет сильное бактерицидное действие. При быстром совместном использоании ламп с длиной волны 254 и 185 нанометров достигается лучший бактерицидный эффект по сравнению с другими распространенными методами, однако озон ядовит, поэтому используются бактерицидные лампы с подавленной волной в 185 нм.

На третьем этапе работы анализ различных вариантов решения показал, что оптимальная архитектура соответствует представленной на рис. 2.

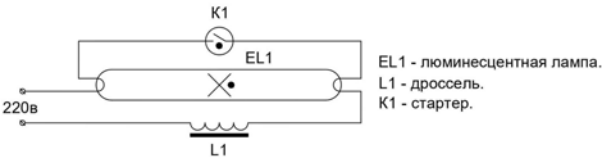


Рис. 2. Электрическая принципиальная схема устройства

3. Прототипирование ультрафиолетового рециркулятора

Реализация найденных решений доведена нами до уровня изготовления прототиповультрафиолетового рециркулятора.

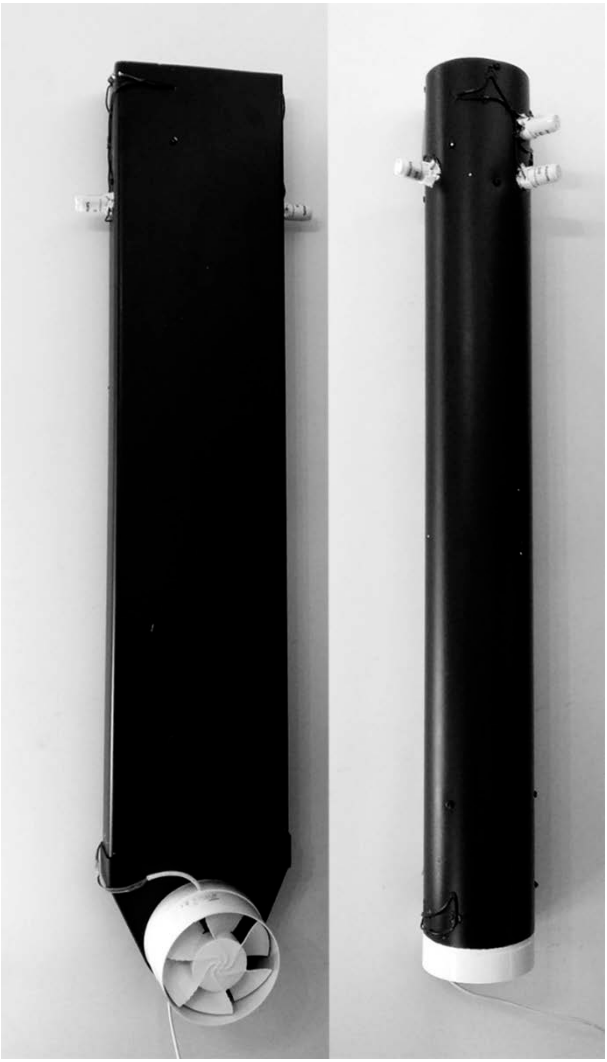


Рис. 3. Прототипы ультрафиолетового рециркулятора

Все элементы устройства расположены в корпусе. Корпус должен быть металлическим или же оклеен отражающей пленкой изнутри, провода тоже следует защитить от прямого ультрафиолетового излучения, так как оно разрушающе действует на пластмассы, резины и ПВХ. Далее были сделаны отверстия под соединительные провода. В начальной плоскости проделано крепежное отверстие, за которое прибор будет держаться на стене. Внутрь корпуса монтируются крепления ламп, на которые позже крепятся (2-3) лампы Philips TUV мощностью 30 Вт. На каждую лампу приходится по одному стартеру типа S10 и дросселю (балласту) типа L-26 фирмы Helvar и два патрона для каждой лампы. Соединяем детали прибора следуя схеме, изображённой на рис. 2 (для двухлампового варианта потребуется собрать две схемы, для трёхлампового — три).

Для того, чтобы прогонять воздух возле лампы потребуется вентилятор, обратим внимание на то, что скорость воздуха в рециркуляторе не должна быть слишком высокой для лучшего обеззараживания воздуха. На рис. 3 представлен общий вид разработанных прототипов.

Были изготовлено несколько прототипов рециркуляторов с двумя бактерицидными лампами в прямоугольном корпусе и трёхламповый вариант в цилиндрическом корпусе. Площадь облучаемой поверхности составила 5000 см2 двухлампового прибора и 7850 см2 трёхлампового.

Доза облучения двухлампового прибора составила 4 000 мкВт (сек/см2), а трёхлампового 4 800 мкВт (сек/см2). Этой дозы достаточно для полного уничтожения любых видов микробов. Измерения, проведённые в одном из корпусов университета в аудитории 307 П, показали, что после двух часов работы двухлампового варианта рециркулятора в воздухе не обнаруживаются следы болезнетворных микроорганизмов.

Установка прибора предпочтительна на высоте 1,5 м над уровнем пола. Во время работы на лампу запрещается смотреть незащищенными глазами — это может привести к ожогу роговицы.

Заключение

Статья посвящена разработке ультрафиолетового рециркулятора воздуха на основе лампы Philips TUV мощьностью 30 Вт. Наш рециркулятор можно эксплуатировать какв присутствии людей, так и в безлюдном помещении, поскольку лампа не выделяет озон во время своей работы и находится в закрытом от прямого контакта с незащищённым глазом корпусе.

В результате предложено техническое решение задачи по выбору комплектующих и их компоновке.

Все детали собраны в работающую цепь, бесперебойное функционирование прибора протестировано в учебной аудитории.

Данный рециркулятор справляется с задачей очистки воздуха наравне с профессиональными приборами. Объем обрабатываемых воздушных масс может быть увеличен или уменьшен изменением скорости вентилятора и количества ламп в рециркуляторе.

В дальнейшем планируется усовершенствование прибора путем помещения его основных комплектующих в металлический полностью светонепроницаемый корпус.

Список литературы

1. Kowalski, W. UVGI disinfection theory [Электронный ресурс] / W. Kowalski //Ultraviolet germicidal irradiation handbook. – Springer, Berlin, Heidelberg, 2009. – P. 17-50. – Режим доступа : https://doi.org/10.1007/978-3-642-01999-9_2

2. Tseng, C. C. Inactivation of viruses on surfaces by ultraviolet germicidal irradiation [Электронный ресурс] / C. C. Tseng, C. S. Li // Journal of occupational and environmental hygiene. – 2007. – V. 4. – №. 6. – P. 400-405.. – Режим доступа : https://doi.org/10.1080/15459620701329012

3. Beggs, C. B., Avital E. J. Upper-room ultraviolet air disinfection might help to reduce COVID-19 transmission in buildings: a feasibility study [Электронный ресурс] / C. B. Beggs, E. J. Avital //PeerJ. – 2020. – V. 8. – P. e10196. – Режим доступа : https://doi.org/10.7717/peerj.10196

4. Tseng, C. C. Inactivation of virus-containing aerosols by ultraviolet germicidal irradiation / C. C. Tseng, C. S. Li // Aerosol Science and Technology. – 2005. – V. 39. – №. 12. – P. 1136-1142.

5. Whalen J. J. Environmental control for tuberculosis; basic upper-room ultraviolet germicidal irradiation guidelines for healthcare settings guide [Электронный ресурс] / J. J. Whalen // – 2009 — 87 p. – Режим доступа : https://stacks.cdc.gov/view/cdc/5306

Разработка платформы для видеоконференций

А. С. Колесов

Студент бакалавриата

Д. И. Соломатин

Старший преподаватель

Введение

Видеоконференцсвязь уже прочно вошла в повседневную жизнь и используется так же часто, как и телефония. При этом видеоконференцсвязь имеет множество преимуществ сравнении с телефонными звонками и личными встречами. Видеоконференции являются довольно востребованным средством для проведения учебных лекций и занятий. Особенно они стали популярны во время пандемии коронавируса.

Однако, не все из популярных сейчас платформ для видеоконференций удовлетворяют необходимым условиям: большинство из них имеют ограничения по бесплатному использованию и не соответствуют всем требованиям для проведения учебных занятий. Например, большой проблемой является отсутствие возможности подключения и управления курсором и клавиатурой студента - известные решения вроде AnyDesk используются только для соединений двух участников, и для учебного процесса не подходят.

1. Постановка задачи

Целью данной курсовой работы является реализация автономной платформы для проведения видеоконференций с возможностью удаленного управления.

2. Обзор WebRTC

WebRTC [1] – проект с открытым исходным кодом, предназначенный для организации передачи потоковых данных между браузерами или другими поддерживающими его приложениями по технологии точка-точка. Эта технология хороша тем, что позволяет устанавливать связь между пользователями, используя только браузер. WebRTC не требует установки дополнительных плагинов. Обычно он используется для создания видеоконференций, но этим не

ограничивается - технология позволяет передавать файлы любого формата и текстовые сообщения.

Для использования WebRTC необходимо реализовать signaling server – сервер, через который клиенты будут обмениваться служебными сообщениями (SDP offer/answer, ICE Candidate).

В рамках трансляции видео с помощью WebRTC применяют следующие подходы:

– Mesh. Клиенты подключаются каждый к друг-другу. Основным недостатком этого метода является большая нагрузка на клиентов, так как при процессе сжатия видео требует много вычислительной мощности.

– MSU. Клиент поддерживает всего 1 соединение с сервером. Сервер берет на себя задачу мультиплексирования видео. При таком подходе значительно экономится трафик в сети, но при этом сервер требует огромных затрат по ресурсам.

– SFU. Способ, при котором участник отправляет свое видео на сервер, а сервер рассылает его всем другим участникам. Это самый оптимальный и гибкий способ, который подойдет для решения нашей задачи.

3. Архитектура приложения

Система реализована в виде клиент-серверной архитектуры. Серверная часть состоит из следующих компонент:

– Backend-app - приложение, реализующее всю бизнес логику, в том числе и процесс сигналинга.

– Kurento [2] – медиасервер, который выполняет задачу соединения и обработки видео пользователей. Видео/аудио потоки проходят через pipeline(конвейеры), с помощью которых к ним могут быть применены различные фильтры. В рамках этой работы был использован фильтр для сжатия клиентского видео. Для взаимодействия с данным медиасервером поставляются официальные библиотеки для различных языков программирования.

– Mongo – документоориентированная база данных, в которой хранятся все данные системы.

– Coturn – STUN/TURN сервер, необходимый для работы протокола WebRTC.

Для удобства разработки все вышеперечисленные компоненты backend были упакованы в Docker, благодаря чему серверную часть можно быстро развернуть в облачном сервисе с минимальной конфигурацией. Диаграмма развертывания приложения указана на рис. 1.

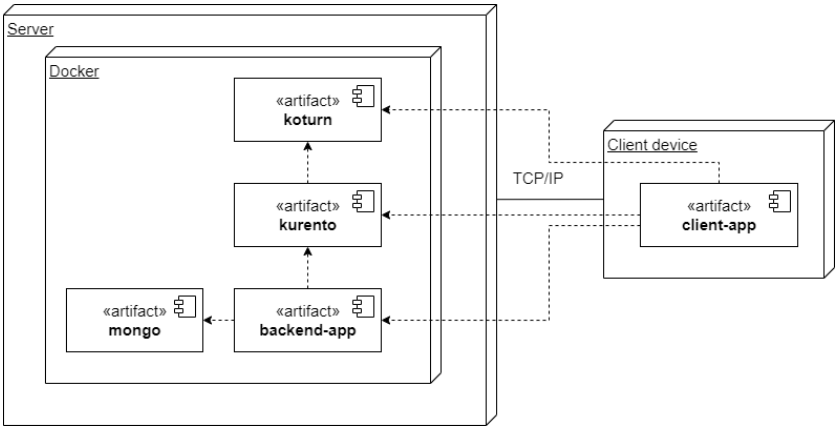


Рис. 1. Диаграмма развертывания приложения

4. Серверное приложение

Серверное приложение написано на java с использованием фреймворка Spring Boot. Для взаимодействия с клиентом имеет REST API и WebSocket API.

REST API используется для следующих задач:

– Регистрация, авторизация и управление пользователями;
– Создание и управление комнатами.

WebSocket API необходим для обслуживания соединений пользователей на стриме, а также передач команд для удаленного управления экраном.

Процесс взаимодействия клиента с сервером во время подключения к видеоконференции указан на рис. 2.

5. Клиентское приложение

Для реализации клиентской части используется фреймворк Angular с применением Electron.

Electron [3] - фреймворк, разработанный GitHub. Он позволяет разрабатывать нативные графические приложения для операционных систем с помощью веб-технологий, комбинируя возможности Node.js и библиотеки веб-рендеринга Chromium. С его помощью можно легко написать приложение с использованием HTML, CSS и JavaScript, которое «из коробки» будет совместимо с Mac, Windows и Linux.

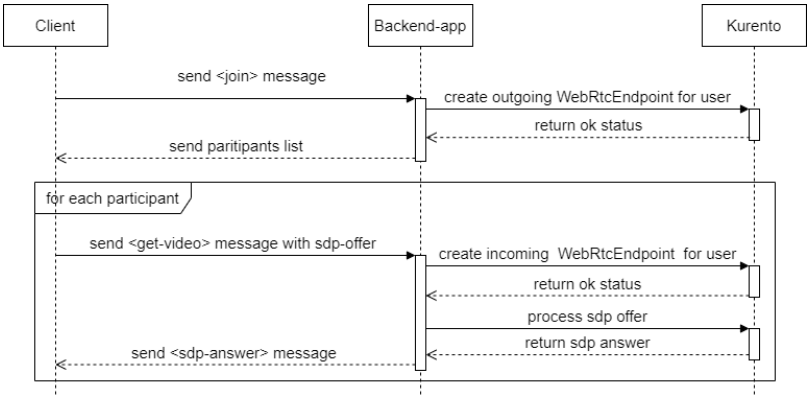


Рис. 2. Процесс подключения клиента к видеотрансляции

Управление устройством клиента производится с помощью библиотеки RobotJS. Клиент принимает от сервера команды управления, и интерпретируя их, производит перемещение мыши или нажатие клавиш.

Рассмотрим некоторые из реализованных страниц приложения.

– Страница регистрации. Используется создания профиля для нового пользователя приложения. Необходимыми для регистрации данными являются имя пользователя, его адрес электронной почты и пароль. Станица регистрации изображена на рис. 3.

– Страница трансляции. На этой странице можно общаться с участниками конференции, а также управлять их экранами. По умолчанию на странице отображаются все участники (рис. 4). Также у пользователя есть возможность выбрать конкретного участника и развернуть его видео на весь экран; при этом у пользователя появляется возможность управлять экраном (рис. 5).

Рис. 3. Страница регистрации

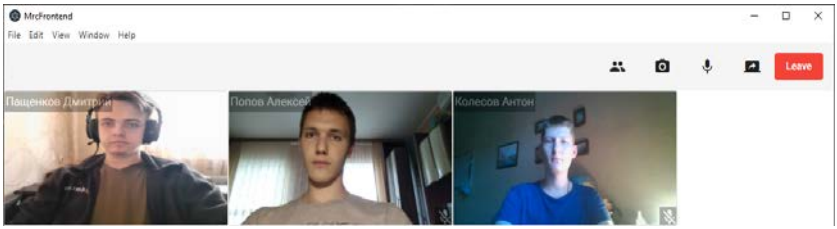


Рис. 4. Страница трансляции

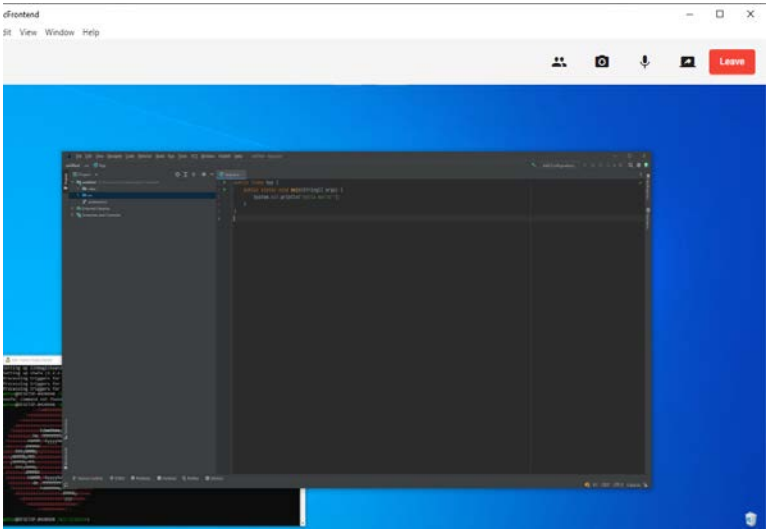


Рис. 5. Управление экраном участника

Заключение

Данная статья посвящена разработке и реализации платформы для проведения видеоконференций. Описаны используемые алгоритмы и технологии. В итоге данной разработки получилась функционирующая платформа, реализованная в виде веб-сервера и клиента. Одними из особенностей данной платформы является независимость от глобальных сервисов, а также быстрота развертывания благодаря использованию Docker. В дальнейших планах рассматривается создание API для интеграции данной системы в образовательные порталы и применение ее для учебного процесса.

Список литературы

1. WebRTC [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/WebRTC
2. Официальный сайт проекта Kurento [Электронный ресурс]. – Режим доступа: https://www.kurento.org/
3. Документация по Electron [Электронный ресурс]. – Режим доступа: https://www.electronjs.org/docs/latest

Реализация проекта «ToDo List» в рамках курса дополнительного образования «Основы промышленного программирования» Лицея Академии Яндекса

А В. Порядин

Студент бакалавриата

Е. А. Копытина

Старший преподаватель

Введение

Одной из насущных проблем настоящего времени, с которой сталкивается практически каждый человек, является проблема грамотного распределения времени. Люди, надеясь на свою память, просто забывают о делах, задачах, которые необходимо выполнить в определенный срок, опаздывают на важные встречи, пропускают мероприятия. В таких случаях человек рискует как потерять работу, так и быть отчисленным из высшего учебного заведения из-за опозданий или пропусков. Проект, выполненный в рамках курса дополнительного образования «Основы промышленного программирования» Лицея Академии Яндекса, о котором говорится в [1,2], и описываемый в данной статье, является крайне актуальным и востребованным, так как он способен решить столь серьёзную в наше время проблему, проблему тайм менеджмента, которая рассмотрена в [3,4].

1. Постановка задачи

К описываемому в статье приложению «ToDo List» были выдвинуты следующие требования:

- Аутентификация пользователя.
- Возможность создания, редактирования и удаления задачи с датой выполнения;
- Возможность просмотра статистики выполненных задач.
- Назначение задаче приоритета.
- Поиск определенной задачи.

2. Реализация логики

Приложение написано на объектно-ориентированном языке Python, который изучается на протяжении двух лет в Лицее Академии Яндекса.

Для реализации были использованы следующие библиотеки:

– Flask,
– Datetime,
– Itertools,
– Flask_wtf,
– Sqlalchemy,
– PIL,
– Marshmallow.

Разработанное приложение имеет клиент-серверную архитектуру. Клиент и сервер взаимодействуют посредством обмена HTTP-запросов.

Серверная часть представлена следующими модулями, содержащимися в папках todolist/tasks, todolist/users и todolist:

– _init_.py,
– forms.py, где расположены классы форм регистрации, логина,
– routes.py, где реализована логика регистрации/логина, загрузка задач из/в БД, логика поиска задач,
– config.py,
– db_session.py служит для инициализации и подключения БД,
– models.py, где реализованы классы задач, пользователя.

Также разработан модуль, запускающий приложение – run.py. Подобный подход описан в [5].

В основном модуле routes.py были реализованы функции-запросы, взаимодействующие с клиентской частью. Например, функция add_task(form), которая создает «задачу» и сохраняет ее в БД.

Клиентская часть представлена следующими JavaScript скриптами, содержащимися в папке todolist/static/scrpits:

– today.js,
– upcoming.js.

При их реализации была применена библиотека для работы с БД – sqlalchemy.

Также были разработаны следующие представления пользовательского интерфейса, содержащиеся в папке todolist/templates:

– base.html,
– dashboard.html,
– edit_task.html,
– index.html,
– login.html,
– main.html,
– projects.html,
– register.html,
– upcoming_tasks.html,

– update_account.html.

И CSS стили, содержащиеся в папке todolist/static/css - styles.css

3. Реализация интерфейса

Работа с программой начинается с авторизации пользователя. Вид окна авторизации представлен на рис. 1.

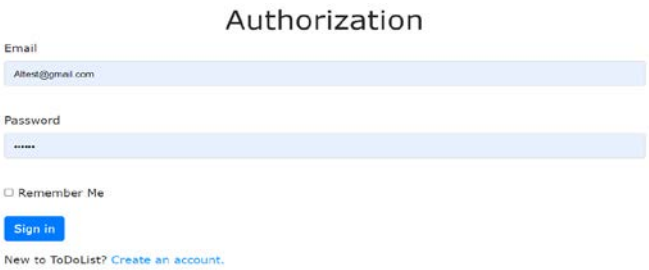


Рис. 1. Страница авторизации пользователя

После прохождения авторизации пользователь переходит на страницу создания задачи (см. рис. 2), соответственно при нажатии на кнопку «Add Task» происходит генерация задачи.

Рис. 2. Меню основного окна

Пользователь в диалоговом окне задает значения, указанные на рис. 3.

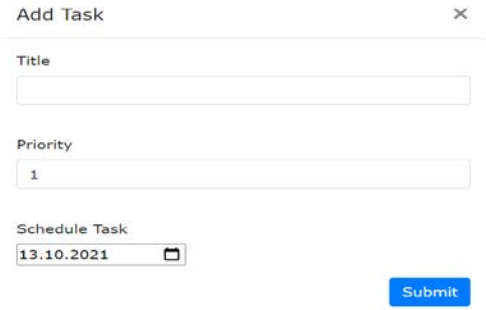


Рис. 3. Страница настроек задачи

При переходе в пункт «Statistics» выводится статистика пользователя, представленная на рис. 4 и рис. 5.

Рис. 4. Общая статистика

Weekly Activity

Day	Score
Wednesday	0
Tuesday	0
Monday	0
Sunday	0
Saturday	0
Friday	0
Thursday	0

Рис. 5. Подробная статистика

Также имеется возможность поиска задачи по названию, как показано на рис. 6.

Рис. 6. Строка поиска

Заключение

В данной статье была рассмотрена реализация веб-приложения «ToDo List», выполненного в рамках образовательного проекта Лицея Академии Яндекса, расположенного на базе Воронежского государственного университета факультета компьютерных наук. Все поставленные в начале реализации приложения функциональные требования были выполнены. Приложение является востребованным и полностью работоспособным.

Список литературы

1. Копытина, Е. А. Дополнительное образование школьников в сфере информационных технологий на примере курса «Основы программирования на языке Python» Яндекс.Лицея / Е. А. Копытина, А. В. Копытин, Ю. В. Шишко // Информатика: проблемы, методы, технологии: Материалы XX Международной научно-методической конференции, Воронеж, 13–14 февраля 2020 года / Под редакцией А.А. Зацаринного, Д.Н. Борисова. – Воронеж: «Научно-исследовательские публикации» (ООО «Вэлборн»), 2020. – С. 1830-1834.

2. Копытина, Е. А. Аспекты дистанционного формата обучения дополнительного образования школьников в сфере информационных технологий на примере курсов Яндекс.Лицея / Е. А. Копытина, А. В. Копытин, Ю. В. Шишко // Информатика: проблемы, методы, технологии: Материалы XXI Международной научно-методической конференции, Воронеж, 11–12 февраля 2021 года. – Воронеж: Общество с ограниченной ответственностью «Вэлборн», 2021. – С. 2082-2086.

3. Другалев, А. И. Разработка модуля постановки и контроля задач на платформе 1С: Предприятие 8.3 с применением технологии SMART / А. И. Другалев, Е. А. Копытина // Информатика: проблемы, методология, технологии: материалы XVI международной научно-методической конференции, Воронеж, 11–12 февраля 2016 года. – Воронеж: Научно-исследовательские публикации, 2016. – С. 74-79.

4. Копытин, А. В. О распределении вознаграждений в проектах / А. В. Копытин, Д. И. Соломатин, Е. А. Копытина // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2016. – № 1. – С. 72-76.

5. Погорелов, Р. И. Адаптация тиражируемого решения от фирмы 1С (1С: Управление нашей фирмой) и разработка приложения на мобильной платформе 1С / Р. И. Погорелов, А. В. Копытин, Е. А. Копытина // Труды молодых учёных факультета компьютерных наук ВГУ : Сборник статей / Под редакцией Д.Н. Борисова. – Воронеж : Общество с ограниченной ответственностью «Вэлборн», 2021. – С. 121-125.

Решение задач Text Mining с использованием модели Bert

Е. Д. Коротеева

Студент бакалавриата

В. В. Гаршина

Доцент

Введение

Bert (Bidirectional encoder representations from transformers) – это разработанная Google техника машинного обучения, основанная на трансформаторах. Методика предназначена для предварительной подготовки обработки естественного языка.

Разработка и представление Bert с быстрым подходом на основе механизма внимания являются прорывными в сфере Data Mining, ведь модель заменила реккурентные нейронные сети LSTM и GRU, обученные на упорядоченной последовательности (слева направо или справа налево), методика Bert позволяет изучать контекст слова на основе слов, находящихся вокруг него. Другими словами, эта модель умеет «читать» и выявлять контекст, в котором используется слово в запросе, анализируя и предугадывая его общий смысл в конкретном предложении. Например, в словосочетаниях «bring up» и «go up» предлог «up» имеет два разных значения, которые не очевидны большинству поисковых систем. Bert умеет различать языковые нюансы, обеспечивая более точные результаты и соответствуя информационным нуждам пользователя [1].

Человеческий разум не в состоянии воспринимать огромное количество разнородной информации. В среднем каждый из нас может различать только две-три взаимосвязи в достаточно необъемных выборках. Именно поэтому в настоящее время актуально изучать и развивать модели обработки естественного языка для ускорения и облегчения процесса текстовой классификации.

На данный момент Bert обучена двум алгоритмам по работе с текстом: она умеет подбирать подходящие по контексту и смыслу специально пропущенные слова в достаточно объемных текстах, а также проверять, могут ли два предложения, идущие друг за другом быть согласованы по смыслу. Стоит отметить, что модель имеет открытый исходный код, а это значит, что любой желающий может изучить и

дообучить модель на те методы, которые ему необходимы для работы с текстом.

Целью данной работы поставлены анализ существующих решений по интеллектуальному анализу текста, выявление их слабых сторон для русскоговорящего пользователя, изучение и последующее дообучение модели Bert под один из возможных сценариев работы с текстом и тестирование созданного нами решения. Для разработки и исследования нами будут использоваться онлайн-версия модели Bert, инструмент для разработки и представления проектов Data Science в интерактивном виде Jupyter Notebook и язык программирования Python.

1. Предварительное обучение

Чтобы Bert научился обрабатывать различные задачи, на входе можно подавать как одно, так и пару предложений в одной и той же последовательности токенов. Модель использует эмбеддинг WordPiece со словарным запасом в 30 000 токенов. Первый токен в каждой последовательности — это специальный токен для классификации предложения [CLS], окончательное скрытое состояние которого используется в качестве представления совокупной последовательности для задач классификации. Предложения в последовательности различают двумя способами. Можно разделять их с помощью специального токена [SEP], а можно добавить заученный эмбеддинг к каждому токену, чтобы указать, принадлежит ли он предложению A или предложению B. Последний скрытый вектор специального токена обозначается как C, а последний скрытый вектор некоторого входного токена *i* обозначается как T*i [2]*.

Для данного токена его входное представление строится путем суммирования соответствующих эмбеддингов токена, сегмента и позиции.

Стандартные модели можно обучать только слева направо или справа налево, в свою очередь двунаправленное обусловливание позволяет каждому слову непоследовательно видеть себя, а модель может тривиально предсказывать следующее слово в многослойном контексте. Чтобы обучить глубокое двунаправленное представление, случайный процент (15% в рассматриваемом примере) входных токенов маскируют и тренируют нейросеть предсказывать закрытые маской токены. Финальные скрытые векторы, соответствующие маскированным токенам, передаются в выходной softmax слой из словаря. Однако в результате получается несогласованность между предварительным обучением и точной настройкой, поскольку токен, используемый для маскировки [MASK], не появляется во время точной

настройки. Чтобы исправить этот недостаток, маскированный токен заменяется на [MASK] в 80% случаев, заменяется случайным токеном в 10% случаев и остается неизменным в 10% случаев.

Многие последующие задачи основаны на понимании отношений между предложениями. Это в какой-то степени отражает суть языкового моделирования. Чтобы обучить модель понимать отношения между предложениями, ее предварительно обучают бинаризованной задаче прогнозирования следующего предложения. При подготовке примеров предложений A и B для предварительного обучения в 50% случаев B — это фактическое следующее предложение, которое следует за A, а в 50% случаев B — случайное предложение из корпуса.

Корпуса, используемые для предварительного обучения:

– BooksCorpus (800 млн слов)
– English Wikipedia (2500 млн слов)

2. Точная настройка Bert

В BERT точная настройка выполняется путем простой замены соответствующих входных и выходных данных в зависимости от того, включают ли последующие задачи один текст или текстовые пары. Для каждой конкретной задачи подключают входные и выходные данные соответственно и полностью настраивают модель.

3. Результат предварительного обучения

No NSP обучается без задачи прогнозирования следующего предложения. LTR & No NSP обучается как LM слева направо без задачи предсказания следующего предложения, как OpenAI GPT.

+ BiLSTM добавляет случайно инициализированный BiLSTM поверх модели LTR + No NSP во время точной настройки.

4. Проведение процедур

Задачей для дообучения и тестирования модели Bert нами была выбрана задача разбиения текста на несколько меток, то есть классификация заданного текста. Обычно в задачах такого рода предполагается, что один элемент относится к одной метке или одному классу, на деле любой текст может независимо и одновременно быть отнесен к нескольким классам. Такие задачи применимы не только при работе с текстом, но и в прикладных сценариях: приписывание нескольких жанров к сериалам, разбиение товаром по категориям, в сфере клиентского обслуживания такие они так же могут быть использованы, к примеру, для определения угрозы или оскорбления в комментариях пользователей социальных сетей. Последнему мы и будем обучать нашу модель.

Для начала мы определим классы комментариев, по которым наша модель должна будет раскидывать предоставленные ей данные: неприемлемый, очень неприемлемый, ненавистный, неприличный, с элементами оскорбления, с элементами угрозы.

Для работы мы возьмем опубликованную Google Research модель Bert-base[5], которая отличается от классической тем, что при помощи токенайзера приводит исходный текст сразу к нижнему регистру и удаляет маркеры акцента.

Все наши данные будут представлены в классе Input и разбиты по трем группам:

- first_text – комментарии в виде текста
- second_text – текст, который не используется
- labels – или ярлыки, классы, по которым мы распределяем комментарии

Для работы нам требуется изменить наш класс в признак, с которым сможет работать модель. Это признак мы помещаем в класс Features.

- ids – список уникальных номеров для токенизированного текста
- mask – значения 0 или 1, зависит от того реальный токен или для заполнения
- segment – список единиц
- labels – one-hot кодированные класса для текста

Изначально Bert-base использует словарь, который состоит более, чем из тридцати тысяч слов, токенизация в случае модели означает распределение входного текста на токены, то есть слова, которые есть в словаре. В случае если встречающегося слова нет, модель использует алгоритм, основанный на WordPiece токенизации[3], который называется BPE. С помощью этого алгоритма, слово, которого нет в словаре постепенно расчленяется на меньшие части, то есть слово теперь представляется массивом из его составляющих. Доводится слово до того вида, когда все его составляющие найдены в словаре, что позволяет узнать его контекст, в конечном варианте этот контекст является набором контекстов его составляющих.

Для дообучения адаптируем исходный класс BertForSequenceClassification[4]. Основное форматирование заключается во внедрении бинарной кросс-энтропийной функции потерь вместо находящейся в первоначальном варианте классической кросс-энтропийной функции. Такая функция дает нашей модели возможность присваивать данным независимые вероятности.

Исходный код предоставляет доступ к информации о слоях модели:

- Embeddings – входной слой вложения

– Encoder – 12-attention слов
– Classifier – созданный нами классификатор нескольким классов с числом признаков – 6, что соответствует нашим созданным 6 классам

Алгоритм обучения похож на оригинальную реализацию Bert, мы дообучаем нашу модель на протяжении 4 эпох, беря максимально возможную для дообученных моделей длину последовательности – 512. Основной параметр learning-rate делаем равным 3e-5.

В связи с тем, что мы не будем использовать при дообучении технику FP16, так как по ряду причин наша бинарная функция не совместима с этой техникой, время на обучение будет немного более длительным, чем обычно.

5. Тестирование

Тестирование нашей дообученной модели проводилось при использовании различного рода комментариев, созданных нами лично и взятый из многообразных интернет ресурсов. Нами были видоизменен ряд комментариев при попытке уловить слабые стороны модели и выявить ее недостатки, мы получила следующие результаты:

– Тренировочные потери – 0.018
– Потери на валидации – 0.021
– Точность на валидации – 99.27%

В качестве метрической функции для каждого класса комментариев нами была использована кривая ROC-AUC, с ее помощью нами были получены следующие результаты:

– Неприемлемые – 0.9986
– Очень неприемлемые – 0.9940
– Ненавистные – 0.9989
– Угрожающие – 0.9987
– Оскорбляющие – 0.9977
– Неприличные – 0.9979

Полученный график ошибок по эпохам представлен на рисунке.

Заключение

В настоящей работе была рассмотрена модель интеллектуального анализа текста Bert. Основной задачей в нашем исследование стало проектирование дообучения модели Bert для последующего применения созданного алгоритма в прикладных задачах. Нами было решено написание обучения для мультиклассовой классификации текстов. Такая задача может быть применима во многих прикладных сферах, так как она заключается в разбиение входного текста на определенное количество меток или классов, так же она заключает в себе возможность

независимого попадания элементов в несколько групп. Нами было выбрано узкое направление: классификация негативных комментариев по шести выбранных нами меткам.

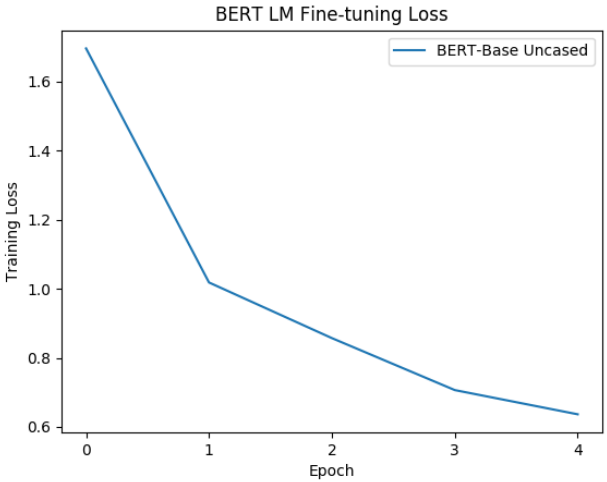


Рисунок. График ошибок по 4 эпохам

Было проведено дообучение модели Bert при помощи нескольких инструментов проектирования. Нами был успешно создан и протестирован алгоритм разбиения, позволяющий группировать исходные данные.

Стоит отметить, что изучение и последующее обучение модели было крайне доступным и легким, так как она имеет открытый код и доступна на 104 языках, в том числе и на русском.

Нами был произведен успешный запуск дообученной модели, тестирование с минимальными погрешностями в результатах, а также плодотворный анализ данных из доступных нам ресурсов, что свидетельствует о достигнутых целях, поставленных нами перед началом исследований.

Список литературы

1. Вилкок, Г. Introduction to Linguistic Annotation and Text Analytics / Грэм Вилкок ; Т8 Издательские технологии – 1-е изд.., 2009. – 160 с.

2. Коту, В. Data Science: Concepts and Practice / Виджэй Коту : Морган Кауфманн. – 3-е изд., 2018. – 568 с.

3. Сеннрих, Р. Neaural Machine Translation of Rare Words with Subword Units / Р. Сеннрих, Б. Хэддоу, А. Берч : Association for Computational Linguistics. – 1-е изд., 2016. – С. 1715-1805.

4. Github [Электронный ресурс] : веб-сервис для хостинга и разработки IT-проектов. – Режим доступа: https://github.com/huggingface /transformers

5. Github [Электронный ресурс] : веб-сервис для хостинга и разработки IT-проектов. – Режим доступа: https://github.com/google-research/bert

Проектирование системы классификации правонарушений наркоконтроля по Воронежской области

А. Н. Кропачев

Студент бакалавриата

Е. А. Копытина

Старший преподаватель

Введение

Не каждый человек соблюдает правила, а часто и нарушает их. Так происходит и с законами стран. Для предотвращения правонарушений была создана система контроля и наказания несоблюдения законов – правоохранительные органы, которая является развитием системы, описанной в [1]. К сожалению, большое число правонарушений даже такой развитый государственный аппарат контроля не способен предотвратить. Часто это происходит из-за недостатка информации, невозможности «связать» одно правонарушение или преступника с другим. Но что, если попытаться классифицировать преступления по определенным признакам на основе уже существующей истории нарушения закона. Ведь у каждого правонарушения существует ряд факторов, которые могут совпадать и различаться с факторами другого правонарушения. Это позволит не только определить склонность человека к дальнейшим правонарушениям, но и провести подробный анализ преступлений по общим факторам, составить классы нарушений закона, определить сезонность и тренд, выявить, какие признаки наиболее часто и сильно влияют на совершение преступлений. Из всего вышесказанного следует вывод о том, что актуальность реализации системы классификации правонарушений обусловлена не только теоретическим интересом, но и дальнейшей практической выгодой. Информационные технологии и компьютерные вычисления помогут обработать огромный объем данных, провести классификацию преступлений и проанализировать результат. Подобные подходы были описаны в [2-5].

1. Постановка задачи

К описываемой в статье системе классификации правонарушений были выдвинуты следующие требования:

1. Реализация возможности преобразования «человеческих» данных в «компьютерные» и сопоставления результата классификации с исходными данными.

2. Реализация возможности сохранения результатов классификации в памяти компьютера.

3. Реализация возможности формирования отчетов на основе данных проведенных экспериментов или сохраненных результатов в памяти компьютера ранее проведенных экспериментов.

2. Реализация логики

В качестве главного средства реализации выбран инструмент, позволяющий не только проектировать информационные системы, но и выполнять сложнейшие расчеты, связанные с большими объемами данных, – язык программирования Python.

Для реализации методов машинного обучения выбрана широко известная библиотека Sklearn. Главные преимущества данной библиотеки:

1. Поддержка предварительной обработки данных.
2. Возможность уменьшения размерности выборки данных.
3. Расчет регрессии.
4. Методы классификации и кластерного анализа.

При помощи данной библиотеки был реализован метод кластерного анализа и построена модель линейной регрессии.

Для построения графиков на основе проведенных экспериментов выбрана библиотека Matprolib. Главным преимуществом является возможность формирования 2D графиков из огромного множества представленных библиотекой моделей визуализации данных.

Для подключения к базе данных была использована библиотека Pyodbc, главным преимуществом которой является возможность подключения к базам данных на базе Microsoft Access, в частности к формату данных «.mdb».

В ходе проектирования системы классификации правонарушений были разработаны следующие модули:

1. CKS.py;
2. OnlyTable.py;
3. ML.py;
4. CondWindow.py;
5. SaveDialog.py;
6. Dialog.py;
7. AuthDialog.py.

3. Реализация интерфейса

В классе AuthDialog.py реализован процесс авторизации в систему. Необходимо указать путь к базе данных, пользователя и пароль. При неверном вводе логина или пароля система сообщит в окне ошибки и попросит ввести логин и пароль заново. Можно указать стандартный путь к базе данных, который устанавливаться по умолчанию при каждом запуске системы в поле ввода пути (см. рис. 1).

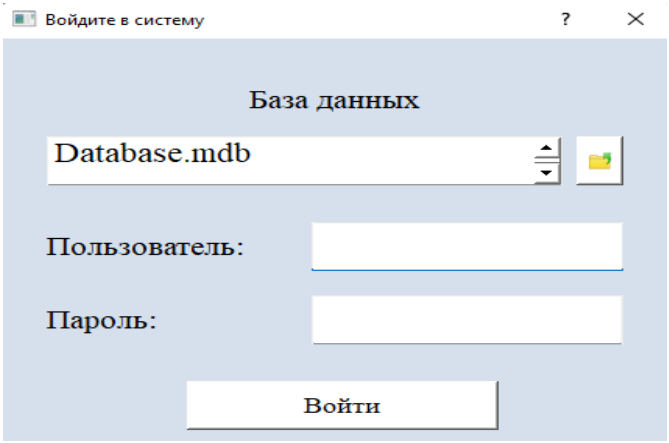


Рис. 1. Окно авторизации

После прохождения авторизации пользователь переходит в главное окно системы, которое дает доступ к просмотру всех таблиц базы данных или классификации данных (см. рис. 2). При нажатии на кнопку «Классификация» происходит открытие окна классификации данных.

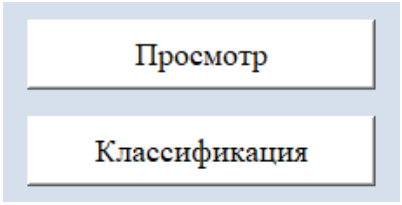


Рис. 2. Меню основного окна

Пользователю необходимо выбрать таблицу, преобразовать исходные данные определенным образом и выбрать соответствующий метод классификации данных (см. рис. 3).

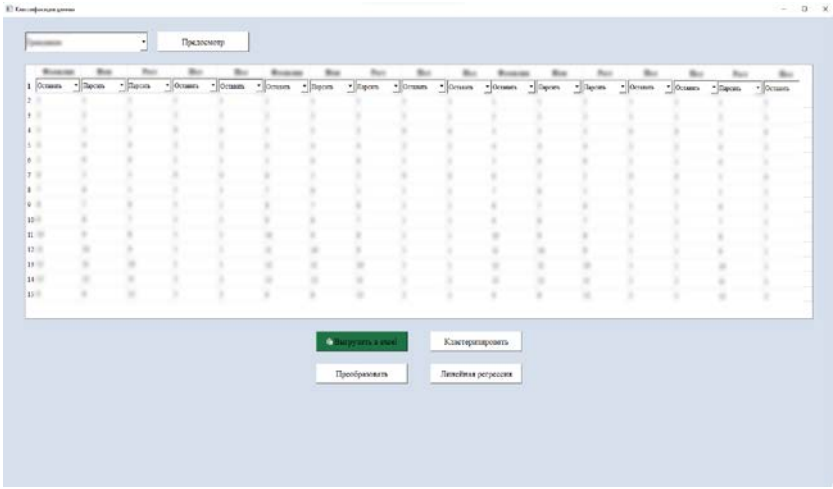


Рис. 3. Окно классификации данных

При выборе метода классификации данных, появится окно с описанием указанного метода и требованием ввести дополнительные уникальные параметры. Например, для метода кластерного анализа необходимо указать количество кластеров (см. рис. 4).

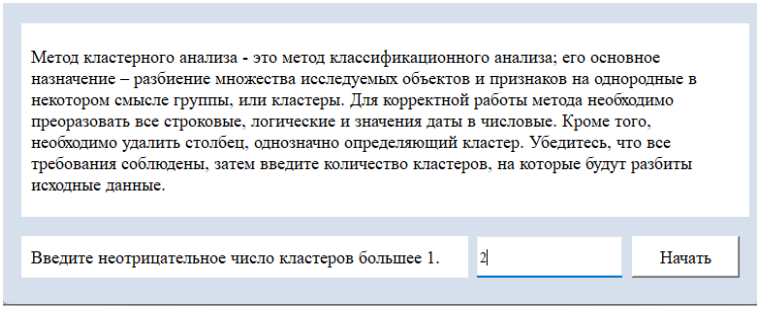


Рис. 4. Метод кластерного анализа

После того, как система выполнит анализ, появится диалоговое окно с сообщением «Успех» и появится окно результата с кратким и подробным описанием (см. рис. 5).

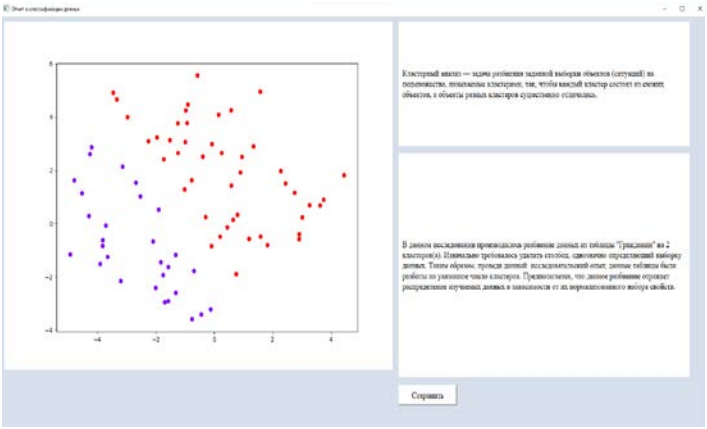
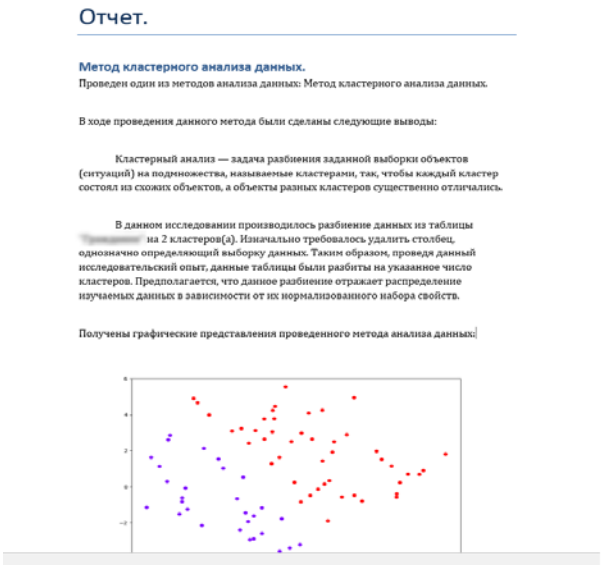


Рис. 5. Результат выполнения классификации данных

При нажатии клавиши «Сохранить» данных результат будет сохранен в файл формата «.docx», необходимо указать путь и название для файла (см. рис. 6).



Отчет.

Метод кластерного анализа данных.

Проведен один из методов анализа данных: Метод кластерного анализа данных.

В ходе проведения данного метода были сделаны следующие выводы:

Кластерный анализ — задача разбиения заданной выборки объектов (ситуаций) на подмножества, называемые кластерами, так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались.

В данном исследовании производилось разбиение данных из таблицы [redacted] на 2 кластеров(а). Изначально требовалось удалить столбец, однозначно определяющий выборку данных. Таким образом, проведя данный исследовательский опыт, данные таблицы были разбиты на указанное число кластеров. Предполагается, что данное разбиение отражает распределение изучаемых данных в зависимости от их нормализованного набора свойств.

Получены графические представления проведенного метода анализа данных:

Рис. 6. Строка поиска

Заключение

В данной статье было рассмотрено проектирование системы классификации правонарушений наркоконтроля по Воронежской области, Все поставленные в начале реализации приложения функциональные требования были выполнены. Приложение является востребованным и полностью работоспособным.

Список литературы

1. Кропачев, А. Н. Формирование отчета деятельности наркоконтроля по Воронежской области средствами офисных приложений / А. Н. Кропачев, Е. А. Копытина // Труды молодых учёных факультета компьютерных наук ВГУ: Сборник статей / Под редакцией Д.Н. Борисова. – Воронеж: Общество с ограниченной ответственностью "Вэлборн", 2021. – С. 80-84.
2. Копытина, Е. А. Определение тенденции развития строительной организации на основе прогнозирования временных рядов / Е. А. Копытина, Н. А. Петрикеева, Д. М. Чудинов // Информационные технологии в строительных, социальных и экономических системах. – 2020. – № 1(19). – С. 87-91.
3. Радчук, М. А. Разработка программного средства для предсказания оттока клиентов с помощью методов машинного обучения / М. А. Радчук, Е. А. Копытина // Сборник студенческих научных работ факультета компьютерных наук ВГУ : Сборник статей. В 2-х частях / Под редакцией Д.Н. Борисова. – Воронеж: Воронежский государственный университет, 2019. – С. 190-196.
4. Копытина, Е. А. Моделирование и реализация системы расчета заказов торговой сети / Е. А. Копытина, А. В. Копытин // Информатика: проблемы, методы, технологии : Материалы XXI Международной научно-методической конференции, Воронеж, 11–12 февраля 2021 года. – Воронеж: Общество с ограниченной ответственностью "Вэлборн", 2021. – С. 1240-1249.
5. 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. – Воронеж: Воронежский государственный университет, 2020. – С. 143-147.

Разработка собственного менеджера памяти для игрового движка

С. С. Крупенин

Студент бакалавриата

Д. И. Соломатин

Старший преподаватель

Введение

Игровой движок – это центральный программный компонент любой компьютерной игры. Такой программный компонент очень важен для игры и игрового процесса, ведь он обеспечивает такие взаимодействия как: взаимодействие игрока на виртуальное окружение, объектов этого окружения с игроком и объектов между друг другом. Игровой движок обеспечивает функциональность, включающую такие компоненты как: система рендеринга для 2D или 3D графики, физическую симуляцию, систему обнаружения коллизий (столкновений) объектов, звук, анимацию, искусственный интеллект, чтение сигналов с устройства ввода.

Одной из основных проблем при разработке движка является выбор необходимой архитектуры, которая сможет эффективно распределять ресурсы, осуществлять взаимодействия объектов виртуального пространства и при этом иметь гибкость, позволяющая изменять механики игры без каких-либо проблем. Но эта проблема достаточно просто решается, если использовать в основе движка паттерн Entity-Component-System (ECS).

Второй проблемой является распределение ресурсов при работе движка в течении игрового процесса, так как эффективность и скорость работы зависит от скорости выделения, освобождения и взаимодействия с ресурсами. Поэтому стоит избегать системных вызовов. Единого решения данной проблемы не существует.

В данной работе будет представлено собственное решение распределения памяти для игрового движка, основанного на паттерне ECS.

1. Особенности паттерна ECS

ECS – это шаблон проектирования, обеспечивающий огромную гибкость в проектировании общей архитектуры программного обеспечения.

Распределение различных проблем и задач между сущностями (Entities), компонентами (Components) и системами (Systems). Это три основных понятия связаны между собой довольно свободно.

Сущности обычно используются для создания уникального идентификатора, предоставления среде информации о существовании отдельного элемента и функции как своего рода корневого объекта, объединяющего множество компонентов.

Компоненты – это не что иное, как объекты-контейнеры, не обладающие никакой сложной логикой. В идеале они являются объектами с простой структурой данных.

Системы – это логические компоненты, в каждой из которых описывается конкретная механика. Они не владеют ни сущностями, ни компонентами. Они имеют доступ к ним через независимые объекты-диспетчеры, которые в свою очередь управляют жизненным циклом сущностей и компонентов [1].

Все компоненты и сущности распределены в специальных контейнерах или пулах по типу.

Таким образом, эффективность самого движка зависит от эффективности работы этих пулов, так как они являются точкой связи игрового движка с ресурсами компьютера.

2. Постановка задачи

Стандартная работа с динамической памятью, основанной на системных вызовах, не может быть достаточно эффективной. Тем более выявлено, что скорость выделения памяти зависит от операционной системы. Так было проанализированы выделение памяти на системах семейства Linux и Windows. Для наглядности этого на рис. 1 показаны графики времени выделения от размера требуемой памяти в зависимости от ОС.

По этой причине выбор разработки собственного диспетчера памяти, который не будет зависеть от системы, является вполне логичным.

3. Разработка системы распределения памяти

Вся система распределения памяти состоит из разных типов распределителей, где у каждого своя задача.

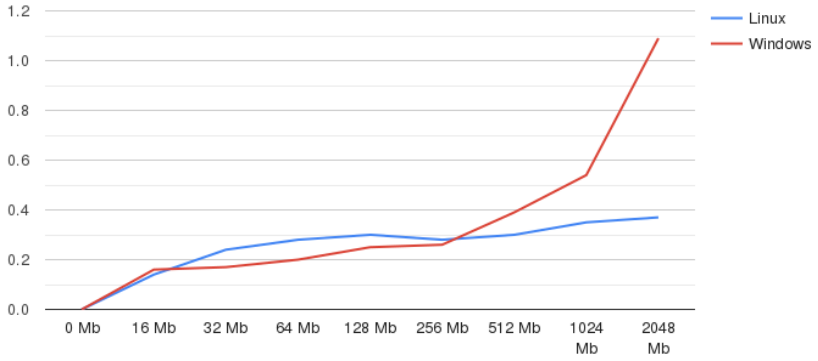


Рис. 1. Сравнение скорости выделение памяти для ОС Windows и семейства Linux

Система состоит из следующих распределителей:

– Linear Allocator – идея состоит в том, чтобы сохранить указатель на начало блока выделенной памяти, а также использовать другой указатель, который будет перемещаться каждый раз, когда происходит выделение. Данный распределитель полезен для объектов с жизненным циклом равным циклу всего приложения.

– Stack Allocator – позволяет управлять памятью, как стеком. Так же, как и раньше, мы сохраняем указатель вместе с заголовочным блоком на текущий адрес памяти и перемещаем его вперед для каждого выделения. Данный аллокатор очень полезен для создания глобального распределителя памяти, так как позволяет объединять освобожденные сегменты памяти.

– Pool Allocator – идея заключается в том, что он разделяет некоторый большой участок памяти на более мелкие участки одинакового размера [2].

Pool Allocator хорошо подходит для работы с однотипными сущностями и компонентами паттерна ECS, чей размер одинаков. Так же это отлично подходит для реализации пула компонентов или сущностей. Из-за того, что выделенные участки хранятся в списке, есть возможность итерироваться по всем объектам, лежащих в одной части, с последующим приведением типов.

Но хранение ссылок на выделенные объекты в простом линейном массиве вызывает линейную сложность, что в свою очередь уменьшает эффективность. Так как после удаления некоторых элементов появляются промежутки пустых ячеек, по которым придется итерироваться. И для этого необходимо сделать двусвязный список,

который указывает на индексы предыдущего и последующего элемента, но также имеет связь с индексом текущего элемента. Это можно реализовать с помощью линейного массива, который будет хранить элементы связного списка. Таким образом, мы сохраняем информацию о всех трех необходимых компонентах. Также, добавим к этой схеме массива стек, который будет содержать информацию о пустых ячейках. Данная схема показана на рис. 2. Благодаря этому скорость итерации по объектам пула будет равна скорости итерированию по односвязному списку, а скорость создания и удаления произвольного элемента будет константой.

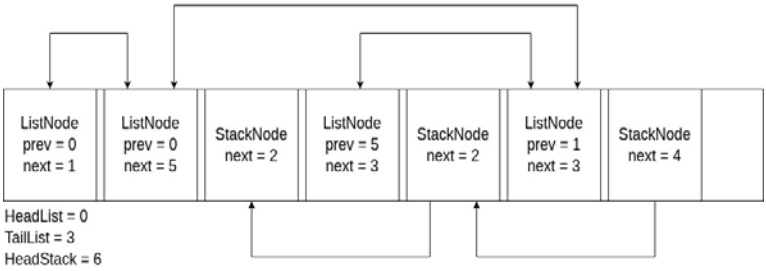


Рис. 2. Графическое представление массива свободных элементов

Связь между этим массивом и памятью, находящейся в распределителе, можно легко установить с помощью того, что известен начальный адрес и размер объекта. А зная это, можно узнать индекс текущего элемента по его указателю с помощью формулы:

$$\text{index} = \frac{\text{currPtr} - \text{startPtr}}{\text{sizeObject}},$$

где index – это индекс элемента массива, currPtr – указатель текущего объекта, startPtr – начальный указатель на выделенную область памяти, sizeObject – размер объекта.

В итоге скорость выделения стала значительно выше, что можно увидеть на графике рис. 3.

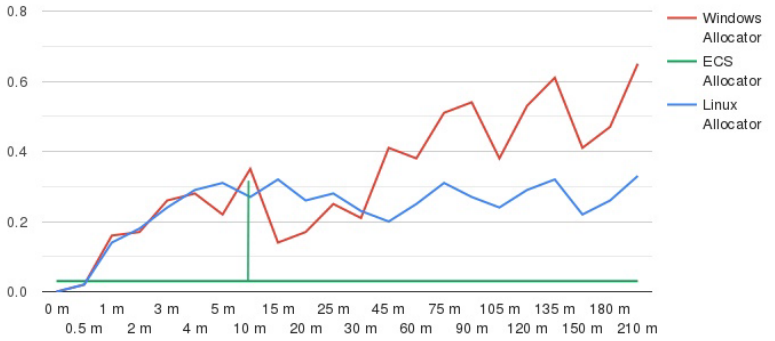


Рис. 3. График времени выделения(с) памяти во время работы приложения(мин) с использованием распределителями памяти ОС Windows, семейства Linux и разработанного распределителя

На графике видно, что в некоторые моменты времени скорость выделения памяти значительно снижается. Это связанно с тем, что в случае нехватки памяти будет выделен дополнительный объем у системы. Данное явление показано на рис. 4.

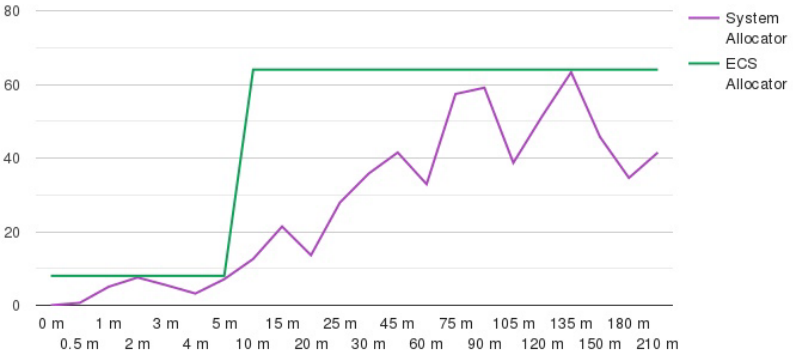


Рис. 4. График потребления памяти(Мб) во время работы приложения(мин) с использованием распределением памяти ОС и разработанного распределителя

Графики, представленные на рис. 3 и рис. 4 показывают, что кол-во вызовов к системной у обычного подхода достаточно большое, что вызывает сильную фрагментацию памяти. С течением времени это сильно сказывается на производительности. [3]

Заключение

Таким образом, получилось создать распределитель памяти для компонентов и сущностей паттерна ECS, который позволяет ускорить игровой движок в несколько сотен раз. По сравнению с системным распределением и стандартным хранением свободных частей памяти, разработанный алгоритм отличается большой эффективностью и малой алгоритмической сложностью.

Список литературы

1. Entity Component System [Электронный ресурс]. – Режим доступа: https://tsprojectsblog.wordpress.com/portfolio/entity-component-system/
2. Аллокаторы памяти [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/505632/
3. Менеджмент памяти или реже стрелять в ногу [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/473294/

Автоматизация взаимоотношений с клиентами в SEO-компании

С. Ш. Манукян

Студент бакалавриата

Е. А. Копытина

Старший преподаватель

Введение

В настоящее время взаимоотношения с клиентами являются главной целью каждой компании, ведь это напрямую влияет на ее статус и репутацию. Возможность увеличения клиентской базы - шаг к достижению целей по росту выручки, а также развитию внешних и внутренних отношений компании. Проект, описываемый в данной статье, является крайне актуальным и востребованным, так как он способен решить столь серьёзную в наше время проблему, проблему оптимизации деятельности SEO-компаний путем автоматизации бизнес-процессов, и, как следствие – рост финансовых показателей организации.

В любой компании требуется регулярно повышать эффективность производства, совершенствовать качество обслуживания клиентов, осуществлять эффективное прогнозирование, как показано в [1-5]. Это обуславливает актуальность темы: для достижения вышеперечисленных целей и улучшения показателей, необходима автоматизация процесса взаимодействия с клиентами.

1. Постановка задачи

К описываемому в статье приложению «CRM-система» были выдвинуты следующие требования:

- Авторизация проектного менеджера компании;
- Возможность неавторизованному пользователю оставить заявку для первичного звонка с проектным менеджером компании;
- Возможность создания, редактирования и удаления задачи на канбан-доске;
- Возможность задать точное время на выполнение задачи;
- Возможность просмотра статистики выполненных задач.

2. Реализация логики

Приложение написано на объектно-ориентированном языке Python.

Для реализации были использованы следующие инструменты веб-разработки:

- Фреймворк Django,
- HTML,
- CSS,
- JavaScript,
- СУБД MySQL.

Разработанное приложение имеет клиент-серверную архитектуру. Клиент и сервер взаимодействуют посредством обмена сообщений типа GET, POST протокола HTTP.

Серверная часть представлена следующими модулями:

- models.py, где реализованы классы задач, пользователя,
- urls.py,
- views.py.

Например, модуль urls.py содержит в себе логику по обработке запросов и состоит из пар запрос - ответ в виде функции, описанной в файле views.py приложения проекта.

Перед тем разработать CRM веб-сервис, на основе анализа предметной области был разработан прототип БД и выявлены следующие сущности:

- user,
- request,
- project,
- task.

Также были разработаны следующие представления пользовательского интерфейса, содержащиеся в папке templates:

- index.html,
- card.html,
- newrequest.html,
- cards.html,
- projects.html,
- project.html,
- statistic.html.

3. Реализация интерфейса

Главная страница пользователя представлена на рис.1. Пользователь видит описание компании и предоставляемых ею услуг. На этой странице можно оставить заявку на выполнение проекта.

Рис. 1. Главная страница пользователя

После нажатия на кнопку «Создать заявку» она попадает на главную страницу менеджера по продажам (см. рис. 2).

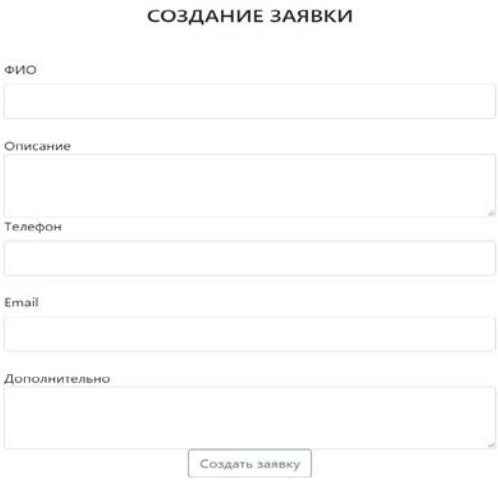


Рис. 2. Страница создания заявки

Главная страница для менеджера, проектного менеджера и разработчика выглядит следующим образом.

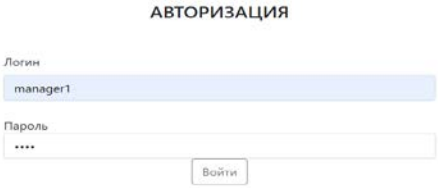


Рис. 3. Страница авторизации

Статистика представлена следующими диаграммами на рис 4.

Рис. 4. Страница статистики

Страница каждого конкретного проекта представляет собой канбан-доску (см. рис 5). Здесь проектных менеджер может добавлять новые задачи, а также назначать на каждую из них исполнителя из списка всех исполнителей. Все задачи делятся на три типа: не начатые, в процессе и выполненные.

Заключение

В данной статье была рассмотрена реализация веб-сервиса «CRM-система». Все поставленные в начале реализации приложения функциональные требования были выполнены. Приложение является востребованным и полностью работоспособным.

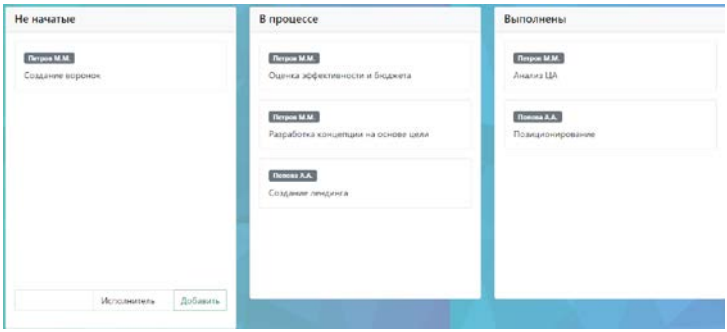


Рис. 5. Страница проекта для проектного менеджера

Список литературы

1. Копытина, Е. А. Моделирование и реализация системы расчета заказов торговой сети / Е. А. Копытина, А. В. Копытин // Информатика: проблемы, методы, технологии : Материалы XXI Международной научно-методической конференции, Воронеж, 11–12 февраля 2021 года. – Воронеж: Общество с ограниченной ответственностью «Вэлборн», 2021. – С. 1240-1249.

2. Погорелов, Р. И. Адаптация тиражируемого решения от фирмы 1С (1С: Управление нашей фирмой) и разработка приложения на мобильной платформе 1С / Р. И. Погорелов, А. В. Копытин, Е. А. Копытина // Труды молодых учёных факультета компьютерных наук ВГУ: Сборник статей / Под редакцией Д. Н. Борисова. – Воронеж : Общество с ограниченной ответственностью «Вэлборн», 2021. – С. 121-125.

3. Радчук, М. А. Разработка программного средства для предсказания оттока клиентов с помощью методов машинного обучения / М. А. Радчук, Е. А. Копытина // Сборник студенческих научных работ факультета компьютерных наук ВГУ: Сборник статей. В 2-х частях / Под редакцией Д.Н. Борисова. – Воронеж: Воронежский государственный университет, 2019. – С. 190-196.

4. Копытина, Е. А. Определение тенденции развития строительной организации на основе прогнозирования временных рядов / Е. А. Копытина, Н. А. Петрикеева, Д. М. Чудинов // Информационные технологии в строительных, социальных и экономических системах. – 2020. – № 1(19). – С. 87-91.

5. Землянухин, Д. В. Разработка системы контроля и управления параметрами производственных процессов с GSM-каналом телеметрии / Д. В. Землянухин, Е. А. Копытина, А. А. Головкин // Сборник

студенческих научных работ факультета компьютерных наук ВГУ: Сборник статей. В 2-х частях / Под редакцией Д.Н. Борисова. – Воронеж: Воронежский государственный университет, 2019. – С. 78-84.

Разработка приложения для подготовки к Единому Государственному экзамену

М. А. Мотенко

Студент бакалавриата

А. И. Чекмарев

Старший преподаватель

Введение

Каждому одиннадцатикласснику предстоит написание Единого Государственного Экзамена (ЕГЭ), по результатам которого будет решена их дальнейшая профессиональная деятельность. Для того чтобы поступить в престижное Высшее Учебное Заведение (ВУЗ) на техническое направление, выпускникам необходимо набрать высокие баллы по физике и профильной математике.

В настоящее время существует огромное количество интернет-ресурсов, помогающих школьникам в подготовке к выбранным дисциплинам. Однако выпускники часто сталкиваются с недостатками подобных сервисов. Например, некоторые направлены только на решение заданий или просто на разбор вариантов. Для получения хорошего результата ученики должны не только решать задания экзамена, но и качественно знать теоретический материал. Поэтому зачастую им приходится искать отдельные ресурсы для изучения теории. Также ученики не могут сравнивать свои успехи с другими учащимися, что резко увеличивает их уверенность в том, что их уровень знаний хороший. Если каждый школьник будет набирать баллы за свой прогресс в изучении материала, он сможет сравнивать себя с другими пользователями и иметь мотивацию постоянно совершенствовать свои достижения.

Для помощи выпускникам было разработано веб-приложение «EasyExam» , которое предоставит возможность подготовиться к ЕГЭ по физике и профильной математике без использования дополнительных ресурсов и поможет реально оценивать свой прогресс с помощью рейтинга.

1. Цель работы

Разработать веб-приложение для подготовки к Единому Государственному экзамену, которое поможет пользователям:

- изучать теоретический материал по темам экзамена;
- закреплять материал с помощью практических заданий;
- получать отзыв о своем решении от преподавателя;
- писать пробные экзамены;
- набирать баллы за изучение теории, решение заданий и вариантов;
- сравнивать свой прогресс со всеми пользователями.

2. Актуальность

В настоящее время существуют различные сервисы для подготовки к ЕГЭ, но средние баллы по профильной математике в табл. 1 [1] и физике в табл. 2 [1] показывают, что данные сервисы недостаточно эффективны.

Таблица 1

Средние баллы ЕГЭ по профильной математике за последние 6 лет

Год	Средний балл
2016	46,2
2017	47,1
2018	49,8
2019	56,5
2020	54,2
2021	55,1

Таблица 2

Средние баллы ЕГЭ по физике за последние 6 лет

Год	Средний балл
2016	50
2017	53,2
2018	53,2
2019	54,4
2020	54,5
2021	55,1

После проведения анализа сервисов для подготовки к экзаменам были выявлены их основные недостатки, представленные в табл. 3.

Таблица 3

Анализ аналогов

Недостаток	Решение
Нет возможности получения отзыва от преподавателя по решению второй части. На одном из известных порталов такая функция доступна, но ее использование неудобно для пользователей.	Создать кабинет преподавателя, которому будут поступать на проверку работы второй части. И добавить функцию сохранения предыдущих работ с удобным интерфейсом.
Один из порталов не удаляет неактуальные задания экзамена	Создать кабинет преподавателя, обязанностью которого будет являться добавление новых заданий (подходящих формату экзамена) и удаление неактуальных.
Есть сайты, которые содержат теорию по темам. Есть сайты, содержащие задания, разбитые по темам экзамена. Есть сайты для отработки вариантов. Чаще всего ученику нужно посетить три разных сайта, чтобы получить нужные навыки. (Стоит отметить, что существует один известный сайт, содержащий в себе все перечисленные элементы. Но найти теорию можно, если тщательно рассмотреть все вкладки сайта)	Сделать разбиение по темам экзамена так, чтобы теория и задания были в полном объёме и располагались рядом. Также вкладка для прохождения пробного экзамена должна находится на видном месте на сайте.
Нет возможности контроля своего прогресса	Реализовать возможность набора баллов за изучение теории, за верное решение практических заданий и варианта. Создание рейтинга на основе баллов пользователей.

В таблице видно, что имеются несколько важных недостатков подобных сервисов, которые мешают выпускникам правильно подготовиться к выбранным предметам.

Стоит отметить, что приложение EasyExam направлено не только на выпускников, но и на учеников 5-8 и 10 классов. Все темы школьной программы, входящие в состав экзаменов, в удобном теоретическом и практическом формате будут помогать школьникам в изучении и закреплении тем.

3. Архитектура приложения

На рис. 1 представлена диаграмма развертывания, чтобы определить какие аппаратные компоненты («узлы») существуют, какие программные компоненты(«артефакты») работают на каждом узле и как различные части этого комплекса соединяются друг с другом.

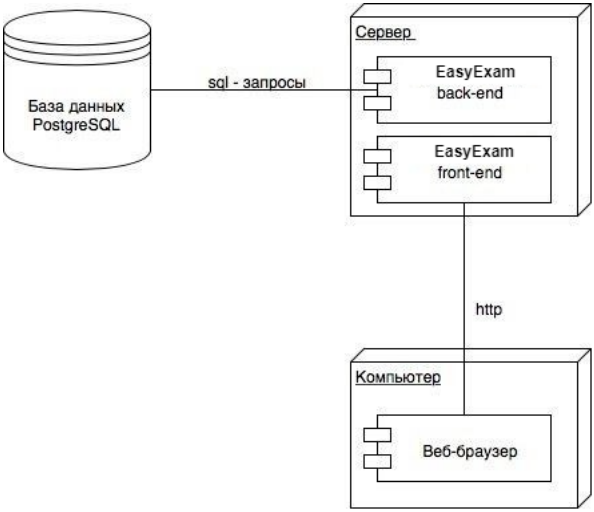


Рис. 1. Диаграмма развертывания

Для разрабатываемого веб-приложения узлом устройства является компьютер, сервер и база данных, а в качестве узла среды выполнения выступает веб браузер. На серверной части развернуты front-end и back-end приложения и отдельно база данных.

В качестве средств реализации приложения были выбраны следующие технологии:

Back-end:

- Python;
- Flask; [2]
- PostgreSQL; [3]
- SQLAlchemy.

Front-end:

- jinja2; [4]
- JavaScript;
- Bootstrap; [5]
- CSS 3;
- HTML 5.

4. Функциональность приложения и интерфейс пользователя

Рассмотрим основные страницы приложения, которые реализуют те функции, которые отличают данный сервис от подобных приложений.

В шапке сайта рядом с вкладкой «Предметы» располагается вкладка «Результаты» рис. 2.

Предмет	Первычиный балл	Тестовый балл	Потрачено времени	Статус	Просмотр работы
Математика	0.0 / 2	0 / 100	00:00:18	Не было отправлено	Смотреть работу
Математика	0.0 / 3	0 / 100	00:00:05	Не было отправлено	Смотреть работу
Математика	0.0 / 3	0 / 100	00:00:05	Не было отправлено	Смотреть работу
Математика	0.0 / 3	0 / 100	00:00:05	Не проверено	Смотреть работу
Математика	2.0 / 2	9 / 100	00:00:07	Проверено	Смотреть работу

Рис. 2. Страница с результатами предыдущих работ

Для неавторизованного пользователя откроется страница авторизации.

Для авторизованного пользователя будут доступны результаты предыдущих работ.

Также есть возможность просмотреть все свои предыдущие работы и отзывы преподавателей рис. 3.

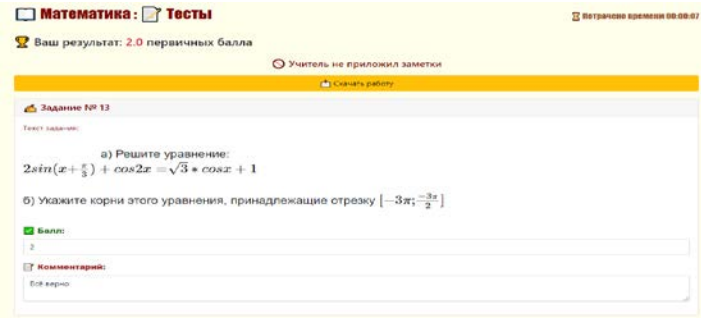


Рис. 3. Переход по кнопке «Смотреть работу»

На рис. 4 представлена страница с рейтингом, который был разработан для поддержания мотивации учеников. Пользователи получают баллы за прочтение теории и решение заданий.

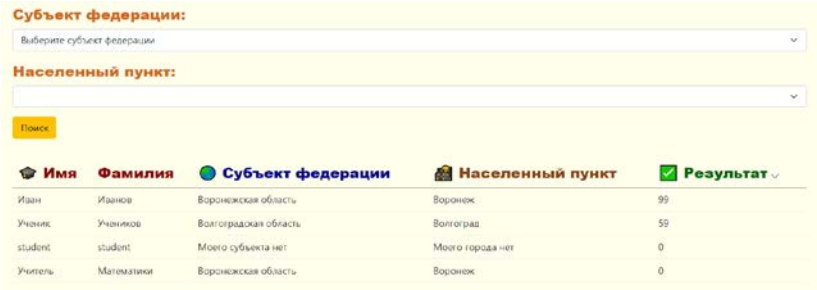


Рис. 4. Страница рейтинга

Данный рейтинг позволяет сравнивать свои результаты не только со всеми пользователями, но и с пользователями из своего населенного пункта рис. 5.

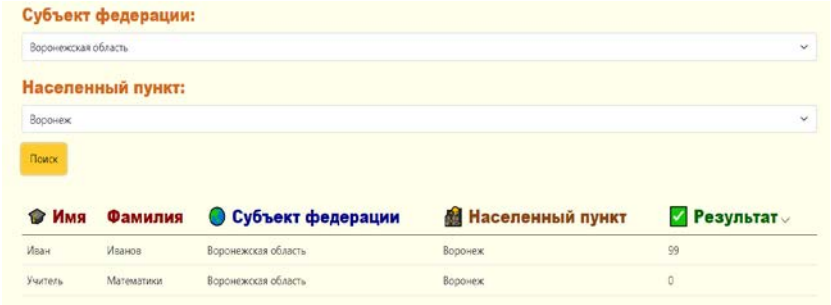


Рис. 5. Сортировка рейтинга по населенному пункту

Заключение

В заключение стоит отметить, что данное приложение полностью реализовано с учетом вышеописанной архитектуры, удовлетворяет схемам, а также выполняет все цели и поставленные задачи.

Данное приложение позволяет ученикам самостоятельно готовиться к выпускным экзаменам по профильной математике и физике.

Список литературы

1. Данные о средних баллах ЕГЭ: [Электронный ресурс]. – Режим доступа: https://blog.maximumtest.ru/post/srednij-ball-egeh-po-predmetam.html

2. Обзор фреймворка Flask: [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/346306/

3. Сравнение баз данных: [Электронный ресурс]. – Режим доступа: https://proglib.io/p/databases-2019

4. Основы шаблонизатора jinja2: [Электронный ресурс]. – Режим доступа: https://lectureswww.readthedocs.io/6.www.sync/2.codding/3.templates/jinja2.html

5. Изучение Bootstrap: [Электронный ресурс]. – Режим доступа : https://getbootstrap.com/

Разработка приложения для совместного просмотра видео

Е. Д. Нартов

Студент бакалавриата

Д. И. Соломатин

Старший преподаватель

Введение

В век информационных технологий медиа-контент получил широкое распространение. Особенно это стало заметно во время пандемии коронавируса, когда на замену личным встречам пришли видео-звонки, а на замену кинотеатрам пришли онлайн-трансляции. В связи возрастающей востребованностью таких платформ требования к качеству доставляемого на них мультимедийного контента все время увеличиваются.

Целью данной работы является создание приложения для совместного просмотра видео, подразумевающего качественную параллельную передачу медиа-контента нескольким пользователям с синхронизацией таймлайна с организатором просмотра.

Каждый клиент может просматривать контент на разных устройствах с разным качеством интернет-соединений. На выходе требуется получить наиболее быстрое, плавное и качественное воспроизведение аудио и видео, подстраивающееся под пропускную способность сети каждого пользователя. Одним из интенсивно развивающихся и распространенных способов доставки мультимедиа контента является потоковое вещание, которое позволяет решить вышеперечисленные проблемы. Для его реализации возникает потребность в наиболее эффективных протоколах потоковой передачи данных.

1. Требования к приложению

При разработке данного приложения выдвигались следующие требования:

возможность создания нового пользователя;

загрузка видео в некоторых форматах с последующим преобразованием для потокового вещания;

поиск видео по названию;

совместный синхронный просмотр видео с возможностью перемотки и остановки организатором просмотра.

2. Потоковое вещание видео

На сегодняшний день существует множество протоколов потокового вещания, наиболее популярные и распространенные из которых – RTMP, HLS, MPEG-DASH, Adobe HDS и др.

Для решения поставленной задачи наиболее подходящим и удобным проколом является HLS [1]. В отличие от многих других он нацелен не только на прямые эфиры, но и способен транслировать видео по запросу, что и требуется в рассматриваемой задаче. Также он является адаптивным протоколом, что позволяет плавно подстраивать качество потока под пропускную способность сети клиента на протяжении всего видео.

Данный протокол разделяет входящий файл или трансляцию на сегменты по 5-10 секунд в формате .ts и формирует из них манифест с расширением .m3u8 для разных битрейтов. Далее все созданные манифесты формируют общий мастер-манифест с тем же расширением .m3u8. Таким образом, получается адаптивный видеофайл готовый к потоковой передаче. Далее по этому манифесту осуществляется сегментная передача мультимедийных данных клиенту по стандартному протоколу HTTP, который позволяет легко преодолевать межсетевые экраны и прокси-сервера. Переданные данные запоминаются в буфер-памяти и воспроизводятся сразу, как только буфер становится полным. При этом, если манифест будет адаптивным, то качество следующего сегмента будет выбрано исходя из текущей пропускной способности сети.

Таким образом, проигрывание видео будет осуществляться плавно, без остановок и задержек.

3. Реализация

Для реализации данного приложения был использован язык Java с использованием фреймворка Spring Boot [2]. Для хранения данных о видео использовано облачное хранилище Amazon S3(Amazon Simple Storage Service) [3].

На рис. 1 приведена общая архитектура приложения, которая была разделена на несколько микросервисов:

– Frontend-service – предоставляет пользовательский интерфейс клиенту;

– Backend-service – собирает данные с других микросервисов и формирует общий ответ;

– User-management-service – организует всю работу с данными пользователей;

– Video-management-service – отвечает за агрегирование данных о фильмах;

– Video-streaming-service – отвечает за формирование комнаты для совместного просмотра и загрузку файлов на облачное хранилище.

Такой вариант архитектуры позволяет грамотно распределить нагрузку между сервисами.

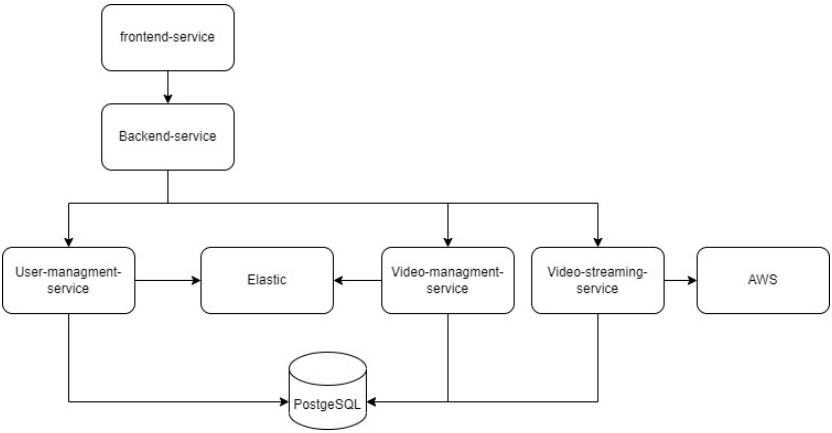


Рис. 1. Архитектура приложения

Каждый микросервис работает со своей базой данных PostgreSQL и осуществляет все необходимые преобразования над моделями данных (Entity). Также в каждый сервис внедрена логика отделения представления данных в базе и передаваемыми объектами с помощью средств Spring Boot.

Для быстрого поиска добавленных видео и пользователей в Video-management-service и User-management-service данные дублируются в базу Elasticsearch, которая позволяет осуществлять гибкий и быстрый поиск данных по запросу.

Для реализации загрузки видео и последующей его обработки для совместного просмотра в Video-streaming-service была интегрирована библиотека FFmpeg [4] с помощью Docker. При помощи данной библиотеки был реализован алгоритм преобразования видео в HLS. Загружая медиафайл, создается 3 конструктора для создания мастер-манифеста для разных битрейтов – 500 кбит/с, 1000 кбит/с, 2000 кбит/с.На выходе конвертации получается адаптивный файл для

последующей его передачи в формате HLS. Пример файла манифеста для одного из битрейтов можно увидеть в листинге.

Листинг

Манифест сегментов видео

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:8
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:8.333333,
index0.ts
#EXTINF:4.166667,
index1.ts
#EXTINF:8.333333,
index2.ts
#EXTINF:4.166667,
index3.ts
#EXTINF:5.033333,
index4.ts
#EXT-X-ENDLIST
```

Для последующего использования данного манифеста необходимо сохранить его метаданные (сегменты видео). В качестве хранилища для них будет выступает Amazon S3, взаимодействие с которым осуществляется посредством библиотеки AWS SDK [5]. Ссылки на сохраненные в нем данные необходимо проставить в Video-management-service для последующего их использования при совместном просмотре.

Благодаря данной реализации при запросе видео на Frontend-service приходит ссылка на манифест в облачном хранилище, при встраивании которой в плеер и осуществляется плавная подача медиаконтента.

4. Совместный просмотр видео

Помимо плавной передачи видео необходимо реализовать синхронизацию таймкода и состояния видео со всеми зрителями трансляции. Для решения этой проблемы использован обмен сообщениями через WebSocket по протоколу Streaming Text Oriented Message Protocol (STOMP) [6].

STOMP позволяет получать уведомления всем зрителям трансляции о состоянии видео. Организатор трансляции перед её началом создаёт брокера сообщений, на который подписываются все участники просмотра. При любых действиях с изменением видео или заходом другого участника просмотра в комнату организатор посылает изменение таймкода и состояния видео всем пользователям, благодаря чему создаётся эффект совместного просмотра видео. На рис. 2 можно

увидеть диаграмму последовательностей взаимодействия при подключении нового пользователя к трансляции.

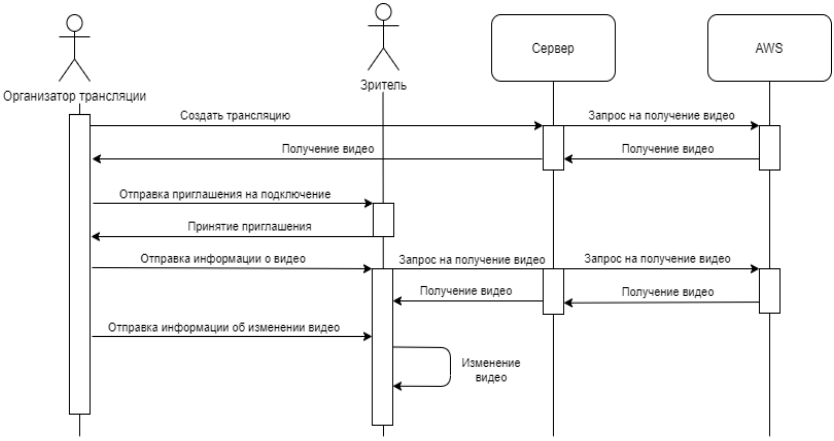


Рис. 2. Диаграмма последовательностей взаимодействия

Для реализации данного функционала были реализованы следующие функции для обработки событий видео и комнаты на плеере:

– onPlayVideoButtonClick – при срабатывании нажатия кнопки плей на видео отправляет сообщение о его состоянии;

– onPauseVideoButtonClick – при срабатывании нажатия паузы на видео отправляет сообщение о его состоянии;

– onVideoTimeSeeked – при срабатывании передвижения таймлайна на видео отправляет сообщение о его состоянии;

– connectToSession – срабатывает при первом подключении пользователя и подписывает его на события данной сессии и отправляет информацию о текущем состоянии видео.

– Результат работы можно увидеть на рис. 3.

Заключение

В результате проделанной работы было создано клиент-серверное приложение, предоставляющее пользователям возможность совместно просматривать загружаемые видео. Был описан и реализован функционал создания и передачи адаптивного видеопотока пользователям, который позволяет подстроиться к сети каждого пользователя по отдельности.

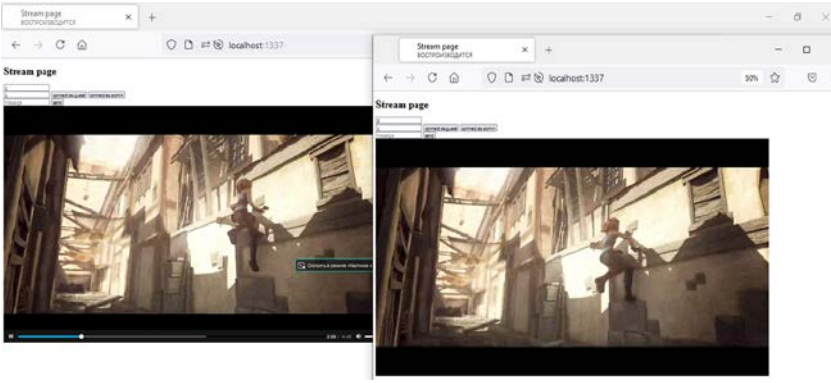


Рис. 3. Пример совместного просмотра

Список литературы

1. Описание HTTP Live Streaming [Электронный ресурс].– Режим доступа: https://web.archive.org/web/20130823145710/http://www.larryjordan.biz/app_bin/wordpress/archives/2369

2. Документация Spring Boot [Электронный ресурс]. – Режим доступа: https://spring.io/projects/spring-boot

3. Работа с Amazon S3 [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/ru/s3

4. Документация FFmpeg [Электронный ресурс]. – Режим доступа: https://ffmpeg.org/

5. Документация AWS SDK [Электронный ресурс]. – Режим доступа: https://docs.aws.amazon.com/sdk-for-java

6. Работа с WebSocket по STOMP [Электронный ресурс]. – Режим доступа: https://spring.io/guides/gs/messaging-stomp-websocket/

Разработка парсера для автоматизации деятельности Контрольно-счетной палаты Воронежской области

А В. Порядин

Студент бакалавриата

А. Г. Потапов

Студент бакалавриата

Т. Д. Братышев

Студент бакалавриата

С. И. Бутовецкая

Студент бакалавриата

А. Е. Меркулов

Студент бакалавриата

И. В. Храпов

Студент бакалавриата

Е. А. Копытина

Старший преподаватель

С. В. Борзунов

Доцент

И. В. Селютин

Председатель
Контрольно-счетной палаты
Воронежской области

А. А. Крыловецкий

Декан

Введение

В современном мире люди регулярно сталкиваются с большим количеством информации, которая может быть представлена в самых различных видах и которую требуется каким-либо образом обрабатывать. Однако в профессиональной среде объем данных, которые необходимо хранить и анализировать, растет в геометрической прогрессии, следовательно пользоваться уже такими инструментами, как Microsoft Excel или LibreOffice Calc становится слишком неэффективно, ведь большую часть работы приходится делать вручную, как показано в [1-4].

Объем информации, используемый Контрольно-счетной палатой Воронежской области, за последние годы многократно возрос. Это связано, в том числе с необходимостью проведения мониторинга реализации на территории региона национальных проектов, анализа формирования и реализации документов стратегического планирования Воронежской области и областного бюджета, усилением аналитической составляющей контрольной деятельности.

Для качественной и оперативной реализации полномочий контрольно-счетного органа необходима автоматизация ее деятельности. Одним из направлений такой автоматизации является создание реляционной базы данных.

Однако здесь появляется проблема переноса информации в базу, ведь делая это вручную, придется потратить слишком много времени. И именно тут могут помочь скрипты, автоматически переносящие данные по заданным условиям - парсеры, время работы которых значительно выше. Именно одному из таких парсеров посвящена эта статья. Описываемый далее проект нацелен на ускоренное формирование полноценной базы данных, исходя из информации, находящейся в таблицах Microsoft Excel, а потому и является крайне актуальным в профессиональной среде.

1. Постановка задачи

К описываемому в статье проекту по реализации скриптов-парсеров для создания реляционной базы данных, предназначенной для анализа формирования и реализации документов стратегического планирования Воронежской области, были выдвинуты следующие требования:

1. Чтение информации из таблиц формата «xlsx»;
2. Возможность подключения к базе данных напрямую и дальнейшее отправление SQL-запросов;
3. Автоматическое создание SQL-запроса и применение его результатов;

4. Правильное использование типов данных;
5. Правильное форматирование ячеек.

2. Реализация логики

Проект реализуется посредством создания тестовой базы данных на сервере ВГУ и загрузки в неё отношений, скрипты создания которых написаны на диалекте PostgreSQL.

В процессе работы были использованы следующие языки программирования:

6. Диалект PostgreSQL языка SQL. Использовался: а) для создания структуры будущей базы данных в виде совокупности схем и готовых отношений; б) для заполнения созданных отношений предварительно обработанными данными;

7. Python 3.9. Мощный объектно-ориентированный язык программирования, с помощью которого были написаны парсеры для перевода данных по созданной структуре из формата XLSX в формат SQL;

8. Для реализации проекта было использовано следующее ПО:

9. pgAdmin 4. Позволяет эффективно работать с базами данных на языке PostgreSQL. Была создана БД «project-cs», в которую были загружены обработанные отношения;

10. – PyCharm Professional. Среда разработки на языке Python, позволяющая создавать как скрипты-парсеры, так и SQL запросы, и помогающая работать с БД.

3. Реализация логики

Скрипт-парсер написан на объектно-ориентированном языке Python. Для реализации были использованы следующие библиотеки:

1. Openpyxl,
2. Openpyxl.utils,
3. Psycopg2.

Разработанный парсер взаимодействует с реляционной базой данных PostgreSQL посредством SQL-запросов.

Само написание скрипта логически делится на три этапа:

1. Модуль, инициализирующий подключение к .xlsx таблице и подключение к базе данных;

2. Классы, написанные под определенные таблицы в БД и упрощающие работу с SQL-запросами, которые составляются автоматически;

3. Сам скрипт-парсер, использующий вышеперечисленные этапы и перебирающий .xlsx таблицу по строкам и ячейкам.

Первый этап представлен следующими функциями в модуле, содержащийся в файле PyScripts/base/base_functions.py:

1. db_conn() - функция, инициализирующая подключение к реляционной базе данных и возвращающая cursor и connection;
2. xlsx_connect() - функция, инициализирующая подключение к .xlsx таблице и возвращающая её в виде списка строк, которые в свою очередь являются списком ячеек;
3. unmerge_all_cells() - функция, где реализована логика разбиения всех объединенных ячеек в .xlsx таблице, для более удобного чтения самих ячеек;
4. parser_init() - функция, объединяющая в себе все вышеперечисленные функции и возвращающая сразу cursor, connection и список строк с ячейками.

Второй этап представлен классами, которые упрощают работу с БД и автоматизируют многие процессы. Классы содержатся в папке PyScripts/TableClasses/PublicClasses:

1. BasicTables.py - универсальный класс базовых таблиц, которые повторяются в базе данных по своей структуре (id - числовое значение, title - наименование элемента) чаще всего;
2. Events.py, Gosprogram.py, GrowthPoints.py - классы таблиц, имеющие уникальные особенности, исходя из которых невозможно написать универсальный класс.

Третий этап являемся самим скриптом-парсером, находящимся в папке PyScripts/Parsers:

1. Plan_meropriatii - папка, содержащая парсеры под каждую страницу .xlsx таблицы «Плана мероприятий».
2. CUR - папка, содержащая парсеры под каждую .xlsx таблицу, связанную с «ЦУР».
3. IndustriesParsers - папка, содержащая парсеры под .xlsx таблицы, которые содержат данные об определенных отраслях.

Заключение

В данной статье была рассмотрена реализация парсера для автоматизации деятельности Контрольно-счетной палаты Воронежской области в сфере формирования и реализации региональных документов стратегического планирования. На данный момент в БД были загружены все базовые таблицы. Планируется модернизировать скрипты-парсеры и классы таблиц, расширив их функциональные возможности.

Список литературы

1. Kopytina, E. A. Smart Service for Medical Care of the Region's Population / E. A. Kopytina, A. V. Kopytin, F. A. Desyatirikov // Proceedings

of the 2021 IEEE International Conference "Quality Management, Transport and Information Security, Information Technologies", T and QM and IS 2021, Yaroslavl, 06–10 сентября 2021 года. – Yaroslavl, 2021. – P. 206-209. – DOI 10.1109/ITQMIS53292.2021.9642885. – EDN AGGPGB.

2. Кропачев, А. Н. Формирование отчета деятельности наркоконтроля по Воронежской области средствами офисных приложений / А. Н. Кропачев, Е. А. Копытина // Труды молодых учёных факультета компьютерных наук ВГУ: Сборник статей / Под редакцией Д. Н. Борисова. – Воронеж: Общество с ограниченной ответственностью "Вэлборн", 2021. – С. 80-84. – EDN IUBYHI.

3. 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. – Воронеж: Воронежский государственный университет, 2020. – С. 143-147. – EDN MPFXVD.

4. Копытина, Е. А. Бизнес-аналитика в libreoffice Calc: Учебно-методическое пособие / Е. А. Копытина, А. В. Копытин; Федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет». – Воронеж: Воронежский государственный университет, 2021. – 72 с. – EDN HOBRXW

Реализация проекта по созданию реляционной базы данных для Контрольно-счётной палаты Воронежской области

А. Г. Потапов

Студент бакалавриата

А. В. Порядин

Студент бакалавриата

Т. Д. Братышев

Студент бакалавриата

С. И. Бутовецкая

Студент бакалавриата

А. Е. Меркулов

Студент бакалавриата

И. В. Храпов

Студент бакалавриата

Е. А. Копытина

Старший преподаватель

С. В. Борзунов

Доцент

И. В. Селютин

Председатель
Контрольно-счетной палаты
Воронежской области

А. А. Крыловецкий

Декан

Введение

В современном мире люди привыкли регулярно сталкиваться с большим количеством информации, которая может быть представлена в самых различных видах и которую требуется каким-либо образом обрабатывать, как показано в [1-3]. И если в повседневной жизни для работы с этой информацией, как правило, вполне достаточно привычных и наглядных технических средств, таких как Microsoft Word или Microsoft Excel, то в профессиональной среде объёмы данных возрастают настолько сильно, что работать с ними вручную уже попросту не представляется возможным. На сегодняшний день автоматизация процесса обработки информации является необходимым условием эффективного внешнего государственного финансового контроля. Аналитическая составляющая работы контрольно-счетных органов становится все более востребованной, в том числе в связи с включением в состав полномочий органов внешнего контроля оценки реализуемости, рисков и результатов достижения целей социально-экономического развития субъекта РФ, предусмотренных документами стратегического планирования (стратегического аудита).

Для проведения оперативного и качественного анализа значительного объема информации, содержащейся в документах стратегического планирования Воронежской области и отчетах об их реализации, было принято решение о создании реляционной базы данных, способной структурировать практически неограниченно большие объёмы информации и автоматизировать большинство трудоёмких процессов их обработки.

1. Постановка задачи

К описываемому в статье проекту по созданию реляционной БД были выдвинуты следующие требования:

– Проведение полноценного анализа всего массива имеющихся данных, включая выявление зависимостей между ними (данных, содержащихся в Стратегии социально-экономического развития Воронежской области на период до 2035 года, Плане мероприятий по реализации Стратегии, государственных программах Воронежской области, региональных проектах Воронежской области и отчетах об их реализации за ряд лет);

– Написание SQL-запросов для создания всех возможных таблиц из предоставленного массива;

– Загрузка обработанных и структурированных данных в СУБД;

– Организация возможности оперативной загрузки вновь появляющихся данных;

– Создание пользовательского интерфейса.

2. Средства реализации

Проект реализуется посредством создания тестовой базы данных на сервере ВГУ и загрузки в неё отношений, скрипты создания которых написаны на диалекте PostgreSQL.

В процессе работы были использованы следующие языки программирования:

– Диалект PostgreSQL языка SQL. Использовался: а) для создания структуры будущей базы данных в виде совокупности схем и готовых отношений; б) для заполнения созданных отношений предварительно обработанными данными;

– Python 3.8. Мощный объектно-ориентированный язык программирования, с помощью которого были написаны парсеры для перевода данных по созданной структуре из формата XLSX в формат SQL.

– Для реализации проекта было использовано следующее ПО:

– pgAdmin 4. Позволяет эффективно работать с базами данных на языке PostgreSQL. Была создана БД «project-cs», в которую были загружены обработанные отношения;

– draw.io. Бесплатная альтернатива офисному продукту «Microsoft Visio» для создания ER-диаграмм;

– Microsoft Excel. Необходим для анализа исходных таблиц, имеющих, как правило, формат XLSX;

– Microsoft Word. Необходим для работы с документацией, объясняющей структуру исходных данных, а также для анализа некоторых из таблиц, изначально представленных в формате DOCX (впоследствии были с помощью Python-скрипта переведены в формат XLSX);

– gedit Text Editor. Простой и удобный редактор кода для различных языков программирования (в том числе, SQL);

– PyCharm. Среда разработки на языке Python.

3. Реализация

Исходные данные представлены в виде совокупности папок с показателями для различных госпрограмм, находящихся в большом массиве таблиц. Всего имеется 25 государственных программ Воронежской области, для каждой из которых представлены данные для 2016 – 2020 гг. Показатели для каждого года анализируются в 6-7 различных таблицах.

В первую очередь, потребовалось тщательно проанализировать все имеющиеся данные и составить для каждой Excel-таблицы

ER-диаграмму, отражающую связь данных внутри таблицы друг с другом.

Примеры ER-диаграмм для нескольких таблиц представлены на рис. 1-2.

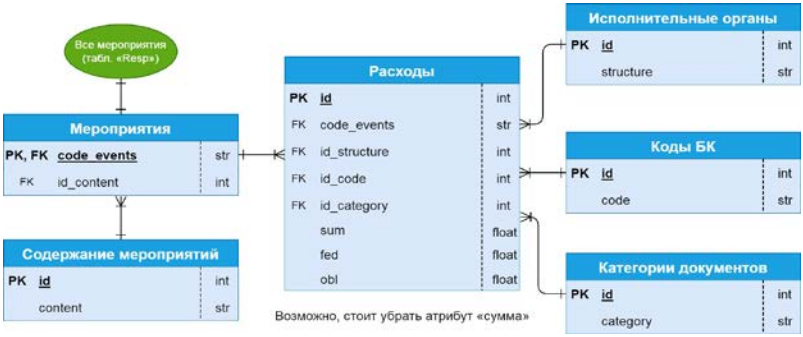


Рис. 1. ER-диаграмма для таблицы «ГРБС»

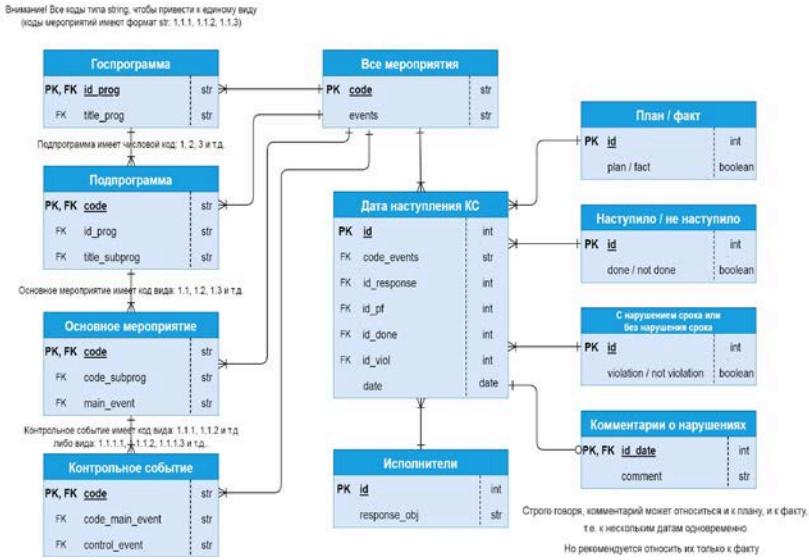


Рис. 2. ER-диаграмма для таблицы «Контрольные события»

Для каждой таблицы требовалось написать SQL-запрос, создающий необходимое количество отношений и учитывающий их взаимозависимость.

Некоторые таблицы построены по одним и тем же шаблонам, но многие из них имеют свои индивидуальные особенности. Поэтому

нужно было составить сравнительную Excel-таблицу для всех госпрограмм и для каждого года, чтобы учесть все индивидуальные особенности каждой таблицы и написать для каждой из них корректный SQL-запрос по вновь составленной уточнённой ER-диаграмме (см. рис. 3).

Развитие транспортной системы	2016*	+ (с мероприятиями)	+ (с ФПСР)***	+	+	+	+**
	2017	↑	↑	+ (без Дорожного фонда)			
	2018	+ (с мероприятиями)	+ (с ФПСР)(5)	+	+	+	+ (Обр. 2017)
	2019	↑	↑	+	+	+	↑
	2020	+ (с мероприятия(6)	↑	+	+	+	↑
	Примечание	*Данные для ОБЕСПЕЧЕНИЯ ОБЩЕСТВЕННОГО ПОРЯДКА. Подпрограмма №1, Основное мероприятие 1. Но есть табл. 15 - Экономия для транспортной системы за 2015 год. (V)** Как в «Развитии предпринимательства» за 2016 г., но вместо основных мероприятий просто мероприятия. Суммы продублированы для осн. мероприятия, подпрограммы и госпрограммы. ***Иногда написано явно «Отклонений нет». Вносить это значение в отношение с отклонениями нет смысла. **** В НАСТОЯЩЕЙ транспортной системе есть пометки «Дорожный фонд» для некоторых мероприятий. Добавить отношение «Дорожный фонд». Контрольные события: +, встречается формат id: 1.1.. ; 1.3.. (5)в отклонениях есть кортежи, где написано "Отклонений нет". (6) в "Мероприятиях" идет "точка" после номера *.*.*.					
Развитие физкультуры	2016	+ (без мероприятий)	+ (с ФПСР)***/****	+	+	+	+
	2017	↑	+ (с ФПСР)****	+	+	+	↑ (без ДС)*
	2018	↑	↑	+	+	+	↑ (без закр - ия)
	2019	↑	↑	+	+	+	↑
	2020	↑	+ (без ФПСР)****	+	+	+	↑**
		Как в «Развитии предпринимательства» за 2016 г. *Жёстко закреплена большая часть таблицы. Тяжело просматривать, нужно следить за тем, чтобы ничего не упустить. После 2017 г. разделение на фед и обл. **Называется «районы». Возможно, стоить упразднить суммы для подпрограмм и госпрограммы (и не только для данной отрасли). Контрольные события: +. *** "Основные мероприятия" #1. точка и не везде + Осн. Мероприятия повторяются; **** есть обоснование					

Рис. 3. Фрагмент сравнительной таблицы по госпрограммам

Результатом обработки всех таблиц является набор из 26 схем, загруженных в СУБД, каждая из которых соответствует определённой госпрограмме (см. рис. 4).

Рис. 4. Схемы для госпрограмм

В каждой схеме содержится от 76 до 85 отношений. Как правило, одной таблице в формате Excel соответствует от 1 до 10 отношений (не считая внешних зависимостей).

Заключение

В данной статье была рассмотрен первоначальный этап реализации проектирования базы данных для Контрольно-счётной палаты Воронежской области. На данный момент в СУБД было загружено 1976 отношений. В дальнейшем планируется модернизировать базу данных, расширять её функциональность, включать информацию других документов стратегического планирования Воронежской области.

Список литературы

1. Kopytina, E. A. Smart Service for Medical Care of the Region's Population / E. A. Kopytina, A. V. Kopytin, F. A. Desyatirikov // Proceedings of the 2021 IEEE International Conference "Quality Management, Transport and Information Security, Information Technologies", T and QM and IS 2021, Yaroslavl, 06–10 сентября 2021 года. – Yaroslavl, 2021. – P. 206-209. – DOI 10.1109/ITQMIS53292.2021.9642885. – EDN AGGPGB.

2. Кропачев, А. Н. Формирование отчета деятельности наркоконтроля по Воронежской области средствами офисных приложений / А. Н. Кропачев, Е. А. Копытина // Труды молодых учёных факультета компьютерных наук ВГУ: Сборник статей / Под редакцией Д.Н. Борисова. – Воронеж: Общество с ограниченной ответственностью "Вэлборн", 2021. – С. 80-84. – EDN IUBYHI.

3. 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. – Воронеж: Воронежский государственный университет, 2020. – С. 143-147. – EDN MPFXVD.

Разработка библиотеки для единообразной фильтрации данных

Е. Д. Проскуряков

Студент бакалавриата

А. И. Чекмарев

Старший преподаватель

Введение

В последнее время особой популярностью пользуется микросервисная архитектура, которая подразумевает разделение слоя бизнес-логики приложения на некоторое количество маленьких сервисов, взаимодействующих друг с другом и наделенных определенной функциональностью.

Бывают такие архитектуры, в которых предусмотрено использование нескольких различных СУБД для разных нужд. Например, использование классической реляционной базы данных для хранения основной информации и использование документно-ориентированных СУБД для хранения иерархических структур данных. Каждая из этих СУБД может быть подключена к разным сервисам в микросервисной архитектуре.

Обычно для пользователя пытаются реализовать концепцию прозрачных данных, то есть он имеет доступ ко всем данным, к которым должен иметь доступ, независимо от того, где и в каком виде они хранятся. То есть необходимо, чтобы клиентская часть позволяла одинаково просто отображать данные независимо от того, какой сервис их предоставляет и из какого хранилища он их извлекает. Следовательно, было бы удобно, если бы и клиентская часть этого не знала, и для всех запросов на извлечение, фильтрацию и сортировку данных отправляла одинаковые запросы независимо от того, в какой сервис они придут и для какой базы данных предназначены. Именно для этого было принято решение разработать библиотеку для единообразной фильтрации данных.

1. Цель работы

Целью данной работы является проектирование и реализация библиотеки для фильтрации, позволяющей обрабатывать запросы на выборку и сортировку данных по единому формату.

К разрабатываемой библиотеке выдвигаются следующие требования:

– Единый внешний интерфейс для всех сервисов, использующих библиотеку;

– Массовость, позволяющая использовать данную библиотеку во всех приложениях с микросервисной архитектурой;

– Модульность, позволяющая в пределах сервиса использовать только тот модуль, который подходит под используемую СУБД;

– Расширяемость, позволяющая добавлять к библиотеке модули под новые виды СУБД.

2. Реализация

Библиотека состоит из нескольких модулей [1]:

– Web-filter-api – модуль с моделями, которые будут использоваться во внешнем интерфейсе и делать точку входа единообразной и универсальной;

– Web-filter-jpa – модуль, который переводит пришедший запрос в запрос к реляционной базе данных;

– Web-filter-mongo – модуль, который переводит пришедший запрос в запрос к документно-ориентированной базе MongoDB.

На рис. 1 продемонстрирована диаграмма классов-фильтров, которые хранятся в модуле внешнего представления. Был введен единый универсальный интерфейс Filter<T>, который реализуют классы-фильтры под разные задачи выборки. Существуют фильтры по регулярному выражению, по дате, фильтры сравнений и много других. Данный список можно расширять и добавлять более узкоспециализированные фильтры с более тяжелой логикой работы.

3. Архитектура приложения

На рис. 2 продемонстрирована возможная архитектура приложения, использующего библиотеку фильтрации. В данной архитектуре есть один или несколько backend сервисов, работающих с СУБД PostgreSQL, и есть один или несколько сервисов, работающих с СУБД MongoDB. Каждый из этих сервисов разделен на модуль с внешним интерфейсом, на модуль работы с базой данных, и на модули, отвечающие за логику, которые тут собраны в единый абстрактный модуль «core».

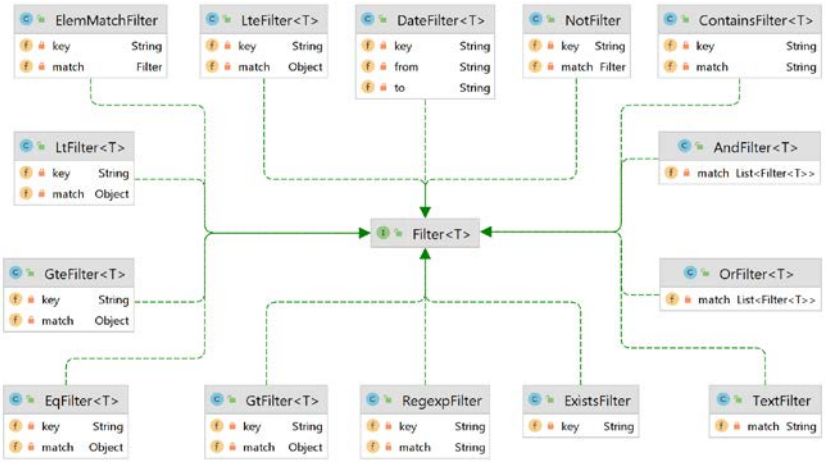


Рис. 1. Диаграмма классов Filter

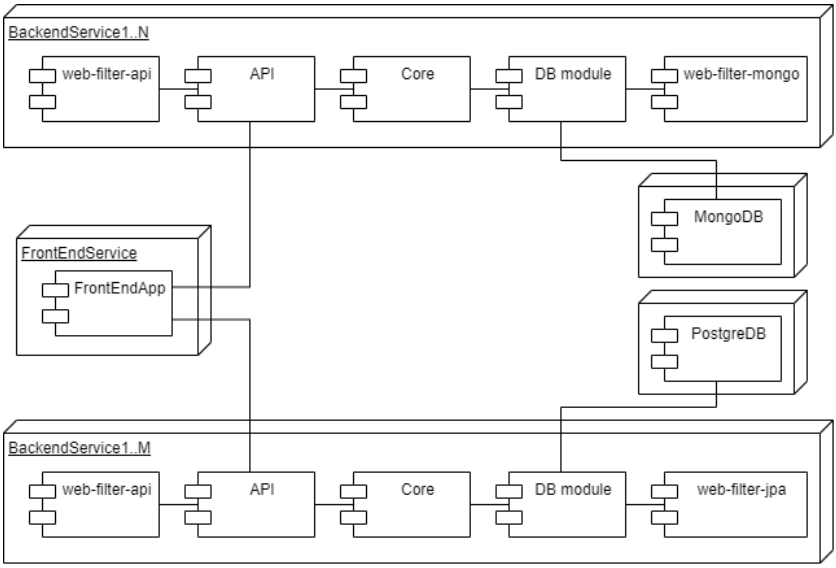


Рис. 2. Пример возможной архитектуры, использующей библиотеку для фильтрации

В сервисы внедряется модуль web-filter-api, который присоединяется к модулю с внешним интерфейсом и расширяет его

возможности, предоставляя модели для фильтрации данных. Во всех сервисах этот модуль одинаковый, соответственно они способны обрабатывать одинаковые запросы.

К сервисам, которые работают с СУБД PostgreSQL добавлен модуль web-filter-jpa, который используется в модуле работы с базой данных. В нем реализовано преобразование запроса, пришедшего с внешнего интерфейса, в конструкцию из предикатов, которые понимает spring-data-jpa [2]. В сервисах, работающих с СУБД MongoDB, ситуация аналогичная. Модуль web-filter-mongo преобразует запрос в конструкцию из bson элементов, которая делает выборку из документно-ориентированной базы данных [3].

При необходимости перевести сервис на другое хранилище, достаточно будет подключить другой реализованный модуль или реализовать новый модуль по работе с новым хранилищем. При этом, внешний интерфейс никак не изменится.

4. Использование

На рис. 3 приводится диаграмма классов SearchParams и SearchResult, которые используются при реализации конечной точки в внешнем интерфейсе.

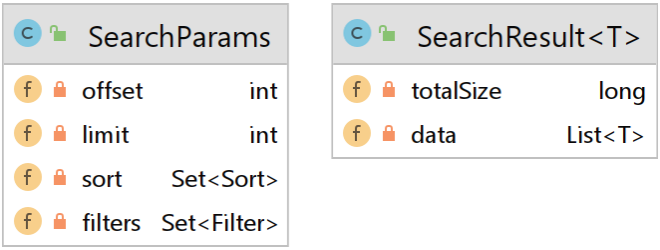


Рис. 3. Модель запроса данных и результата выборки

На вход сервисы присылается запрос, содержащий json тело, которое десериализуется в SearchParams [4]. В листинге продемонстрирован пример такого запроса. Дальше этот запрос передается в другой модуль библиотеки, где он преобразуется в запрос к базе данных, после чего результат оборачивается в SearchResult и возвращается пользователю.

Листинг

Пример запроса на фильтрацию

```
{
    "offset":0,
    "limit":100,
    "sort":[{"key":"height","direction":"DESC"}],
    "filters":[
        {"@type":"date","key":"birthday","from":"2002-02-
30T00:00:00.000Z","to":"2002-03-31T00:00:00.000Z"},
        {"@type":"gt","key":"height","match":"190"}
    ]
}
```

Данный запрос написан для получения сущности User, диаграмма класса которого предоставлена на рис. 4. В запросе, в качестве ключа, передается путь до поля сущности, а дальше, в зависимости от фильтра, передается или желаемое значение, или регулярное выражение, или, как показано в листинге, два значения from и to для фильтрации по промежутку дат.

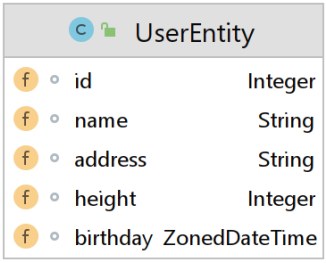


Рис. 4. Класс User

Заключение

Результатом выполнения работы является реализованная библиотека, предоставляющая единый внешний интерфейс получения данных, и позволяющая отделить работу с хранилищем по выборке этих данных в независимый модуль.

Данная библиотека является оптимальным решением для приложений с микросервисной архитектурой, которые используют несколько различных хранилищ или находятся на стадии прототипирования. В силу того, что в последнее время таких приложений становится больше, и все чаще вместо классической реляционной базы данных стали использовать документно-ориентированные или аналитические, то данная библиотека будет востребована.

Список литературы

1. Документация по Maven [Электронный ресурс]. – Режим доступа: https://maven.apache.org/pom.html
2. Документация по работе с JPA Criteria API [Электронный ресурс]. – Режим доступа: https://docs.jboss.org/hibernate/jpa/
3. Документация по работе с фильтрацией в MongoDB из SpringBoot приложения [Электронный ресурс]. – Режим доступа: https://docs.spring.io/spring-data/mongodb/docs/current/reference/html/
4. Документация по REST API [Электронный ресурс]. – Режим доступа: https://www.restapitutorial.com/

История развития цифровых программных синтезаторов

Н. А. Селиверстов

Студент бакалавриата

А. А. Вахтин

Доцент

Введение

В современном мире музыкальная индустрия, а также индустрия работы со звуком играют важную роль. Профессиональные музыканты и музыканты-любители, меломаны, крупные студии, занимающиеся разработкой игр и созданием фильмов, пользователи мобильных телефонов и персональных компьютеров формируют большой спрос, который порождает развитие аппаратных и программных средств для создания, записи, обработки и воспроизведения звука. Действительно, практически все современные смартфоны или компьютеры содержат звуковые платы, способные качественно воспроизводить звук, существует большое количество программ и приложений для работы со звуком, а многочисленные стандарты в этой области актуальны на протяжении многих лет и продолжают поддерживаться.

Востребованность работы со звуком, в частности средств для синтеза музыки и звука, делает перспективной разработку новых программных средств и развитие старых. Множество крупных компаний, образованных годы и десятилетия назад, продолжают успешно существовать, вместе с этим появляются новые, получающие внимание и признание публики.

Одним из направлений музыкальной и звуковой индустрии является разработка синтезаторов звука. Данные инструменты, появившись в прошлом веке, заняли важное место в музыке и звуковом дизайне. С развитием цифровых, а позже и компьютерных технологий большое распространение получили цифровые программные синтезаторы – приложения, осуществляющие генерацию и обработку цифровых звуковых сигналов.

Чтобы понять роль, которую сыграли синтезаторы в становлении компьютерного звука, а также выявить закономерности развития программных синтезаторов, необходимо рассмотреть историю развития

синтезаторов от появления первых аналоговых моделей до настоящего времени.

1. Развитие аналоговых средств синтеза звука

Развитие искусства в целом и музыки в частности в конце 19-го и начале 20-го веков поставило перед композиторами и изобретателями вопрос создания новых музыкальных инструментов. Одновременно с этим начали быстро развиваться электронные технологии. Под действием этих факторов были разработаны различные электронные музыкальные инструменты, например, телармониум, построенный Таддеусом Кэхиллом в 1897 году, и терменвокс, созданный советским изобретателем Львом Терменом в 1920 году [1].

Несмотря на революционность, ранние электронные инструменты не получили широкого распространения. Одной из причин этого послужило отсутствие удобных средств записи звука [2]. Усовершенствование магнитофонов в 40-50-х годах дало толчок для новых экспериментов со звуком. Студия Карлхайнца Штонкхаузена, основанная на немецком радио в начале 50-х годов, была оборудована генераторами синусоидальных тонов и шумов, модуляторами и фильтрами. На этой студии было записано несколько музыкальных композиций, которые получили мировую популярность и привлекли внимание к инструментам и методам, использовавшимся при их создании [1].

В 1957 году Гарри Олсон и Герберт Белар, получив грант Рокфеллера в размере 175000 долларов, разработали синтезатор RCA Mark II, который занимал целое помещение и работал при помощи ламповой схемы. Данный синтезатор стал первым синтезатором, высота звука в котором управлялась при помощи напряжения. [3]

Следующим важным событием в развитии аналоговых синтезаторов стал выпуск первого коммерческого аналогового синтезатора Moog. Данный инструмент, разработанный Робертом Мугом в 1964 году, состоял из модулей, которые генерировали и обрабатывали сигналы и были соединены между собой кабелями. Синтезатор Moog, в отличие от RCA Mark II, работал при помощи транзисторов и был значительно меньше. Данный синтезатор быстро обрел высокую популярность и практически стал индустриальным стандартом. В 1970 году была разработана уменьшенная версия синтезатора Moog под названием Minimoog. Данная версия стала первым аналоговым синтезатором, продаваемым в розничных магазинах [4].

В дальнейшем было выпущено большое количество аналоговых синтезаторов от различных компаний, которые следовали уже

установленным принципам и развивали в первую очередь доступность и удобство пользования.

2. Развитие цифровых средств синтеза звука

Развитие программных цифровых средств синтеза звука начинается с исследований Макса Мэтьюса в Bell Labs. Работая на мейнфрейме, Мэтьюс создал первую схему преобразования цифровых данных в аналоговый звук, в результате чего был разработан язык Music I. Это дало начало развитию семейства языков прямого цифрового синтеза Music-N. Также Мэтьюс впервые применил теорию сэмплирования к компьютерным алгоритмам [5].

Программы, написанные на языках прямого синтеза, можно назвать низкоуровневыми программными синтезаторами, так как результатом их работы является сгенерированный и обработанный цифровой звуковой сигнал.

Следующим значимым событием в области цифрового синтеза сигналов стало изобретение Джоном Чоунингом цифрового частотно-модуляционного синтеза (FM-синтез). Данный алгоритм был лицензирован компанией Yamaha в 1973 году [6].

В 1970-х годах появилось несколько моделей коммерческих цифровых синтезаторов, например, сэмплерный синтезатор Fairlight CMI и использующий FM-синтез New England Digital Synclavier II, которые получили определенную популярность, несмотря на высокую стоимость [7, 8].

В 1983 году был принят стандарт MIDI, определяющий взаимодействие цифровых музыкальных инструментов между собой, а также позволяющий подключать их к компьютеру и записывать звуковые сигналы в стандартизированном формате [9].

После продолжительных работ по улучшению и адаптации алгоритма FM-синтеза Yamaha выпустила в 1983 году синтезатор DX-7, работающий на микрочипе. Данный синтезатор стоил около 2000 долларов, что вместе с удобством использования, широкими возможностями по синтезу звука и его высокому качеству, позволило продать 200000 экземпляров DX-7, сделав его одним из самых продаваемых синтезаторов в истории [6].

В конце 80-х годов стало возможно записывать и использовать высококачественные образцы звучания реальных инструментов. Это привело к росту популярности синтезаторов, основанных на воспроизведении и обработке сэмплов.

Успех синтезаторов Yamaha привел к популяризации цифровых синтезаторов, породил множество последователей, а также заставил производителей звуковых карт для персональных компьютеров и

игровых консолей обратить внимание на чипы, осуществляющие синтез звука в цифровых синтезаторах.

3. Развитие компьютерных средств синтеза звука

Первым устройством для воспроизведения звука на персональном компьютере стал PC Speaker, который был установлен в компьютере IBM PC в 1981 году. PC Speaker был предназначен для вывода системных сигналов и позволял генерировать исключительно однобитные квадратные сигналы. Несмотря на примитивность устройства, разработчики начали писать программы, осуществляющие генерацию довольно сложных сигналов при его помощи [10].

До середины 90-х годов наблюдалось развитие звуковых карт. Карты того периода имели раздельные блоки синтеза звуковых сигналов и ЦАП/АЦП. Популярная звуковая карта Sound Blaster, выпущенная компанией Creative в 1989 году, имела чип Yamaha YM3812, который осуществлял FM-синтез и был разработан компанией Yamaha для использования в своих цифровых синтезаторах [11]. Программные синтезаторы того времени уже представляли собой приложения, обладающие графическим интерфейсом и предоставляющие пользователям широкие возможности для генерации и обработки звуковых сигналов, например, синтезатор Turbosynth для компьютера Macintosh, выпущенный в 1988 году, предоставил пользователям возможность работы со звуковыми модулями [12]. Однако отсутствие программных и аппаратных стандартов приводило к тому, что разработка синтезаторов была осложнена необходимостью изучать особенности конкретных звуковых плат и операционных систем, а разработанные приложения работали на весьма ограниченном количестве устройств.

К середине 90-х годов звуковые карты развились до практически современного вида. В то же время начали появляться стандарты Audio API, а также развивались средства программирования, благодаря чему создание программных синтезаторов значительно упростилось и начали появляться популярные приложения для компьютеров с операционной системой Windows, ставшей самой популярной платформой персональных компьютеров. Примером такого синтезатора является Microsoft GS Software Wavetable Synthesizer, впервые представленный для Windows 98.

Значимыми событиями в истории программных синтезаторов стали выпуск компанией Steinberg спецификаций VST в 1996 году и их обновление до версии 2.0 в 1999 году, которое позволило получать MIDI-данные и создавать плагины формата VSTi (Virtual Studio Technology Instrument), осуществляющие генерацию сигнала под

управлением MIDI-устройств [13]. В настоящее время многие популярные и востребованные программные синтезаторы представляют собой VST плагины. Для их создания можно использовать программирование на чистом VST SDK или воспользоваться фреймворками, упрощающими процесс разработки, например, фреймворком WDL-OL для C++.

В настоящее время существует большое количество как профессиональных программных синтезаторов, применяемых при создании звукового дизайна в музыке или кино, так и синтезаторов, ориентированных на использование для любительской звукозаписи.

Одновременно с ростом аудитории программных синтезаторов наблюдается значительное упрощение как создания, так и использования программных синтезаторов. Примером проявления этих закономерностей служит сайт Playtronica, который предоставляет доступ к большому количеству веб-синтезаторов, загружаемых пользователями.

Упрощение разработки связано с развитием стандартов компьютерного звука, а также средств разработки программных синтезаторов: библиотек и фреймворков для различных языков программирования, сред визуального программирования и конструкторов синтезаторов из готовых модулей.

Упрощение использования программных синтезаторов обусловлено теми же факторами: стандартизация позволяет использовать синтезаторы практически на всех персональных компьютерах и даже смартфонах, а упрощение разработки – создавать синтезаторы в виде stand-alone приложений, плагинов или наиболее доступных web-приложений.

Также существует несколько направлений развития функциональной части программных синтезаторов: экспериментирование с алгоритмами генерации и обработки сигналов для получения необычных звуков, моделирование алгоритмов и особенностей функционирования классических аналоговых и цифровых синтезаторов, создание синтезаторов, основанных на глубоком машинном обучении.

Заключение

В данной статье была рассмотрена история развития синтезаторов от первых аналоговых моделей до используемых сегодня программных синтезаторов. Анализ данных позволил выявить основные направления и закономерности развития синтезаторов: упрощение производства, увеличение доступности и простоты использования, увеличение областей применения. Основываясь на полученных результатах, можно

сделать выводы о востребованности и перспективности конкретных программных синтезаторов.

Список литературы

1. Manning, P. Electronic and Computer Music / P. Manning. – Oxford : Clarendon Press, 1985. – 291 p.
2. Morgan, R. Twentieth-century music / R. Morgan. – New York : Norton, 1991. – 554 p.
3. Olson, H. Music, physics and engineering / H. Olson. – New York : Dover Publications, 1967. – 460 p.
4. Pinch, T. Analog days : the invention and impact of the Moog synthesizer / T. Pinch, F. Trocco. – Cambridge (USA) : Harvard University Press, 2002. – 368 p.
5. Bogdanov, V. All music guide to electronica : the definitive guide to electronic music / V. Bogdanov. – San Francisco : Backbeat Books, 2001. – 707 p.
6. Holmes, T. Electronic and experimental music : technology, music, and culture / T. Holmes. – New York : Routledge, 2008. – 462 p.
7. Synclavier early history [Электронный ресурс]. – Режим доступа : http://www.500sound.com/synclavierhistory.html
8. Fairlight – The Whole Story [Электронный ресурс]. – Режим доступа : http://www.anerd.com/fairlight/fairlightstory.htm
9. MIDI History : Chapter 6-MIDI Is Born 1980-1983 [Электронный ресурс]. – Режим доступа: https://www.midi.org/midi-articles/midi-history-chapter-6-midi-is-born-1980-1983
10. Программный синтез звука на ранних персональных компьютерах. Часть 2 : Однобитный синтез [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/348192
11. Sound Blaster 1.0 [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Sound_Blaster#Sound_Blaster_1.0,_CT1310,_CT1320A,_CT1320B
12. Rich, R. Digidesign Turbosynth / R. Rich // Music Technology. – Chatsworth : Sep. 1988. – P. 88-92.
13. VST [Электронный ресурс]. – Режим доступа : https://en.wikipedia.org/wiki/Virtual_Studio_Technology

Автоматическое улучшение геометрии, полученной методом фотограмметрии

В. В. Соболев

Студент бакалавриата

С. В. Власов

Доцент

Введение

С каждым годом разрабатываются новые технологии, с помощью которых можно переносить объекты из реального мира в форму 3д моделей с тектурами.

Метод фотограмметрии обрел популярность сравнительно недавно, так как до этого были популярны не оптические методы получения данных, оборудование было дорогим, а программное обеспечение для данного метода практически отсутствовало.

На данный момент практически каждый может получить 3д модель реального объекта из обычных фотографий по нажатию одной кнопки, однако качество таких моделей не позволяет использовать такие данные в конечном продукте, будь это компьютерная игра, проект по архитектурной визуализации или что угодно, требующее оптимизированной топологии и качественных текстур объектов. Решение, описанное в данной статье предлагает полуавтоматический метод обработки фотограмметрических данных путем подмены участков исходной модели на другие похожие объекты, которые были заранее подготовлены. Конкретно решается задача позиционирования таких объектов.

Применение численных методов оптимизации для улучшения геометрии

Подмена объектов начинается с их локализации, для этого можно использовать существующие методы определения объектов на фотографии. В данном примере это стол, стул и цветы в плошках (рис. 1). На этом же этапе можно классифицировать тип объекта для улучшения

Рис. 1. Исходная геометрия

Далее в зависимости от типа объекта необходимо выбрать модель высокого качества, на которую и будет подменяться локализованный объект. Для этого можно использовать библиотеки моделей из интернета, предварительно установив соответствие каждой модели соответствующий тип объекта, который она будет подменять.

Так как модели кардинально отличаются по топологии для точной подмены будем использовать итеративный алгоритм ближайших точек [2] для задания функции ошибки и метод Гаусса-Ньютона [3] для оптимизации этой функции. Пример тестового приложения на Qt и C++, оптимизированная модель ноги совмещается с моделью, полученной методом фотограмметрии (рис. 2).

Перед использованием метода Гаусса-Ньютона [3] был протестирован метод градиентного спуска [1], который показал себя хуже по показателям скорости сходимости и устойчивости решения при разных моделях и начальных условиях.

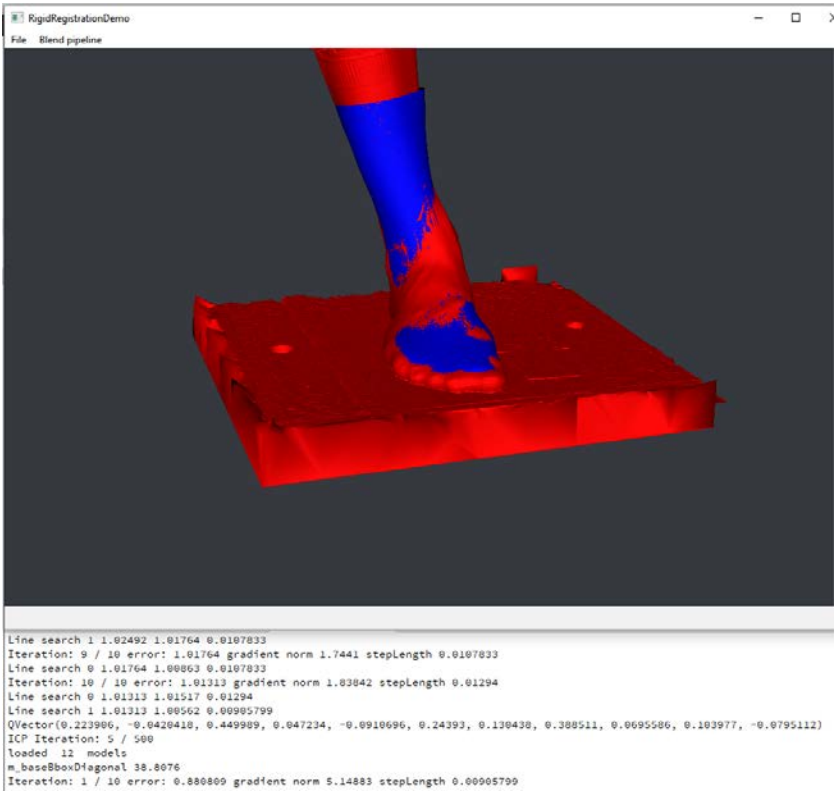


Рис. 2. Заранее подготовленная модель(синяя) совмещается с моделью, полученной методом фотограмметрии(красная)

Так же в данном примере реализован переход между оптимизированными моделями одного типа, что позволяет выбрать наиболее топологически близкую из библиотеки. Для этого на этапе алгоритма итеративных ближайших точек достаточно добавить вершины нескольких моделей.

Возможна работа в более грубом режиме, при которой изменяется только положение всего объекта целиком, без изменения координат каждой вершины.

После выполнения алгоритма получаем объект, который достаточно точно вписан в исходную геометрию, на котором есть качественные текстуры и материалы, а так же что немаловажно он

отделен от основной геометрии, что удобно для ручной настройки, если необходимость в ней всё же возникнет.

В целях ускорения разработки и тестирования в примере использовались модели сканов ног, однако реальные библиотеки моделей предлагают достаточно широкий выбор моделей для различных сцен.

Заключение

В результате был сформулирован алгоритм по применению численных методов оптимизации для улучшения геометрии, полученной методом фотосканирования. Приведены 2 варианта реализации, один со смешением различных моделей, другой с изменением расположения одной модели, каждый из которых имеет свое применение

Список литературы

1. Современные численные методы оптимизации. Метод универсального градиентного спуска : учебное пособие / А. В. Гасников. – М. : МФТИ, 2018. – 291 с. – Изд. 2-е, доп. ISBN 978-5-7417-0667-1

2. Arxiv [Электронный ресурс] – Режим доступа : https://arxiv.org/abs/2007.07627v2

3. Rudnyi E.B. Statistical model of systematic errors: linear error model // Chemometrics and Intelligent Laboratory Systems. – 1996. – V. 34. – P. 41-54.

Разработка математического и программного обеспечения для реконструкции поврежденных фрагментов изображения

А. А. Соловьев

Студент бакалавриата

М. А. Дрюченко

Доцент

Введение

В современном мире многие вещи переходят в цифровой формат, изображения не исключение, вместе с этим на свет появляются различные проблемы, связанные с ними, например: восстановление фотографий после их оцифровки, то есть удаление царапин и дефектов, которые возникают из-за старости или не аккуратного хранения фотографий, удаление текста с изображения, или обработка фото фотографами или дизайнерами в специализированных программах и т.д.

Задача восстановления изображения в поражённых областях заключается в том, чтобы заполнить их информацией, взятой из остальных частей изображения. Раньше эта задача стояла перед художниками и реставраторами, которые восстанавливали картины путём устранения трещин и различных дефектов. Сейчас на место картин пришли цифровые изображения, которые тоже содержат различные дефекты, а с развитием программ по обработке изображений, задача дополнения изображений стала актуальной, ведь каждый хочет убрать с заднего фона фотографии того человека, который закрывает прекрасный вид. Это сложная задача для фотографа или человека, который занимается обработкой изображений, их работу можно упростить, автоматизировав эту задачу. И так на место реставраторов и фотографов пришли различные методы дополнения, закраски частей изображения.

В данной работе я проанализирую различные подходы к проблеме восстановления изображений, реализую один из рассматриваемых алгоритмов.

1. Классификация методов восстановления

На настоящий момент существуют несколько подходов к решению задачи закраски фрагментов изображения:

– Методы реконструкции, привлекающие аппарат дифференциальных уравнений в частных производных
– Методы на основе синтеза текстур
– Гибридные методы

Также можно выделить подход с использованием нейросетей.

2. Методы, основанные на решении дифференциальных уравнений в частных производных

Базовый подход к задаче закраски отсутствующих частей изображений основан на диффузионных методах, в которых процесс диффузии используется для заполнения поражённых участков изображения.

Одним из первых методов, использовавших данную технику, был метод, описанный в работе [3]. Идеи для реализации данного метода были заимствованы из теоретической теплофизики. Алгоритм выполняет заполнение выделенных пользователем фрагментов, путем, использования элементов в небольшой полосе, которые окружают целевую область. Для восстановления границ объектов на изображении применяют, итеративный процесс продления структурных границ областей объектов на изображении внутрь отсутствующей области (рис. 1), это достигается за счёт использования изофот.

$$I_t = \nabla(\Delta I) \cdot \vec{N}, \quad (1)$$

где I – интенсивность пикселя изображения, N – вектор, указывающий направление изофоты и определяющийся как $\vec{N} = \left[-I_y, I_x\right]$.

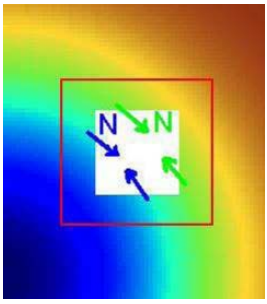
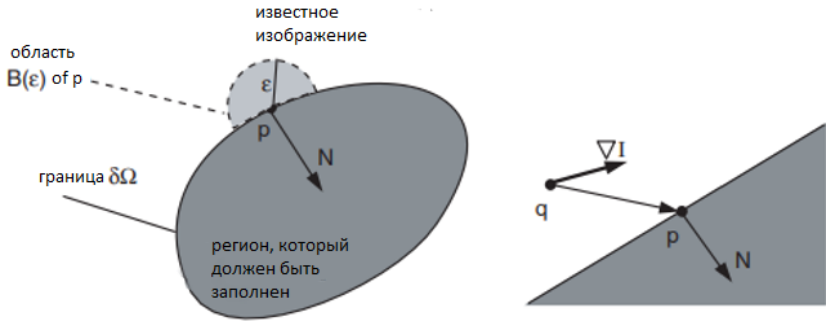


Рис. 1. Иллюстрация работы метода

Одной из реализаций данного подхода является метод Телеа, основанный на распространении гладкости изображения по градиенту изображения. Гладкость изображения оценивается как средневзвешенное значение пикселей в окрестности той точки, которую

мы хотим нарисовать. Для закраски утерянного фрагмента, изображение обрабатывается как наборы уровней и используется метод быстрого прохода (FMM) для продвижения границы в глубь закрашиваемой области (рис. 2).



а б

Рис. 2. Принцип работы метода Телеа

Принцип метода заключается в следующем алгоритме: выбирается точка p на границе области закраски (небольшая линия, которую мы и будем двигать в нужную нам область фрагмента с целью закраски), далее берется район $B(\varepsilon)$ с радиусом ε на известной части нашего изображения, и получаем p как функцию от всех q в $B(\varepsilon)$,

$$I(p) = I(q) + \nabla I(q)(p - q) \qquad (2)$$

Чтобы заполнить всю пораженную область Ω мы итеративно применяем уравнение 2 ко всем дискретным пикселям на границе $\delta\Omega$, продвигая её вглубь области Ω.

Данный подход к восстановлению изображений показал себя достаточно хорошо при визуальной оценке, но, как и говорилось ранее он обладает недостатком, данный подход размывает границы при переходах яркости у объектов на изображении (рис. 3, 4).

3. Методы на основе синтеза текстур и поиску по экземпляру

Одни из первых методов интерполяции использовали методы на основе синтеза текстур. Методы, основанные на текстуре, заполняют поврежденные или пропущенные области используя аналогичные окрестности в изображении, они пытаются сопоставлять статистику поврежденных регионов со статистикой известных областей в окрестности поврежденных пикселей.

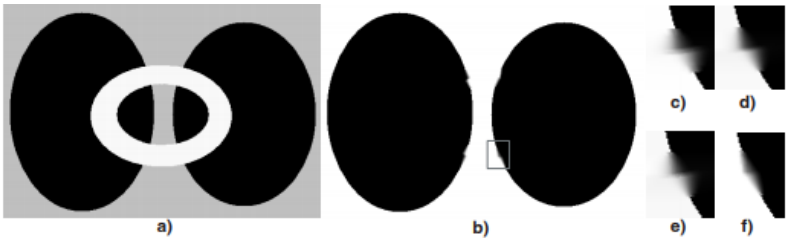
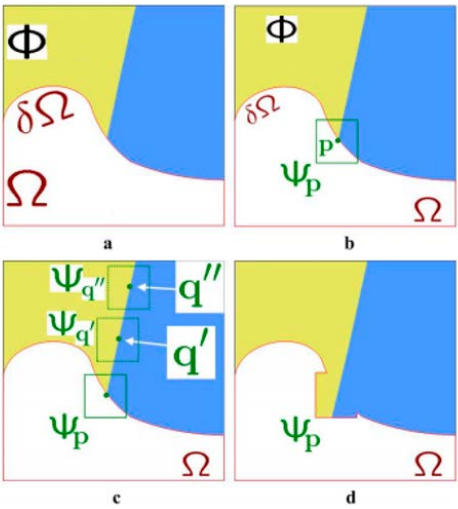


Рис. 3. Область для окраски (a) и результат (b), а изображения, полученные с различными значениями яркости объекта (c), (d), (e), (f).

Рис. 4. Пример потери четких границ

В зависимости от задачи, можно восстановить методом на основе синтеза текстур определенные изображения, с заданными параметрами. Как пример изображения с похожей текстурой, небольшими поврежденными участками пикселей или монотонные. Но недостаток этих методов в том, что они либо требуют точную ручную настройку параметров, либо нужны конкретные указания для восстановления области.

Одной из реализаций является метод Криминиси. Метод, разработанный Криминиси, сильно отличается от предыдущих тем, что он итеративно заполняет неизвестную область, используя, блоки известного изображения, осуществляя поиск наиболее похожих экземпляров из известной области и не использует диффузию и уравнения в частных производных. Отличительной чертой метода является способность восстанавливать значительные повреждения и переходы яркости на изображении (границы объектов) (рис. 5).



Ω - область, которую необходимо восстановить, Φ - известная область изображения, а ∂Ω - граница области Ω, Ψ с индексами блоки для восстановления

Рис. 5. Пример работы метода

На качество восстановления выходного изображения сильно влияет порядок, в котором выполняется процесс заполнения поражённой области. Порядок пикселей оценивается по приоритету блоков восстановления, приоритет вычисляется по формуле:

$$P(p) = C(p)D(p), \tag{3}$$

где C(p) - множитель уверенности, D(p) - множитель данных. Они вычисляются следующим образом:

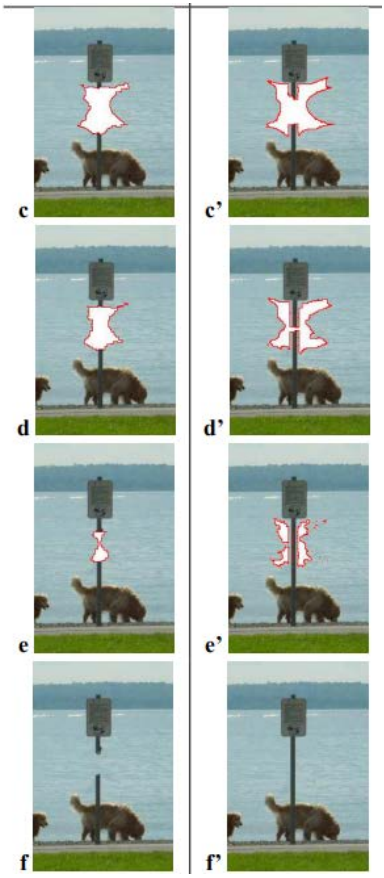
$C(p)$ принимает значения либо 0 если пиксель в области Ω, либо 1 если пиксель в области Φ.

$$D(p) = \frac{\left|\nabla I \perp n\right|}{a}, \tag{4}$$

где a это параметр нормализации.

Далее выбирается сегмент, наиболее подходящий по приоритету, и для него происходит поиск подходящего блока для восстановления.

Результаты работы метода мы можем видеть ниже, авторы сравнивают свой метод с методом, основанным на решении дифференциальных уравнений в частных производных. Как мы видим методы, основанные на PDE, плохо восстанавливают большой утерянный фрагмент, из-за потери чётких границ, а в следствии изображение получается смазанное, однако метод Криминиси визуально хорошо закрасил фрагмент из-за продления чётких границ исходного изображения (рис. 6).



Слева – базовый подход, справа – метод Криминиси

Рис. 6. Результат работы методов

Но также стоит упомянуть и недостатки метода, он достаточно долго выполняет работу, на компьютере с Intel core i7 8th Gen и оперативной памятью 16 гб, процесс заполнения пробела 100 на 50 пикселей и 50 на 50, на изображении с расширением 1000 на 667 пикселей занял более 6 часов, но главный недостаток данного метода проявляется в случаях, когда в процессе работы будет неправильно или недостаточно точно оценен наиболее подходящий фрагмент, тогда это может привести к лавинообразному накоплению систематической ошибки от итерации к итерации и, в конце концов, к абсолютно некорректным результатам работы (рис. 7).

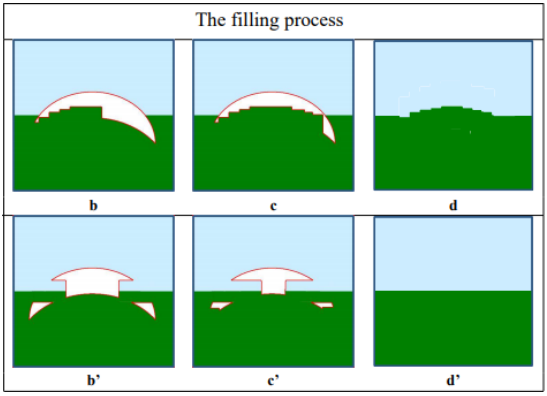


Рис. 7. Пример неверного заполнения областей

4. Гибридные методы восстановления

Гибридные методы воплощают в себе подходы на основе синтеза текстур и диффузных методов реконструкции. Основной идеей таких методов является разделение изображения на непосредственно текстурную и структурную части. Соответствующие части заполняются последовательно при помощи методов, основанных на использовании изофот и методов синтеза текстур. Как правило, такие методы обладают серьезной вычислительной сложностью за исключением необходимости реконструкции областей небольшого размера.

5. Разработка

В качестве практики был выбран метод Криминиси, так как он показал себя наиболее лучшим по сравнению с методами, использующими диффузию, средние значения для изображений размером 1000 на 667 пикселей (рис. 8).

Таблица 1

Сравнительные метрики алгоритмов

Метод	*Размер в пикселях*	*Криминиси*	*Телеа*
Пиковое отношение сигнала к шуму	1000 x 667	44.652440913	44.125503514
Индекс структурного сходства	1000 x 667	0.986897	0.926897

Рис. 8. Пример восстановления

В алгоритме предложенным Криминиси я доработал проверку блока, подходящего для встраивания в изображение, добавив дополнительную структурную проверку, проверка выполняется путем проверки значений пикселей блока и текстурных отличий. Также в ходе экскрементов было выявлено, что использование оператора Собеля для выделения границ пораженных областей и расчета множителя данных, лучше, чем использование оператора Лаплпса.

Данные преобразования позволили ускорить процесс восстановления с 5 часов до 3.5 часов, но большого прироста в качестве

восстановленного изображения я не получил, полученные метрики схожи с оригинальным методом.

Таблица 2

Метрики, полученные при разработке

Метод	*Размер в пикселях*	*Криминиси*	*Предложенная мной реализация*
Пиковое отношение сигнала к шуму	1000 x 667	44.652440913	44.706410311
Индекс структурного сходства	1000 x 667	0.986897	0.987914

Заключение

Данная статья посвящена обзору алгоритмов восстановления изображений, а также попытке реализации метода Криминиси. Работая над созданием статьи, я понял, что универсального метода для восстановления дефектов на изображении нет, у каждого есть свои преимущества и недостатки, и порой нужно человеческой вмешательство, чтобы исправить полученные артефакты после восстановления.

Список литературы

1. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М. : Техносфера, 2005. – 1072 с.

2. Хуанг, Т. С. Быстрые алгоритмы в цифровой обработке изображений: преобразования и медианные фильтры / Т. С. Хуанг. – М. : Радио и Связь, 1984. – 224 с.

3. Image inpainting / M. Bertalmio [et al.] // In Proc. ACM Conf. Comp. Graphics (SIGGRAPH). – New Orleans, LU, July 2000. – P. 417-424.

4. Criminisi, A. Region filling and object removal by exemplar-based inpainting / A. Criminisi // IEEE Transactions on Image Processing, 2004. – P. 13.

5. Telea, A. An Image Inpainting Technique Based on the Fast Marching Method/ A. Telea // Journal of Graphics Tools, 2004. – P. 9.

Использование технологий генерации кода при решении задачи автоматического логирования

А. В. Тарасов

Студент бакалавриата

Н. К. Самойлов

Старший преподаватель

Введение

Анализ работы программы – неотъемлемая часть процесса разработки. На любом этапе разработки и поддержки продукта требуется четкое понимание работы как всей программы, так и каждого ее компонента. Для этого используются логи, которые записываются в файле на сервере с программой, или отправляются на логирующий сервер, или отправляются в порт логирующей программы. Эти операции требуют большого количества времени и довольно затратны для программы. Поэтому логирование добавляется только в ключевые места программы, записи из которых могут указать на причину возникшей ошибки. Однако, при возникновении ошибки зачастую может быть недостаточно таких логов. Для локализации ошибки потребуется логировать код более углубленно. К тому же во многих проектах на ранних стадиях разработки не было добавлено логирование. Таким образом после увеличения проекта и кодовой базы большая часть кода не использует логирование. Это серьезно усложняет поиск ошибок, которые возникают при дальнейшей разработке и поддержании проекта.

Для решения этой задачи существует множество инструментов, облегчающих логирование кода. Специальные инструменты, предназначенные для этого, не позволяют покрыть логирующим кодом весь проект. Они позволяют сократить описание логирования на языке java например с помощью java-аннотаций.

С другой стороны, существуют инструменты, позволяющие дополнить код без его изменения по всему проекту. Но эти инструменты достаточно сложны, как с технической точки зрения, так и с точки зрения использования их. Библиотека основанная на генерации легирующего кода может закрывает поставленные проблемы. Также есть возможность создать структуру логов, которая ускорит анализ

работы программы. С помощью таких структур можно вывести анализ кода на более высокий уровень абстракции. Также, существует большое количество программ для анализа логов (такие как ELK Stack) которые активно развиваются и могут требовать изменения формата или способа логирования, для ускорения работы программы или добавления новых возможностей анализа данных.

1. Принцип работы модуля

Задача модуля состоит в логировании определенных функций в коде. Определение этих функций происходит по определенным признакам, таким как модификаторы доступа, принадлежность определенным классам, входным параметрам и результатам ее работы.

Исходя из задачи понятно, что для получения всей нужной информации о коде лучше всего подходит обработка кода на этапе компиляции. При такой обработке появляется доступ к абстрактному синтаксическому дереву. На этапе компиляции так же доступны встроенные средства генерации кода для языка Java.

Исходя из задач и доступных инструментов, работа модуля сводится к следующему набору действий:

1. Используя рефлексию[1], собирается информация обо всех классах и функциях внутри этих классов, которые должны быть залогированы.
2. С помощью средств генерации кода создаются классы-наследники от классов, которые были найдены на первом этапе. Эти классы содержат логирующий код и являются надстройкой над существующими классами, содержащими бизнес-логику.
3. На последнем шаге нужно заменить использование логируемого класса, использованием классом наследником.

Также происходит встраивание созданного класса в структуру наследования. Таким образом будет обеспечено правильное приведение типов.

2. Инициализация модуля

При инициализации модуля создается синглетон[2] класса, содержащий все описанные ниже данные.

Для фильтрации и модификации кода используются как фильтры и логирование по умолчанию, так и специфические, указанные программистом, использующим этот модуль.

По умолчанию присутствует несколько фильтров. Например:

– фильтр по модификаторам доступа, как класса, в котором находится функция, так и самой функции;
– по статичности функции;

– по аннотации, а именно указывается полное имя аннотации и логируются все классы и функции аннотированные этой аннотацией.

По умолчанию, присутствуют следующие способы логирования:

– отправка логов в стандартный поток вывода;
– отправка логов в порт;
– используя библиотеку log4j;
– использование фасада Slf4J;
– использование стандартного логирования спринг.

Поскольку стандартных способов логирования может не хватить, в связи с особенностями архитектуры или используемых компонентов, программисту дается возможность самостоятельно определить фильтры и логирующие функции. Реализуется это также с помощью рефлексии. Производится поиск всех функций со специализированной аннотацией и в дальнейшем они используются в качестве фильтров и логирующих модификаторов.

В процессе инициализации модуля происходит сбор функций по умолчанию и прописанных пользователем и они добавляются в определенные словари. Также, читаются настройки из окружения. В них должны быть прописаны правила объединения фильтров функций (and/or), перечисление фильтров, которые нужно использовать.

Последним шагом инициализации является сбор всех имен классов, которые должны будут логироваться. Создается список по которому в дальнейшем будет определяться, нужно ли заменять в абстрактном синтаксическом дереве класс на наследника.

3. Фильтрация логируемых функций

Доступ к элементам дерева осуществляется при помощи процессора аннотаций. Он реализуется с помощью наследования от абстрактного класса AbstractProcessor.class. Посещаемые им классы могут определяться какой-либо аннотацией. Для этого она ставится на классы в коде, и так же указывается в настройках к процессору аннотаций. Также есть возможность указать для посещения все классы, в том числе и не аннотированные никакими аннотациями.

Класс в данном случае представлен узлом абстрактного синтаксического дерева. Каждый элемент дерева содержит информацию о данном узле, список потомков и ссылку на родителя.

Для фильтрации используется классы-спецификации, написанные для фильтрации по умолчанию и написанные программистом-пользователем. Фильтры делятся на фильтры для классов и фильтры для функций. Фильтры для классов являются наследниками от абстрактного класса Specification как показано на рис. 1. Каждый класс, написанный программистом-пользователем, должен быть аннотирован

@SpecificationType. По этой аннотации с помощью рефлексии происходит поиск спецификаций.

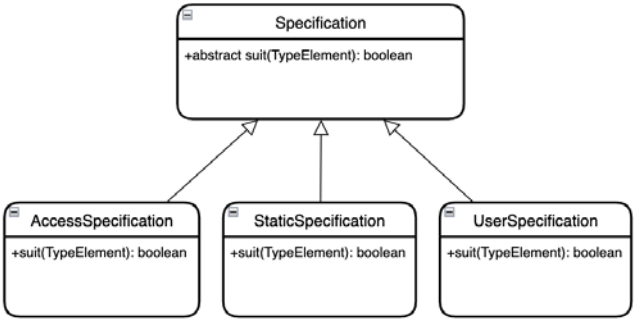


Рис. 1. Диаграмма классов спецификаций для классов

По аналогии с фильтрами классов, фильтры для функций являются наследниками от абстрактного класса SpecificationFunk как показано на рис. 2.

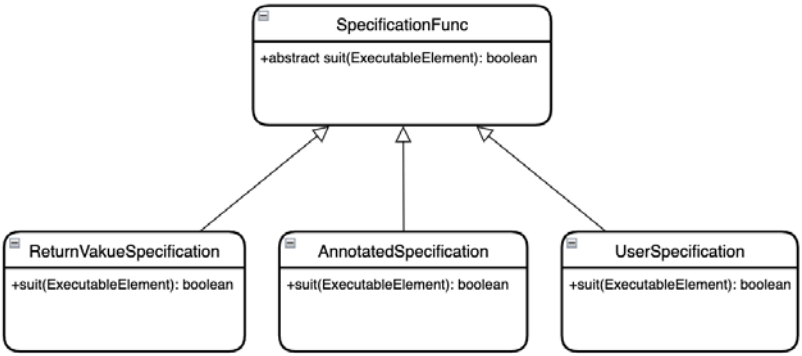


Рис. 2. Диаграмма классов спецификаций для функций

В первую очередь происходит фильтрация классов полученных из аст дерева. После начинается фильтрация классов и функций находящихся внутри класса из аст дерева. По подходящим классам и функциям формируется шаблон создаваемого класса. Он содержит в себе элемент аст дерева логируемого класса и список его функций, которые должны быть залогированы.

4. Определение логирующих функций

По аналогии с поиском спецификаций для элементов абстрактного синтаксического дерева происходит и сбор функций, определяющих способ логирования.

Классы описывающие способ логирования наследуются от абстрактного класса Logging и должны быть помечены аннотацией @LoggingType. Диаграмма классов логирования, определенных по умолчанию и написанных программистом-пользователем представлена на рис. 3.

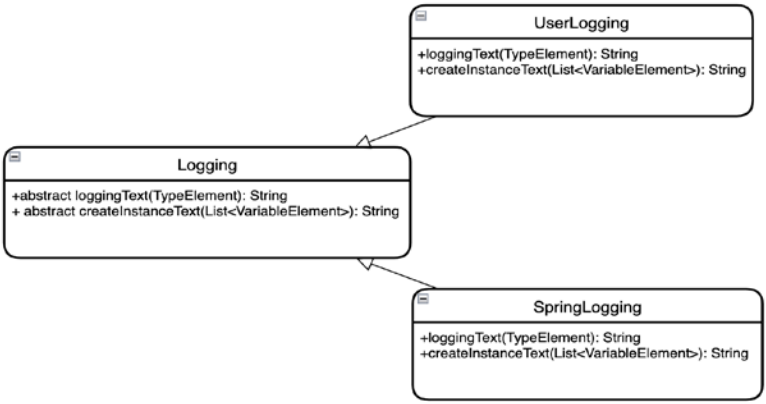


Рис. 3. Диаграмма классов генераторов логирующего кода

В наследниках реализуются две абстрактные функции суперкласса Logging:

– первая возвращает строку кода, отвечающую за генерацию сущности легирующего класса в качестве поля по данным о легируемом классе. В качестве входных параметров в нее передается класс-элемент AST дерева - полученный при выборке легируемых классов;

– вторая генерирует код, который отвечает за логирование. В качестве параметров эта функция получает список элементов AST-дерева которые описывают переменные передаваемые в бизнес-функцию или же возвращаемое значение.

5. Генерация кода

Последним этапом работы модуля - создание класса, наследующегося от логируемого класса, в котором переопределены функции, которые должны быть залогированы.

Генерация происходит на основе шаблона легируемой функции. Название класса генерируется как имя класса с добавлением определенного в настройках постфикса. Если он не определен, используется постфикс по умолчанию - Autogenerated. В качестве пакета указывается тот-же, что и у класса, на основе которого генерируется класс.

В качестве полей добавляются логирующие классы, в том числе добавляется способ их получения. За генерацию этого кода отвечает функция createInstanceText в классах-наследниках от Logging.

Для каждой функции из шаблона генерируется переопределяющая ее функция. Ее тело состоит из логирующего кода и вызова переопределенного метода. Условно, этот код можно разделить на три блока:

– первый блок - код логирующий входные параметры функции. За генерацию этого кода отвечает функция loggingText наследников Logging.class;

– второй блок - вызов функции суперкласса. Если возвращаемое значение не void, то результат работы этой функции записывается в переменную;

– если возвращаемое не void также каждым логером логируется результат работы функции и это значение возвращается в качестве результата.

В результате будут созданы соответствующие классы в папке generated. Пример класса приведен в листинге кода. Так же происходит модификация кода в аст дереве. Модификация аст дерева происходит с помощью «посетителей» [3]. На рис. 4 показаны деревья до и после модификации, при логировании функции test.

Листинг

```
public class TestAutologAutogenerated extends TestAutolog {
  org.slf4j.Logger logger =
LoggerFactory.getLogger(TestAutolog.class);
  public Integer test(String arg) {
    logger.info(arg);
    Integer temp = super.test(arg);
    logger.info(temp.toString());
    return temp;
  }
}
```

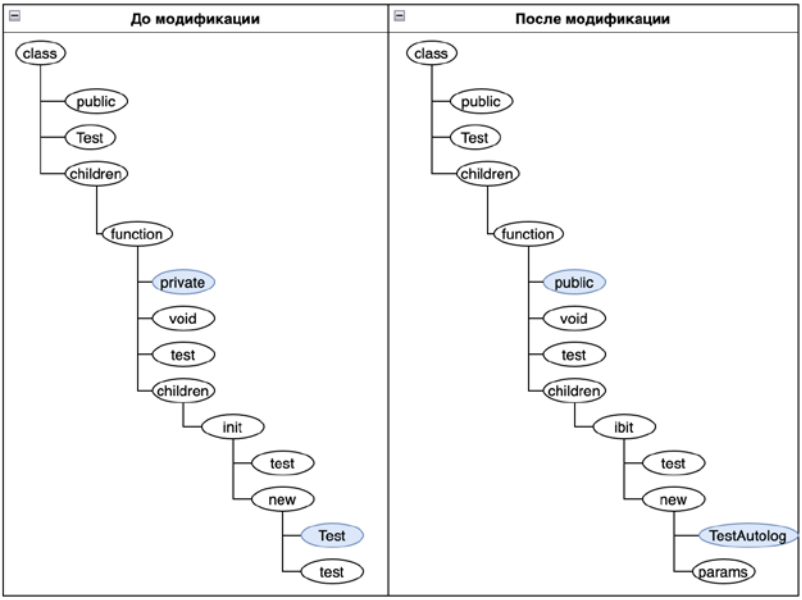


Рис. 4. Обычное и модифицированное ast дерево

Заключение

Данная статья посвящена разработке и реализации программного модуля автоматического логирования кода. Были изучены теория компиляции джава-кода, принцип работы компилятора и библиотеки, позволяющие встраиваться в процесс компиляции. Так же были изучена встроенная в java функциональность отвечающая за генерацию кода. Реализован и описан программный модуль, позволяющий с помощью определенных настроек логировать весь код проекта. В дальнейшем предполагаются усовершенствования настроек фильтров и способов логирования, также усовершенствования способов комбинирования фильтров.

Список литературы

1. Блох, Дж. Effective Java / Джошуа Блох. – 1-е изд. – М.: Просвящение, 2009. – 233 с.

2. Брюс, Э. Философия Java / Э. Брюс. – 4-е изд. – СПб.: Питер, 2015. – 362 с.

3. Соломатин, Д. И. Основы синтаксического разбора, построение синтаксических анализаторов / Д. И. Соломатин, А. В. Копытин, А. И. Другалев. – Воронеж: ВГУ, 2014. – 69 с.

Особенности реализации in-memory хранилища на основе B/B+-деревьев

Д. В. Титов

Студент бакалавриата

Н. К. Самойлов

Старший преподаватель

Введение

В сфере вычислительной техники кэш – это высокоскоростной уровень хранения, на котором требуемый набор данных, обычно, временного характера. Доступ к данным, хранящимся на этом уровне, осуществляется намного быстрее, чем к основному месту их хранения. Кэширование делает возможным эффективное повторное использование данных, вычисленных ранее.

Само по себе кэширование – это процесс, при котором копии данных или файлов хранятся во временном хранилище – или в кэше – чтобы к ним можно было получить доступ значительно быстрее. Кэш временно сохраняет данные для приложений, серверов и веб-браузеров, благодаря чему пользователям не нужно загружать информацию каждый раз, когда они обращаются к веб-сайту или приложению.

Кэшированные данные обычно включают мультимедийные материалы, такие как изображения, файлы и сценарии, которые сохраняются на устройстве, когда пользователь впервые открывает приложение или посещает веб-сайт. Это используется для быстрой загрузки информации о приложении или веб-сайте каждый раз, когда пользователь впоследствии открывает или посещает его.

Кэширование является решением проблемы так называемого «узкого места архитектуры Фон Неймана», которое проявляется в ограничении пропускной способности между ЦП и памятью по сравнению с объемом памяти. Из-за невозможности одновременного доступа к памяти программ и памяти данных, пропускная способность канала «процессор-память» и, соответственно, скорость работы самой памяти в значительной степени ограничивают скорость работы ЦП [1].

В данной статье будут рассмотрены особенности реализации in-memory кэш-хранилища на основе B, B+ деревьев, так как для этих

целей они являются наиболее подходящим выбором за счет следующих характеристик:

– Регулярное добавление и удаление элементов из коллекции эффективнее реализованы в деревьях, так как перемещение всех элементов в больший контейнер на хэш-карте занимает O(n) времени. Удаление элементов из хэш-карты не освобождает столько памяти, сколько их удаление дерева.

– При работе с большими коллекциями деревья обеспечивают более быстрое время поиска, нежели хэш-карта.

1. Хранение объектов в B-дереве

B-деревом называется дерево, удовлетворяющее следующим свойствам:

1. Ключи в каждом узле обычно упорядочены для быстрого доступа к ним. Корень содержит от 1 до 2t-1 ключей. Любой другой узел содержит от t-1 до 2t-1 ключей. Листья не являются исключением из этого правила. Здесь t — параметр дерева, не меньший 2 (и обычно принимающий значения от 50 до 2000).

2. У листьев потомков нет. Любой другой узел, содержащий ключи $K_1,\ldots,K_n$, содержит n + 1 потомков. При этом:

2.1. Первый потомок и все его потомки содержат ключи из интервала $(-\infty, K_1)$

2.2. Для $2 \le i \le n$, i-й потомок и все его потомки содержат ключи из интервала (K_{i-1}, K_i)

2.3. (n+1)-й потомок и все его потомки содержат ключи из интервала (K_n, ∞)

3. Глубина всех листьев одинакова [2].

Пример B-дерева приведен на рис. 1.

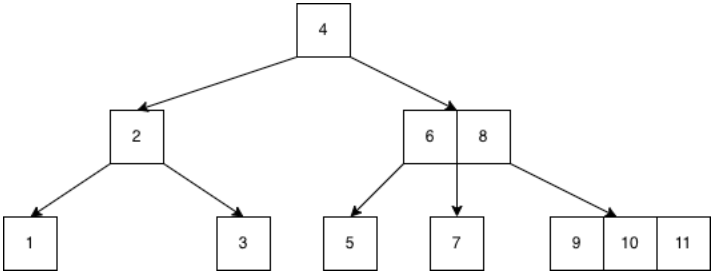


Рис. 1. Пример B-дерева

Для того, чтобы кэш-хранилище было пригодным для реального использования, необходимо добиться того, чтобы было возможным

хранить в нем объекты, состоящие из набора скалярных типов. Отсюда следует, что фильтрация объектов в такого рода хранилище усложнится, так как условие фильтрации может быть как простым, то есть состоять из одного логического выражения, так и сложным, то есть состоять из совокупности логических выражений (суперпозиции логических функций), включающих себя сравнение нескольких свойств объекта одновременно.

В случае, если подобное требование не будет реализовано, применение хранилища в реальной предметной области будет значительным образом ограничено, так как выполнение операций для произвольных объектов в общем виде окажется фактически невозможным.

В связи с этим, в процессе разработки приложения кеширования было необходимо решить задачу, сформулированную выше.

Ситуация осложняется тем, что фундаментальным принципом построение B-дерева служит наличие в каждой вершине (узле) дерева множества ключей (меток), позволяющих ссылаться на дочерние элементы узла. Каждому объекту, содержащемуся в B-дереве сопоставляется единственный ключ (метка). Таким образом, если хранящийся в B-дереве объект не является скалярным значением, а состоит из нескольких свойств, то ключ, сопоставляемый этому объекту, не сможет в полной мере обеспечить полноценный доступ к совокупности объектов с определенным значением одного или нескольких свойств. Поскольку операция обхода дерева основана на свойстве упорядоченности набора ключей, содержащегося в каждом узле (вершине) дерева, нетрудно догадаться, что осуществить эффективное извлечение данных из B-дерева можно лишь по одному, заданному в момент создания дерева ключу.

В качестве решения этой задачи был выбран метод, широко использующийся при проектировании и реализации СУБД (Систем Управления Базами Данных), вполне зарекомендовавший себя и прошедший проверку временем. Задача поиска кортежей в отношениях, атрибуты которых соответствуют заданным Оператором СУБД условиям, похожа на решаемую в нашем случае. Решение же состоит в построении дерева индекса отношения для наиболее часто применяющихся в логических функциях, используемых в условиях поиска, атрибутов отношения. Иными словами, для всякой совокупности атрибутов, часто использующихся в SQL операторах извлечения данных типа SELECT … WHERE, генерируются собственное B-дерево, ключом которого является объект, описывающий такую совокупность атрибутов.

Похожая концепция используется и при решении нашей задачи. Для каждого свойства хранящегося в кэше объекта, строится собственное B-дерево, ключом которого является данное свойство. Дополнительно, строятся деревья с составными ключами для всех сочетаний свойств объекта по 2. Таким образом, конечное количество B-деревьев будет равняться:

$$B-tree\ \text{count} = \mathrm{N} + \mathrm{C}_N^2 = N + \frac{N!}{(N-2)!\times 2!} = \frac{N(N+1)}{2}, \qquad (1)$$

где N – количество свойств объекта.

Естественно, в качестве значения, содержащегося в B-дереве будет применяться не сам объект, а только ссылка на объект, хранящегося в куче (heap), применяющегося в Java Runtime для хранения экземпляров классов. Отсюда следует очевидный минус такого подхода, заключающийся в дополнительном использовании памяти. Одна ссылка на объект в Java Runtime занимает порядка 8 байтов (для 64-битной платформы), таким образом на хранение одного объекта помимо данных самого объекта будет затрачено в качестве накладных расходов на хранение соответственно:

$$Memory = 4N \times (N+1) \text{ байт}, \qquad (2)$$

где N - количество свойств объекта.

Однако, подобный недостаток с лихвой компенсируется высокой скоростью доступа к данным, содержащимся в кэш-хранилище.

В случае, если извлечение данных осуществляется по совокупности значений свойств, не представленных в виде подготовленных B-деревьев, производится разбиение этой совокупности на группы выражений, для которых существует построенное B-дерево с соответствующим простым или составным ключом, и выполняется извлечение для каждой такой группы в отдельности. Для полученных множеств объектов определяется пересечение этих множеств, которое и будет содержать искомые объекты, удовлетворяющие требуемым условиям.

Подобная операция, конечно, будет занимать больше времени, чем извлечение объектов в зависимости от совокупности значений свойств, для который уже существует готовое B-дерево с соответствующим ключом.

Однако, статистически количество таких операций оценивается меньше, чем количество операций, где проводится извлечение объектов, удовлетворяющих одному-двух простым условиям, то есть таким условиям, которые проверяют равенство единственного свойства объекта заданному значения, и для которых существуют предварительно

посчитанные B-деревья с соответствующими этим свойствам ключами. В худшем случае время выполнения операции извлечения будет лишь немного выше, что не повлияет значительным образом на производительность хранилища.[3]

2. Хранение данных в B+-дереве

Наиболее важным вариантом B-дерева, широко используемым для хранения данных на диске, является B+-дерево.

1. Ключи в каждом узле обычно упорядочены для быстрого доступа к ним. Корень содержит от 1 до 2t-1 ключей. Любой другой узел содержит от t-1 до 2t-1 ключей. Листья не являются исключением из этого правила. Здесь t — параметр дерева, не меньший 2 (и обычно принимающий значения от 50 до 2000).
2. У листьев потомков нет. Любой другой узел, содержащий ключи $K_1,\ldots,K_n$, содержит n+1 потомков. При этом:
   2.1. Первый потомок и все его потомки содержат ключи из интервала $(-\infty,K_1)$
   2.2. Для $2 \le i \le n$, i-й потомок и все его потомки содержат ключи из интервала (K_{i-1}, K_i)
   2.3. (n+1)-й потомок и все его потомки содержат ключи из интервала (K_n, ∞)
3. Глубина всех листьев одинакова.
4. Листья имеют ссылку на соседа, позволяющую быстро обходить дерево в порядке возрастания ключей, и ссылки на данные.

Рассмотрим подробнее принцип хранения данных.

Итак, предположим, что в хранилище необходимо сохранить следующие объекты типа «Автомобиль»:

- {Model: “BMW”, Price: 1000};
- {Model: “Subaru, Price: 2000};
- {Model: “Bentley”, Price: 10000};
- {Model: “Fiat”, Price: 500};
- {Model: “Nissan”, Price: 4500};
- {Model: “Skoda”, Price: 7000}.

Во-первых, хранилище создаст уникальный случайный индекс для каждого объекта и преобразует каждый набор свойств в поток битов. В данном случае присвоенный каждому объекту индекс будет являться ключом индексации данной записи в дереве. Далее происходит сохранение всех сгенерированных ключей и запись полученных байтовых последовательностей в B+-дерево.

Таким образом, полученное в результате описанных выше действий B+-дерево можно представить следующим образом (рис. 2).

На рис. 2 можно увидеть, что все записи хранятся в листьях. Ключ – сгенерированный случайным образом индекс, использующийся при создании дерева. На внутренних узлах какие-либо записи отсутствуют. Листья содержат ссылку на следующую запись объекта, что позволяет производить в хранилище как бинарный, так и последовательный поиск, совершая проходы только по листьям.

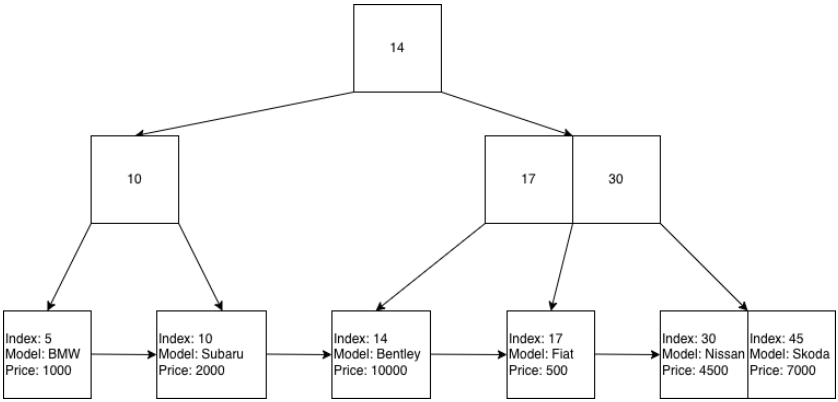


Рис. 2. B+-дерево в хранилище

В случае, когда индексирование не используется, хранилище производит поиск по листовым узлам. Иначе, оно создает три B-дерева для каждого из свойства хранимого объекта следующим образом (рис. 3). Ключ — это ключ B-дерева, используемый для индексации. Как упоминалось в предыдущем разделе, мы храним лишь ссылки на объекты, а потому в качестве этой ссылки будет выступать индекс, который будет указывать на фактическое расположение объекта.

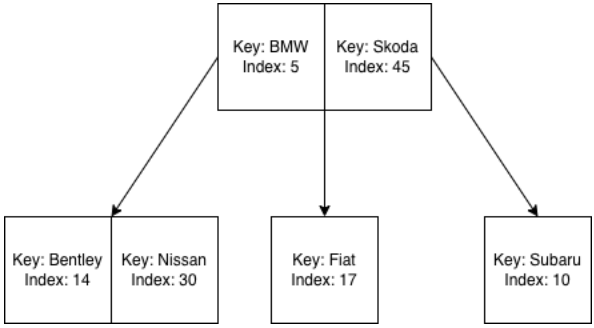


Рис. 3. Индексация внутри хранилища

При индексации на первом этапе заданный ключ в хранилище будет найден за время O (log (n)). На втором этапе выполнится еще один поиск в B+дереве, но уже фактической записи, используя полученный на первом этапе индекс, и пользователь получит запрашиваемый объект.

Заключение

Таким образом, были проанализированы B/B+-деревья, их устройство в памяти, а также пригодность их использования в качестве структуры данных для создания in-memory хранилища временных данных. Предложенный вариант хранилища, а также описанные особенности его реализации на основе рассмотренных деревьев поиска, механизмы хранения и выборки данных, позволяют хранить в нем данные произвольной структуры, а описанный механизм индексации позволяет хранилищу быстрее производить операцию чтения.

Список литературы

1. Smith, A. Cache memories / A. Smith // ACM Computing Surveys. – 1982. – Vol. 14. – Iss 3. – 473-530 p.

2. Graefe, G. Modern B-tree techniques / G. Graefe, H. Kuno // 27th International Conference on Data Engineering. – Hannover, Germany, 2011. – Vol. 3. – Iss. 4. – P. 203-402.

3. Comer, D. Ubiquitous B-Tree / D. Comer // ACM Computing Surveys. – 1979 – Vol. 11. – Iss. 2 – P. 121–137.

«Игровые механики» в неигровых приложениях

Д. А. Тишанский

Студент бакалавриата

Д. И. Соломатин

Старший преподаватель

Введение

Современный рынок мобильных приложений развивается со стремительной скоростью. Все чаще появляются приложения, помогающие людям упростить различные процессы в жизни, начиная от менеджмента задач и заканчивая онлайн-обучением. Ввиду стремительного роста конкуренции на этом рынке и интенсивной интеграции приложений в человеческий быт возникает проблема поддержания постоянной вовлеченности пользователя в сервис.

Одним из решений этой проблемы является применение опыта игрового дизайна в разработке приложений, которые явно не являются играми, в частности это интеграция различных игровых механик.

1. Что такое игра?

Стоит начать с самых истоков и разобраться с тем, что такое вообще игра. Обычно большая часть исследователей игр выделяют три ключевых признака, отличающих игру от любого иного вида человеческой деятельности.

1. Обособленность законов игры или принцип “магического круга”. “Магический круг” – пространство, в котором реальные законы и правила (физические, логические, этикет и т.п.) на время перестают работать и заменяются законами игрового мира. К примеру, игра в монополию несет за собой ценность только в пределах игрового поля, отчерчивая тем самым процесс игры от остальных процессов реального мира.

2. Добровольность. Выполнение или невыполнение действий по собственной воле, без принуждения и осознанно. Игра уже не может считаться игрой, если игрока принуждают в нее играть.

3. Цель. Отличие игры от других видов деятельности – это наличие цели, конечный результат, состояние победы в игре.

Все вышеперечисленные аспекты наряду с игровыми механиками, которые будут рассмотрены далее, создают ощущение “фана” (англ. fun – жаргон, использующийся в игровом сообществе, обозначающий ощущение радости, веселья, легкости, непринужденности, игривости) от процесса игры.

2. Что такое игровые механики?

Игровая механика – это правила и действия, которыми руководствуется игрок, а также обратная реакция игры на действия игрока. С помощью игровых механик определяется, как игра будет взаимодействовать с людьми, которые в нее играют. Если выражаться просто, то механика описывает правила, которым следует игрок, и правила, которым следует сама игра.

3. Понятие игрофикации

Игрофикация – это использование игровых механик в неигровом контексте [1].

Стоит разграничить само понятие игрофикации от игры, т.к. между этими двумя понятиями существуют достаточно важные различия в тех аспектах, которые отличают игру от иных видов деятельности.

Смысл игры, как уже выяснилось, существует только в пределах самой игры, и тот же бросок кубика или деньги в монополии ничего не значат в реальном мире. В игрофикации этот принцип обозначен несколько иначе: к деятельности человека в реальном мире накладывается некий игровой смысл. Тут можно привести забавный пример: при прогулке по площади, которая покрыта плиткой, многие стараются не наступать на места пересечения плиток, представляя, что они шагают по платформам над придуманной пропастью. Таким образом обычная прогулка путем наложения на нее игрового смысла превращается в нечто веселое и забавное.

Цель игры также имеет значение только в пределах самого игрового пространства, т.е. если вы выиграли у другого игрока в монополию, то эта победа актуальна только в контексте самой игры. В игрофикации же цель существует за пределами игрофицированного пространства. Вспоминая пример с пересечением плиток, здесь цель – дойти до конечной точки маршрута, и она может быть выполнена вне зависимости от игрофикации процесса.

4. Принципы игрофикации

Игрофикация оперирует тремя основными принципами [2].

1. Мотивация. Это основа, на которой зиждется любой игровой принцип. Перед людьми, играющими в игры, должна быть конкретная

цель, к которой они стремятся. Игрофикация здесь, наследуя те же подходы, что и в играх, выступает дополнительным стимулятором мотивации к процессу работы, обучения или иной деятельности.

2. Статус. В любой игре тем или иным образом игрок развивает своего игрового персонажа, чем больше испытаний он проходит, тем сильнее становится его персонаж. Игрофикация же из-за более тесной связи с реальностью способствует развитию самого игрока и предлагает явным образом визуализировать его прогресс на фоне себя самого и остальных участников.

3. Вознаграждение. Этот принцип особенно усиливает вовлеченность участников в процесс, но также крайне важна сама награда. Например, стандартная система оценивания учеников не предлагает ничего выше максимальной оценки, тем самым у ученика может пропасть мотивация стараться лучше, достигая новые вершины. Дополнительное вознаграждение же будет мотивировать его действовать быстрее, лучше и дает дополнительный интерес.

Таким образом, игрофикация лишь выступает неким катализатором к повседневной деятельности человека, которая помогает быстрее двигаться к цели за счет игровой вовлеченности.

5. Игрофикация в приложениях

В настоящий момент различные мобильные приложения и сервисы активно пытаются использовать игрофикацию для улучшения одного из ключевых показателей – вовлечённости в использование продукта. Далее будут рассмотрены несколько примеров игрофикации в мобильных приложениях.

1. Todoist Karma. На первый взгляд обычное приложение по менеджменту задач, но с игрофикацией оно становится эффективным инструментом повышения производительности пользователя.

Его балльная система ранжирует и мотивирует игрока к достижению целей. Когда пользователь добавляет и выполняет задачи, он получает очки “кармы”, повышая с помощью них свой уровень. Дополнительную награду игрок получает, если он усложняет себе задачи, создавая из них серию или ставя сроки выполнения.

Таким образом, игрок принимает вызов и после его выполнения получает вознаграждение и может отслеживать свой прогресс в достижении целей (рис. 1).

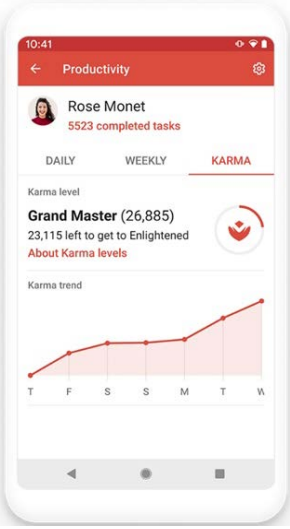


Рис. 1. Шкала прогресса выполнения задач

2. Duolingo. Образовательное приложение для изучения иностранных языков и один из показательных примеров успешного применения игрофикации [3].

Изучение языка построено по вполне стандартной системе уроков, при этом в каждом уроке присутствует элемент игры: пользователи зарабатывают очки за правильные ответы и прохождение курсов. Количество очков, заработанное в определенный день, отображается в виде графика, тем самым визуализируя прогресс пользователя (рис. 2).

Рис. 2. Прогресс пользователя в прохождении уроков

Помимо очков пользователи в качестве награды могут получить достижения (рис. 3), выполнив определенные условия. Такое дополнительное вознаграждение побуждает пользователя выполнять больше заданий и повышать свой уровень владения языком.

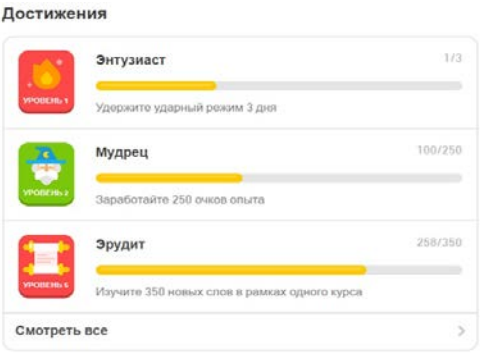


Рис. 3. Достижения пользователя

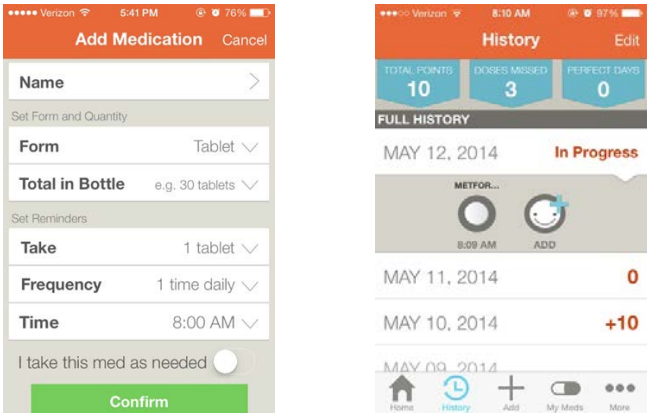
Именно такой подход к изучению языков с применением игрофикации позволяет Duolingo показывать отличные результаты – большинство пользователей приложения используют его на регулярной основе и показывают эффективные результаты во владении языком.

3. Mango Health. Приложение, которое поможет вести учет приема лекарств с помощью игровых механик.

Система уведомлений приложения позволяет не только не забывать о приеме лекарств в определенное время, но и поощряет это, награждая пользователя очками (рис. 4). Эти очки можно обменять на подарочный сертификат, который можно использовать для приобретения реальных товаров у партнеров из сфер страхования и медицины.

4. Codeforces. Сервис, на котором проводятся соревнования по программированию. С точки зрения игрофикации данный веб-сайт также имплементирует некоторые игровые механики. Например, за каждый успешно пройденный “раунд” участнику присуждаются очки, которые суммируются в его рейтинге. В зависимости от рейтинга участнику присуждается определенные звания и открывается возможность участвовать в первом дивизионе, в котором задачи уже несколько сложнее и награда за их решение будет выше.

Участник также в случае неправильного решения задачи или переотправке решения будет получать штрафные балл. При этом имеется возможность “взломать” решения других игроков, получив при этом награду в виде баллов.



а б

а – составление расписания приема лекарств,
б – расписание и прогресс в приеме лекарств

Рис. 4. Интерфейс приложения

По завершению раунда происходит вычисление окончательного результата, учитывающего сложность задач, количество попыток сдачи и результаты “взломов” решений других игроков. На основании итогового балла определяется место участника, от которого уже зависит конечный прирост в общем рейтинге.

Такой комплексный подход с применением игровых механик, таких как система рейтинга и вознаграждений, является хорошим стимулом для участников совершенствовать свои навыки в программировании.

Заключение

Исходя из опыта применения игрофикации в приложениях и принципах, на которых она реализуется, можно сделать вывод, что конечная цель применения игровых механик в неигровых приложениях – обеспечить максимальную вовлеченность пользователя в сервис.

Игрофикация отлично работает при наличии социальных элементов взаимодействия, позволяя пользователям делиться результатами своих достижений, тем самым создавая своего рода соревновательный аспект.

Стоит также отметить, что игрофикация не является созидательным инструментом. Она, по сути, должна лишь улучшать уже работающую функциональность. В противном случае игрофикация не будет влиять на конечного пользователя должным образом.

Список литературы

1. What is Gamification? [Электронный ресурс]. – Режим доступа: https://www.gamify.com/what-is-gamification

2. Zichermann, G. Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps / G. Zichermann, C. Cunningham. – 1st ed. – O'Reilly Media, Inc., 2011. – 182 p.

3. Game Mechanics for Non-Gaming App Businesses [Электронный ресурс]. – Режим доступа : https://sensortower.com/blog/game-mechanics-for-non-game-apps-infographic

Методы выделения именованных сущностей и отношений из предложений на естественном языке

П. А. Хоменко

Студент бакалавриата

В. В. Гаршина

Доцент

Введение

Распространение цифровых вычислительных устройств и их использование человеком в целях коммуникации привели к увеличению спроса на системы и алгоритмы, способные добывать информацию из неструктурированных текстовых данных.

Для того чтобы получить значимые данные из текста, нам нужно следовать методу, который называется Интеллектуальный анализ текста или Text Mining.

Технология Text Mining представляет собой одну из разновидностей методов Data Mining и подразумевает процессы извлечения знаний и высококачественной информации из текстовых массивов. Это обычно происходит посредством выявления шаблонов и тенденций с помощью средств статистического изучения шаблонов.

Такая технология глубинного анализа текстов способна «просеивать» большие объемы неструктурированной информации и выявлять из них только самое значимое, чтобы человеку не приходилось самому тратить время на добычу ценных знаний «вручную».

Ознакомление с технологией Text Mining необходимо начинать с изучения задачи извлечения именованных сущностей (Named-entity recognition, NER) и смысловых связей (Semantic Relations recognition).

Данная работа посвящена проведению сравнительного анализа существующих методов извлечения фактов из текстовых данных и извлечения именованных сущностей на языке программирования Python с помощью библиотеки NLTK.

1. Понятие именованной сущности

Именованные сущности – это объекты определенного типа, чаще всего составные, например, названия организаций, имена людей, даты, места, денежные единицы и. т.д.

В зависимости от прикладных задач, может быть необходимо выделить в тексте, во-первых, имена собственные: имена лиц, топонимы, названия организаций, названия песен и исполнителей, названия товаров и брэндов; [6. 1] во-вторых, такие объекты как числа, даты, денежные единицы.

Наибольшее распространение для широкого спектра задач получила выделение таких сущностей, как:

– Персона (имена, фамилии, отчества людей).
– Локация (топонимы).
– Организация (названия организаций, компаний, объединений).
– Разное (в эту группу входят все прочие типы сущностей, если их более тщательное разделение не требуется для целей исследования).

Таким образом, за одной задачей распознавание именованных сущностей (NER), на самом деле, стоит две: обнаружить, что какая-то последовательность слов – это именованная сущность и понять, к какому классу (имя человека, название организации, город и т. п.) эта именованная сущность относится.

2. Проект iPavlov

DeepPavlov – это фреймворк с открытым исходным кодом для создания диалоговых помощников и анализа текста.

DeepPavlov – это:

– набор предварительно обученных нейросетевых моделей, предварительно определенных компонентов диалоговой системы (ML/DL/Rule-based) и шаблонов конвейера;
– фреймворк для реализации и тестирования диалоговых моделей;
– набор инструментов для интеграции приложений со смежной инфраструктурой (мессенджеры, программное обеспечение helpdesk и т. д.).

Естественный язык использует методы машинного обучения, чтобы понимать структуру и терминологию в тексте. Вы можете извлекать информацию о людях, местах и событиях, а также лучше понимать позитивные/негативные мнения в социальных сетях и интерпретировать разговоры с клиентами. Методы обработки естественного языка позволяют анализировать текст и получать ответы в контексте.

Предлагается к использованию предварительно обученная модель на основе BERT, которая может использоваться для вывода как из интерфейса командной строки (CLI), так и из Python. Перед использованием модели нужно убедиться, что все необходимые пакеты установлены с помощью команды:

```
python -m deeppavlov install ner_ontonotes_bert
```

Чтобы воспользоваться моделью из интерфейса командной строки, используйте следующую команду:

```
python deeppavlov/deep.py interact ner_ontonotes_bert [-d]
```

Где ner_ontonotes_bert имя конфигурации и [-d] необязательный ключ загрузки. Ключ [-d] используется для загрузки предварительно обученной модели вместе с вложениями и всеми другими файлами, необходимыми для запуска модели. Другие возможные команды «train», «evaluate» и «download».

Модель	Набор данных	Язык	Размер вложений	Размер модели	Оценка F1
ner_rus_bert	Сборник3 [1]	RU	700 МБ	1,4 ГБ	**98,1**
ner_collection3_m1			1,1 ГБ	1 ГБ	97,8
ner_rus			1.0 ГБ	5,6 МБ	95,1
ner_ontonotes_bert_mult	Ontonotes	Мульти	700 МБ	1,4 ГБ	**88,8**
ner_ontonotes_bert		En	400 МБ	800 МБ	88,6
ner_ontonotes_m1			347 МБ	379,4 МБ	87,7
ner_ontonotes			331 МБ	7,8 МБ	86,7
ner_conll2003_bert	CoNLL-2003		400 МБ	850 МБ	91,7
ner_conll2003_torch_bert			-	1,1 ГБ	88,6
ner_conll2003			331 МБ	3,1 МБ	89,9
conll2003_m1			339 МБ	359,7 МБ	**91,9**
ner_dstc2	DSTC2		-	626 КБ	97,1
vlsp2016_full	ВЛСП-2016	Vi	520 МБ	37,2 МБ	93,4

Рис. 1. Доступные конфигурации DeepPavlov

3. Томита-парсер

Томита-парсер – созданный компанией Яндекс вариант GLR-парсера (от англ. Generalized Left-to-right Rightmost derivation parser – Обобщенный восходящий магазинный анализатор), впервые описанного Масару Томита в 1984 году. В настоящее время открытый код парсера доступен для разработчиков в коммерческих и некоммерческих целях. В составе парсера три основных лингвистических процессора: токенизатор (осуществляет разбиение входного текста на слова и несловарные токены), сегментатор (разделяет текст на предложения) и морфологический анализатор mystem (производит частеречную разметку) [2].

Основными компонентами парсера являются: газеттир, набор контекстно-свободных (КС) грамматик (пользовательских шаблонов) и набор описаний типов фактов, которые могут фиксироваться

(порождаться) этими грамматиками в результате процедуры интерпретации.

Газеттир – словарь ключевых слов, которые используются в процессе анализа КС-грамматиками. Каждая статья этого словаря задает множество слов и словосочетаний, объединенных общим свойством (например, «мужские имена»)»).

Грамматика представляет собой множество правил на языке КС-грамматик, описывающих синтаксическую структуру выделяемых цепочек.

Грамматики для Томита-парсера состоят из правил. У каждого правила есть левая и правая части, разделенных символом —>. В левой части стоит один нетерминал (S в примере, приведенном ниже). В правой части стоит список терминалов или нетерминалов (S1 ... Sn), после которого указываются условия (Q), применяемые ко всему правилу в целом.

Грамматический парсер запускается всегда на одном предложении. Перед запуском терминалы грамматики отображаются на слова (или словосочетания) предложения. Одному слову может соответствовать много терминальных символов. Таким образом, парсер получает на вход последовательность множеств терминальных символов. На выходе – цепочки слов, распознанные этой грамматикой.

Факты – таблицы с колонками, которые называются полями фактов. Факты заполняются во время анализа парсером предложения. Как и чем заполнять поля фактов указывается в каждой конкретной грамматике (интерпретация). Типы фактов описываются в отдельном файле.

4. Библиотека Natasha

Библиотека Natasha решает базовые задачи обработки естественного русского языка: сегментация на токены и предложения, морфологический и синтаксический анализ, лемматизация, извлечение именованных сущностей [3].

При работе с данной библиотекой пользователь явно инициализирует компоненты: загружает предобученные эмбеддинги (результаты процесса преобразования языковой сущности – слова, предложения, параграфа или целого текста в наборы чисел – числовые векторы), передаёт их в конструкторы моделей. Сам вызывает методы segment, tag_morph, parse_syntax для сегментации на токены и предложения, анализа морфологии и синтаксиса.

Листинг 1

Пример использования библиотеки Natasha

```
from natasha import (Segmenter,NewsEmbedding,NewsMorphTagger,
NewsSyntaxParser, Doc )
segmenter = Segmenter()
emb = NewsEmbedding()
morph_tagger = NewsMorphTagger(emb)
syntax_parser = NewsSyntaxParser(emb)
text = 'Посол Израиля на Украине Йоэль Лион признался, что
пришел в шок, узнав о решении властей Львовской области
объявить 2019 год годом лидера запрещенной в России
Организации украинских националистов (ОУН) Степана Бандеры...'
doc = Doc(text)
doc.segment(segmenter)
doc.tag_morph(morph_tagger)
doc.parse_syntax(syntax_parser)
sent = doc.sents[0]
sent.morph.print()
```

Результат работы:

```
Посол       NOUN|Animacy=Anim|Case=Nom|Gender=Masc|Number=Sing
Израиля     PROPN|Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing
на          ADP
Украине     PROPN|Animacy=Inan|Case=Loc|Gender=Fem|Number=Sing
Йоэль       PROPN|Animacy=Anim|Case=Nom|Gender=Masc|Number=Sing
Лион        PROPN|Animacy=Anim|Case=Nom|Gender=Masc|Number=Sing
```

5. Yargy – парсер

На данный момент среди наиболее распространенных и некоммерческих парсеров для русского языка можно выделить Томита-парсер и Yargy. Однако Томита парсер для описания правил использует свой язык, а в Yargy все правила описываются на языке Python. Также в Томита-парсере все правила и грамматики являются закрытыми, а Yargy, напротив, представляет собой проект с исходным кодом, доступным на Github. Стоит отметить, что скорость работы Томита-парсера превосходит Yargy за счет того, что он реализован на C++.

Работа Yargy-парсера основана на контекстно-свободных грамматиках, словарях и правилах для извлечения структурированной информации из текстов на русском языке. Правила (rule) состоят из предикатов 6. 33].

Парсер имеет множество встроенных предикатов, таких как:

is_capitalized – слово начинается с большой буквы,

is_single – слово в единственном числе, gram (X), где X – это граммема.

Yargy-парсер использует библиотеку pymorphy для определения формы слова, gram(ADJF) – означает, что слово является прилагательным. У парсера есть метод findall, который возвращает список-результат сопоставлений.

В качестве примера рассмотрим задачу определения в тексте заказа наименований напитков. Перечень напитков определим в правиле R_1 через предикат dictionary, перед названием напитка должно идти прилагательное.

Листинг 2

Пример использования Yargy-парсера

```
from yargy import Parser, rule
from yargy.predicates import gram, dictionary

R_1 = rule (gram('ADJF'),
    dictionary({'сок', 'морс', 'компот'}))
parser = Parser(R_1)
text = 'В заказ на доставку входили апельсиновый сок, вишнёвый
морс и абрикосовый компот.'
for match in parser.findall(text):
    print ([x.value for x in match.tokens])
```

Результат:

```
['апельсиновый', 'сок']
['вишнёвый', 'морс']
['абрикосовый', 'компот']
```

Yargy-парсер – это мощный инструмент для извлечения структурированной информации из русско-язычных текстов. Вместе с тем, парсер реализован на python, что делает его использование в проекте чаще всего более удобным в сравнении с аналогами, такими, так Томита-парсер от Яндекс.

6. Программная реализация с использованием библиотеки NLTK

В этом проекте будет использоваться библиотека «NLTK» для извлечения сущностей из предложения и классификации их по категориям сущностей, таких как человек, место или вещь и т.д.

Токенизация исходного текста

NLTK tokenize sentences включает в себя два подмодуля:

– токенизация слов
– токенизация предложений

В нашей программе мы воспользуемся первым подмодулем word_tokenize(), чтобы разбить введенные пользователем предложения на слова. Выходные данные токенизации слов можно преобразовать во

фрейм данных для лучшего понимания текста в приложениях машинного обучения. Его также можно использовать в качестве входных данных для дальнейших шагов по обработке естественного языка [4]/

Текст, введенный в текстовое поле, передается в функцию word_tokenization, которая генерирует токены. Функция word_tokenize возвращает список строк.

Частеречная разметка

Частеречная разметка (автоматическая морфологическая разметка, POS tagging, part-of-speech tagging) – этап автоматической обработки текста, задачей которого является определение части речи и грамматических характеристик слов в тексте (корпусе) с приписыванием им соответствующих тегов. Список возможных обозначений частей речи в обрабатываемом текстовом массиве представлен на рис. 2.

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	*+,%, &*
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	*#*
PDT	Predeterminer	*all, both*	"	Left quote	' or "
POS	Possessive ending	*'s*	"	Right quote	' or "
PRP	Personal pronoun	*I, you, he*	(	Left parenthesis	[, (, {, <
PRP$	Possessive pronoun	*your, one's*	)	Right parenthesis	],), }, >
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	. ! ?
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	: ; ... – -
RP	Particle	*up, off*			

Рис. 2. English POS-tagging

Список, возвращаемый функцией word_tokenize, далее передается функции pos_tag. Эта функция распознает части речи каждого токена, к которой он принадлежит.

Функция ne_chunk

Список токенов вместе с их частями речи, передаются в функцию ne_chunk. Поскольку все сущности являются собственными существительными, такими как имена людей, названия мест, геополитические объекты и денежные суммы, мы рассматриваем только те лексемы, которые соответствуют тегу NNP после возвращения из функции ne_chunk. Результаты работы функции приведены на рис.3.

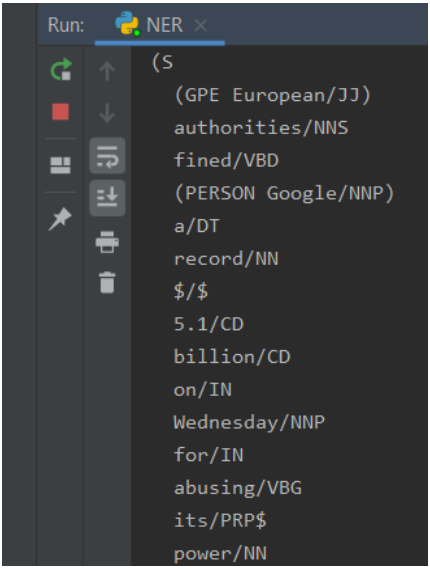


Рис. 3. Результат работы функции ne_chunk

Обработка именованных сущностей

Из строки, сгенерированной функцией ne_chunks, сущности, содержащие похожие теги, объединяются. Пример: (person Christoper/NNP).

Сущности, полученные из текстового поля, сравниваются со строкой возвращаемой функцией ne_chunk, путем обхода строки и поочередного сравнения. Поскольку неоднозначности не удалось избежать, некоторые сущности могут быть неправильно определены. Например, имя Washington может быть человеком или штатом в зависимости от контекста, в котором оно используется. Числовые значения, такие как деньги и проценты, помечаются тегом 'CD', лица помечаются тегом 'PERSON', страны и места – 'GPE' (geo political entity).

Исходный текст:

```
«A multi-agency manhunt is under way across several states and
Mexico after police say the former Los Angeles police officer
suspected in the murders of a college basketball coach and her
last weekend is following through on his vow to kill police
officers after he opened fire Wednesday night on three police
officers, killing one. "In this case, we're his target," Sgt.
Rudy Lopez from the Corona Police Department said at a press
conference. The suspect has been identified as Christopher
Jordan Dorner, 33, and he is considered extremely dangerous
and armed with multiple weapons, authorities say. The killings
appear to be retribution for his 2009 termination from the Los
Angeles Police Department for making false statements,
authorities say. Dorner posted an online manifesto that
warned, "I will bring unconventional and asymmetrical warfare
to those in LAPD uniform whether on or off duty.»
```

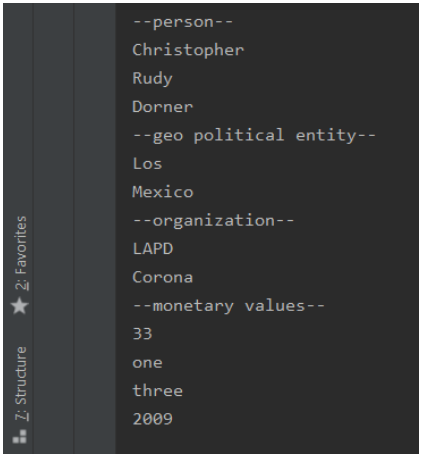


Рис. 4. Результат работы функции по типизации выделенных именованных сущностей

Дерево синтаксического анализа

Для более детального анализа качества построения дерева синтаксического анализа воспользуемся более коротким текстом, состоящим из одного предложения, вида:

«I want to kill my brother».

В этом предложении имеем именную группу «my brother» и глагольную группу «want to kill».

Пример построения дерева синтаксического анализа без регулярного выражения представлен на рис. 5.

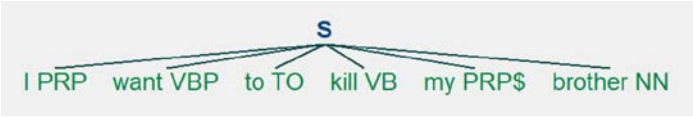


Рис. 5. Пример построения дерева синтаксического анализа на встроенных грамматиках

При использовании встроенных грамматик дерево синтаксического анализа не имеет должной глубинной структуры. Для правильной работы автоматического анализа необходимо вручную задать регулярные выражения, для схлопывания словесных последовательностей в группы.

Регулярное выражение вида:

```
reg_exp1 = r''' NP: {<DT>? <PRP\$>? <JJ>* <NN.?>}'''
```

Означает, что именная группа – это последовательность типа детерминат, состоящая из притяжательного местоимения, неопределенного количества прилагательных и существительного в некой форме.

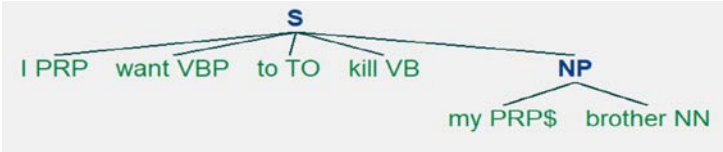


Рис. 6. Пример построения дерева синтаксического анализа с регулярным выражением reg_exp1

Регулярное выражение для глагольной группы вида:

```
reg_exp1 = '''VP:{<RB>?<VB.?><NP>?}'''
```

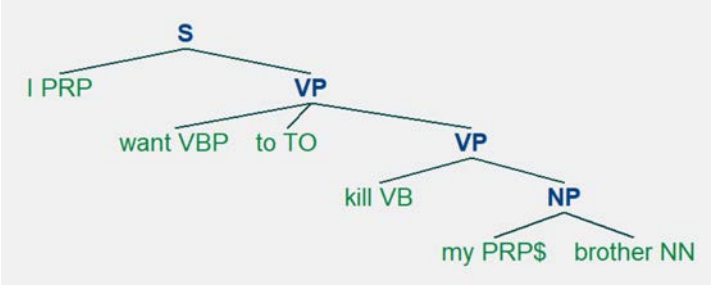


Рис. 7. Результат построения правильного дерева синтаксического анализа

Заключение

Данная статья посвящена проведению обзора существующих инструментов выделения именованных сущностей и реализации программного модуля для извлечения именованных сущностей, по генерации дерева разбора, а также сбора статистической информации для анализа частеречной в предложенном для разбора тексте. В итоге данной разработки получилось полностью рабочие программное решение, реализованное с использованием библиотеки NLTK. В дальнейшем предполагается использование данного модуля в системе обработки разнородных документов, в которых данные могут представлены в структурированном-табличном и неструктурированном виде.

Список литературы

1. Steven Bird, Natural Language Processing with Python / Steven Bird, Ewan Klein, Edward Loper. – O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol , 2009. – 179–205 с.
2. Томита-парсер Руководство разработчика Версия 1.0 [Электронный ресурс] Документ является составной частью технической документации Яндекса. – Режим доступа: https://yandex.ru/dev/tomita/doc/dg/concept/about.html
3. Проект Natasha – набор Python-библиотек для обработки текстов на естественном русском языке [Электронный ресурс]: Руководство разработчика. – Режим доступа: https://natasha.github.io/
4. Steven Bird, NLTK: The natural language toolkit / Steven Bird, Edward Loper. – The Association for Computer Linguistics – International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics , Sydney, Australia, 2006. – 2 с.

Идентификация фишинговых сайтов

Ю. В. Шишко

Студент бакалавриата

Д. И. Соломатин

Старший преподаватель

Введение

Несомненно, сейчас интернет стал куда безопаснее, чем десять лет назад, в частности и сегмент рунета. С одной стороны, этому способствовало развитие в России «опорных столбов» в виде компаний, предоставляющих антивирусы, как, например, Лаборатория Касперского; с другой стороны, нынешние браузеры стали куда больше внимания уделять тому, куда переходят их пользователи: при переходе на подозрительные ресурсы всплывают предупредительные баннеры, причем список таких «подозрительных» ресурсов постоянно обновляется.

Тем не менее, каким бы безопасным не казался этот «современный» интернет, то там, то тут появляются новости – мошенники снова нажились на людях, которые не убедились в том, что переводят деньги, куда нужно. И хотя казалось бы – время больших баннеров с «перейди и получи приз» уже прошло, да и пользователи стали опытнее, осознаннее – тысячи людей ежедневно попадаются на фишинг.

Мошенники подделывают страницы банков, интернет-магазинов, делают зеркальные копии с похожими названиями – и человек уже не замечает, что переходит не на sberbank, а на sberbnak, например. Скопированный в точности интерфейс не вызывает никаких подозрений, разве что работает, может быть, чуть медленнее обычного – все действия и введенные данные нужно передать на оригинал, получить ответ и отдать пользователю – но опять же, пользователь скорее спишет это все на проблемы с интернет-соединением.

К слову, в 2020 году во время пандемии коронавируса количество краж с банковских карт пользователей выросло в шесть раз, сообщала компания Group-IB[1], которая специализируется на предотвращении кибератак. По словам экспертов, мошенники заманивают пользователей на фишинговые сайты, где покупатели вводят платежные данные.

Злоумышленники используют эти данные для обращения к публичным p2p-сервисам банков и перевода денег на свои счета.

Именно поэтому было решено создать систему для идентификации фишинговых сайтов. Система должна представлять собой API, web-приложение и расширение для браузера, чтобы быть удобной и доступной. Реализацию этих компонент рассмотрим ниже.

1. Реализация API

Вне зависимости от того, с какой платформы будет приходить запрос – будет это обращение к API напрямую с помощью Postman, например, запрос из веб-приложения или запрос, отправленный из расширения, работа с ним будет проходить по одному и тому же сценарию (см. рис. 1).

Рис. 1. Общая схема работа API

Приходящий в параметрах запроса url проверяется на корректность, после чего происходит парсинг контента, производятся запросы к сторонним ресурсам и проверка параметров url. Результат проверки отправляется в обученную модель для классификации. Сформированный таким образом результат сохраняется в базу данных для возможности дальнейшего анализа и внесения корректировок в работу алгоритма, а также выдается пользователю отчетом[2].

Для реализации сервера используются Flask, sqlalchemy и mysql. Парсинг контента реализуется с помощью BeatifilSoup – анализируются ссылки, на которые ведет сайт, проверяется, не является ли сайт копией, созданной с помощью HTTrack[3].

Запросы к сторонним ресурсам представляют собой работу с GoogleSafeBrowsing. Этот сервис также предоставляет доступ к информации об известных фишинговых сайтах.

Подробнее о том, какие параметры url исследуются в ходе обработки, будет описано в рамках обучения SVM.

2. SVM

В рамках задачи необходимо было сделать бинарный классификатор, предполагающий, является ли сайт фишинговым. В рамках текущей реализации за классификацию отвечает метод опорных векторов.

Метод опорных векторов (англ. SVM, support vector machine) — набор схожих алгоритмов обучения с учителем, использующихся для задач классификации и регрессионного анализа[4].

Для классификации мною были выбраны ряд параметров:

- проверка умышленных опечаток с использованием расстояния Дамерау-Левентшейна;
- проверка схемы с достаточно известной первой частью URL;
- поддельный префикс www-
- проверка домена;
- проверка перенаправлений с помощью “@”;
- проверка популярных сайтов.

Расстояние Дамерау-Левенштейна позволяет сравнивать слова различной длины для проверки на схожесть с другими сайтами, в том числе намеренные опечатки, такие как sbernak и прочие. Пример схемы с известной первой частью url можно увидеть на рис. 2.

http://paypal.com-webappsuserid29348325limited.active-userid.com/webapps/89980/

protocol	http://
Domain name	active-userid.com
path	/webapps/89980/
Subdomain item1	com-webappsuserid29348325limited
Subdomain item2	paypal

Рис. 2. Схема с известной частью суб-домена

Обучение происходит с помощью библиотеки sklearn и модуля svm. В качестве ядра используется linear, для тестовых данных был использован majestic million (1000000 url для обучения) и список фишинговых сайтов из публичного репозитория (331000 url для обучения).

Помимо этих параметров, были также добавлены несколько базовых параметров – длина url, количество под-доменов, доступность сервиса, наличие проблем с сертификатом.

При разделении выборки на обучающую (80%) и тестовую (20%) была получена точность предсказания 92%. Проблема при дальнейшем использовании может состоять в том, что сайты из списка фишинговых могут блокироваться и доступность сервиса будет иметь больший вес, чем необходимо, при итоговом принятии решения, из-за чего реальная точность при такой модели может быть ниже указанной.

3. Web-приложение

Основной платформой для использования системы является web-приложение. Ему было отдано предпочтение ввиду удобства: подходит для любой машины, нет необходимости скачивать, доступ с любого браузера, который установлен на устройстве пользователя.

Клиент-серверное приложение выполнено с помощью Flask-templates. Заходя на страницу, пользователь видит окно для ввода ссылки. После нажатия на кнопку «проверить ссылку», с веб-страницы отправляется post-запрос к api. Полученный ответ отображается на странице с подробным отчетом, как на рис. 3.

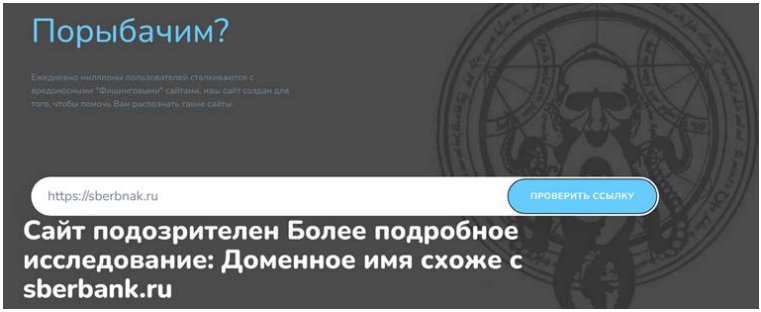


Рис. 3. Работа web-серверного приложения

4. Расширение для браузера

Дополнительной платформой для использования системы было выбрано расширение для браузера ввиду нарастающей популярности и удобства использования.

Для написания расширения для браузера был использован Chrome Extensions. На данный момент большинство популярных браузеров работают на базе Chromium – Google Chrome, Яндекс.Браузер, Opera, Edge.

При нажатии на иконку приложения на сервер отправляется запрос с url с текущей страницы пользователя. Открывается всплывающее окно с текстом «Происходит сканирование страницы…» (рис. 4).

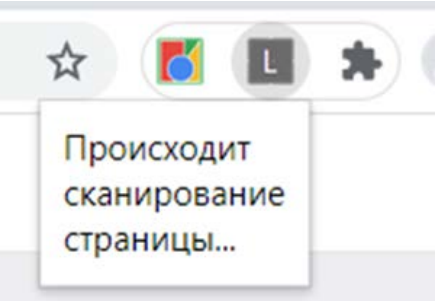


Рис. 4. Запуск расширения для браузера

Как только приходит ответ с сервера, содержимое всплывающего окна перерисовывается – на нем отображаются вердикт относительно сайта и результаты более подробного исследования url. Работа расширения во время нахождения пользователя на подозрительном сайте продемонстрирована на рис. 5.

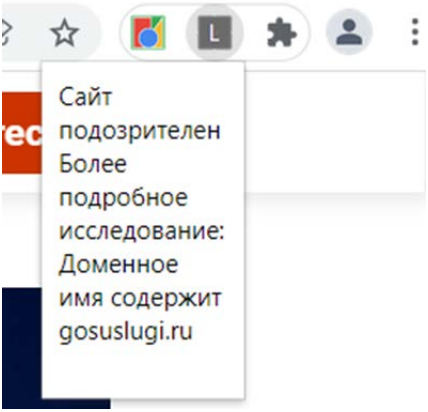


Рис. 5. Скриншот обнаружения потенциально фишингового сайта

Заключение

Полученная система покрывает большое количество параметров “вредоносности” сайта, может защитить пользователя от кражи его

персональных данных или предупредить о том, что ресурс мошеннический и не стоит ему доверять. Разнообразие способов получения доступа к анализу делает систему не только полезной, но и удобной для использования.

Список литературы

1. Отчет Group-IB, Hi-Tech Crime Trends 2019/2020 [Электронный ресурс] : Электрон. журн. – Режим доступа : https://www.group-ib.ru/resources/threat-research/2019-report.html

2. Гергель, В.П. Современные языки и технологии параллельного программирования: Учебник / В.П. Гергель. – М.: МГУ, 2016. – 408 с.

3. Голицына, О.Л. Языки программирования: Учебное пособие / О.Л. Голицына, Т.Л. Партыка, И.И. Попов. – М.: Форум, НИЦ ИНФРА-М, 2017. – 400 с.

4. Плас, Дж. Python для сложных задач. Наука о данных и машинное обучение / Дж. Вандер Плас . – М.: Питер, 2017. – 518 с.

Способы представления в кэш-хранилище данных произвольной размерности

А. Ю. Яровик

Студент бакалавриата

Н. К. Самойлов

Старший преподаватель

Введение

Одним из ключевых требований к приложению, реализующему кэш-хранилище, является требование возможности задания пользователем кэш-хранилища произвольной структуры хранимых объектов. Действительно, не имея полной информации о предметной области, являющейся сферой интересов пользователя хранилища, невозможно предугадать все возможные желаемые комбинации состояний хранилища. Поэтому необходимо реализовать возможность хранения объектов любой заданной структуры.

В настоящей статье в качестве структуры данных для хранения кэшируемых объектов будет рассматриваться R-дерево – древовидная структура данных, предложенная А. Гуттманом в 1984 году [1], основанная на разбиении признакового пространства на иерархически вложенные n-мерные параллелотопы. Пример подобного рода структуры представлен на рис. 1.

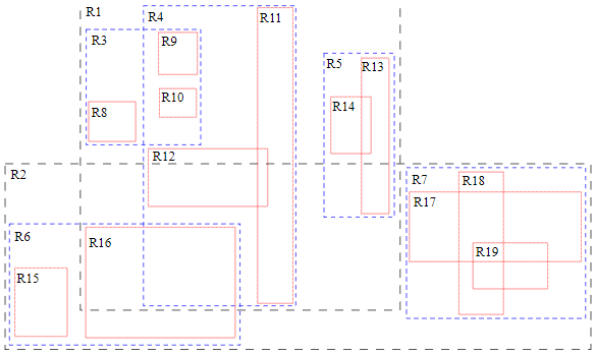


Рис. 1. Пример *R*-дерева

Как видно из рисунка выше, проблемой здесь является то, что в R-дереве при индексации объектов выступают координаты в пространстве. Именно отсюда и вытекает очевидное использование R-деревьев в качестве хранилищ для топографических данных. Однако, когда речь заходит о хранении объектов произвольного типа, неочевиден способ представления такого рода объектов в хранилище.

1. Описание операций в R-дереве

Для более полного понимания задачи, рассмотрим основные доступные нам операции над R-деревом – вставку данных, разделение узла и поиск:

1. Вставка: для вставки данных в R-дерево используется структура данных “узел” со следующим набором полей:

– float[] coords – массив координат вершин области подпространства узла;

– float[] dimensions – массив размерностей области подпространства узла;

– Node parent – ссылка на родительский элемент узла;

– Node[] children – массив ссылок на дочерние элементы узла;

– boolean leaf – булева величина, определяющая, является ли узел “листовым” либо “внутренним”.

2. Поскольку узлы могут перекрываются, конкретный объект данных может быть выбран для вставки в несколько узлов [2]. Процедура вставки начинается с корня. Если существует только один дочерний элемент, куда можно вставить узел, вставка делегируется этому элементу. В противном случае из нескольких кандидатов выбирается такой дочерний элемент, при вставке в который произойдет наименьшее увеличение объема.

3. Разделение: разбиение элемента с $\beta + 1$ дочерних элементов на два элемента, с количеством дочерних элементов не меньших α. В используемой реализации применяется линейный алгоритм разбиения.

4. Поиск: входными данными для поиска является прямоугольник поиска, который задает желаемую область подпространства, в которой мы ищем данные. Начиная с корня дерева, проверяется, перекрывает ли область узла прямоугольник поиска, и в случае успеха процедура повторяется рекурсивно для дочерних элементов узла. Таким образом будут найдены все узлы дерева, соответствующие заданной области поиска [3].

2. Представление многомерных данных в R-дереве

Для решения описанной выше проблемы была принята следующая методология представления и выборки объектов из R-дерева.

1. Задаются домены для типов данных, которым могут принадлежать атрибуты хранимых в кэше объектов, к примеру такие как “символ” или “число”.

2. Каждому из этих доменов сопоставляется общий для конкретного домена однозначный способ преобразования атрибутов, принадлежащих данному домену, в значение некой координаты в одномерном пространстве. Иными словами, выбирается некая функция f(x), удовлетворяющая условиям:

$$\forall x \in D \exists! a = F(x) \tag{2}$$

$$\forall x \in D \lim_{\epsilon \to 0} F(x - \epsilon) = 0 \tag{3}$$

3. Здесь в качестве x выступает переменная, принимающая значения заданного домена, при этом домен может описывать любой скалярный тип данных.

4. Для конкретного объекта, содержащегося в R-дереве, по значению всех его атрибутов при помощи заданного для каждого домена правила сопоставляются его координаты в многомерном пространстве R-дерево. Количеством же измерений этого пространства будет служить количество атрибутов объекта.

5. При добавлении объекта в R-дерево, координаты этого объекта задаются по принципу, описанному в п.3

6. При выполнении операции выборки данных из R-дерева, условия поиска (или другими словами, предикаты), преобразуются в операции сравнения координат в соответствии с доменами этих предикатов по принципу, описанному в п.2. Таким образом, критерии выборки (возможно, составные) данных из R-дерева преобразуются в совокупность координат, описывающих некое подпространство R-дерева, которое и будет содержать искомые объекты (если они, конечно, есть).

Ниже приведено рассмотрение реализации описываемого принципа на примере.

Пусть в кэш-хранилище представлены объекты типа “Товар” следующей структуры:

– атрибут “Цена”, принадлежащий домену целых чисел, описывающий стоимость единицы товара. Правило преобразование значений данного домена в координаты пространство тривиально – целому числу соответствует некая координата оси абсцисс;

– атрибут “Качество”, принадлежащий домену типа “Качество”, принимающему значения {Excellent, Good, Normal} и описывающую субъективную оценку удовлетворенности от использования товара. Можно отыскать множество правил, которые могли бы описать

преобразование значений данного домена в координаты пространства, зададим одно из таких правил следующим образом:

$$f(x) = \begin{cases} -1, & x = \text{Normal} \\ 5, & x = \text{Good} \\ 7.5, & x = \text{Excellent} \end{cases} \quad (4)$$

Предположим, в хранилище содержатся следующие объекты типа “Товар”:

– Товар А (Цена: 10, Качество: Good);
– Товар Б (Цена: 20, Качество: Excellent);
– Товар В (Цена: 5, Качество: Normal).

Поскольку данный объект описывается при помощи набора из двух атрибутов, описываемый выше принцип предписывает задать пространство всех объектов данного типа в виде обыкновенной плоскости (двухмерного пространства). Объекты типа “Товар” будут представлены в этом пространстве в виде точек, абсцисса которой будет соответствовать атрибуту “Цена”, ордината – атрибуту “Качество”, а условие выборки товаров – прямоугольнику пространства. Графическая иллюстрация пространства товаров с принадлежащими ему товарами А, Б и В представлена на рис. 2.

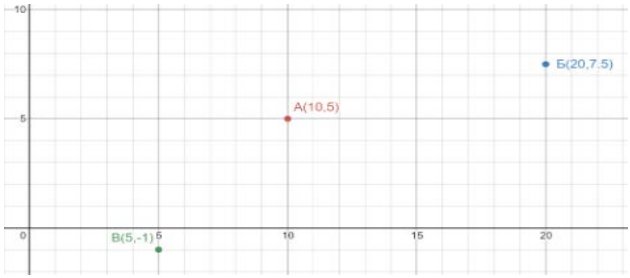


Рис. 2. Графическая иллюстрация принципа представления объектов в R-дереве на примере объектов типа “Товар”

Как видно из рис. 2, объекты типа “Товар” представлены в декартовой системе координат; товар “А” имеет абсциссу, равную 10, ординату, равную 5, товар “Б” имеет абсциссу со значением 20 и ординату со значением 7.5, и товар “В” имеет координаты, равные 5 и -1 соответственно.

Допустим, требуется извлечь объекты со значением цены от 0 до 10 и со значением качества от 1 до 5. Тогда данному критерию выборки будет соответствовать прямоугольник, изображенный на рис. 3.

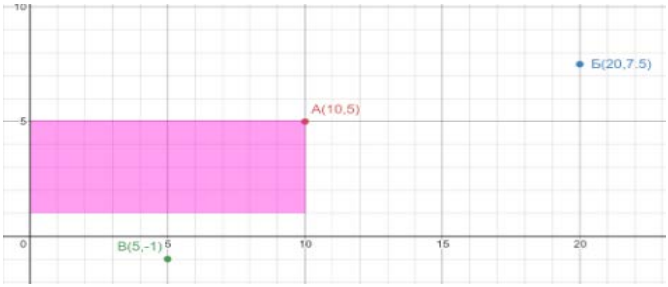


Рис. 3. Графическая иллюстрация принципа представления критерия выборки объектов типа “Товар”

На рис. 3 видно, как закрашенная фиолетовым цветом область соответствует описанному выше критерию. В эту область попадает единственный товар “A”, которые и соответствует одновременно двум условиям, описанным в критерии.

Заключение

Таким образом, описываемый выше принцип представления объектов в R-дереве и условий выборки из R-дереве в виде координат, а самого R-дерева – в виде N-мерного пространства, которое содержит хранимые в R-дереве объекты, позволяет добиться требуемой от приложения, реализующего кэш-хранилище, универсальности: возможности представления в кэш-хранилище объектов любой структуры, заданной пользователем этого хранилища. Это позволяет использовать разрабатываемое приложение в предметным областях самого широкого спектра, и работать с получившимся кэш-хранилищем, используя привычные принципы работы с данными, которые скрывают от пользователя пространственную сущность R-дерева, используемого в качестве основной структуры данных хранилища.

Список литературы

1. Guttman, A. R-TREES. A DYNAMIC INDEX STRUCTURE FOR SPATIAL SEARCHING / A. Guttman. – New York : ACM, 1984. – 47 p.

2. Bhattacharya, A. Fundamentals of Database Indexing and Searching / A. Bhattacharya. – Boca Raton : CRC Press, 2015. – 76 p.

3. Гулаков, В.К. Многомерные структуры данных / В.К. Гулаков, А.О. Трубаков. – Брянск : БГТУ, 2010. – 248 с.

Статьи студентов магистратуры

Использование глубоких нейронных сетей в задаче визуального фенотипирования тепличных растений

Т. В. Акиндинова

Студент магистратуры

М. А. Дрюченко

Доцент

Введение

В настоящее время одной из сфер применения машинного зрения является сельское хозяйство. Крупные аграрные предприятия с теплицами 5го поколения используют автоматические аппаратные и программные комплексы, способные отслеживать состояние буквально каждого растения и подсвечивать о проблемах заблаговременно. Работа таких современных тепличных программно-аппаратных комплексов начинается со съёмки растений с помощью роботизированной платформы, далее полученные изображения попадают в программный комплекс, который проводит операцию визуального фенотипирования растений. Итогом работы являются конкретные интересующие специалистов предприятий численные признаки, отражающие состояние растений.

Данная работа посвящена исследованию возможностей использования глубоких нейронных сетей в задаче визуального фенотипирования растений. В общем случае процедура визуального фенотипирования состоит из нескольких этапов, одним из которых является сегментирование полезного объекта от сложного фона. Этот этап необходимо проводить с целью уменьшения реально значимой информации для дальнейшего упрощённого процесса декомпозиции на отдельные части растений и непосредственного определения биометрических параметров растений. Ниже рассмотрены три различных алгоритма сегментации полезного объекта от фона с использованием глубоких нейронных сетей.

1. Алгоритм на основе сверточных нейронных сетей и прямоугольных фрагментов в качестве входных данных

Данный алгоритм подразумевает использование сверточных нейронных сетей (НС) и прямоугольных фрагментов в качестве

обучающих данных. Сеть на выходе возвращает «ответы» о принадлежности каждого такого фрагмента к одному из заданных классов (фон или объект). Достоинствами алгоритма являются относительная простота подготовки обучающей выборки и полное покрытие кадров за счёт непересекающихся фрагментов. К недостаткам алгоритма относятся низкая точность распознавания вдоль сложных по форме границ объектов и требование проведения ручной разметки.

Для решения задачи сегментации тепличных культур томатов будут использоваться типовые архитектуры НС, построенных из сверточных и субдискретизационных слоёв с последующими функциями активации [1]. После построения сверточной нейронной сети нужно её обучить. Для этого на изображениях, размеченных по сетке с фиксируемым шагом, проводится ручная разметка, путём нажатия левой и правой кнопок мыши. В процессе такой разметки выделенные фрагменты, принадлежащие одному классу, закрашиваются одним цветом. Промежуточный этап формирования обучающих данных представлен на рис. 1.

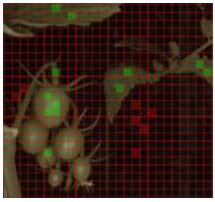


Рис. 1. Ручная разметка растровых фрагментов по классам

Из отдельных фрагментов формируется хранилище паттернов, которое представляет собой вырезанные прямоугольные фрагменты фиксированного размера с различных растровых изображений. Пример таких фрагментов приведён на рис. 2.

Рис. 2. Прямоугольные фрагменты растра для обучающей выборки, слева – объект, справа – фон

При небольшом количестве обучающих данных или недостаточном многообразии его всегда можно пополнить через упомянутую выше ручную разметку. При этом можно ещё регулировать сам размер

обучающих данных. Обычно фрагменты имеют размеры 30-50 пикселей. При обучении необходимо также задать количество для валидационных данных. Для хороших результатов используют 10-15% от общего объёма данных. Также имеется возможность дообучения уже существующего классификатора или использование сохранённой нейронной сети без изменений. При тестировании алгоритма пользователь выбирает растровое изображение, которое отображается с сеткой, где цвет каждого прямоугольного фрагмента говорит о принадлежности к одному классу. Результат работы алгоритма представлен на рис. 3.

слева – изображение до сегментации, справа – после сегментации

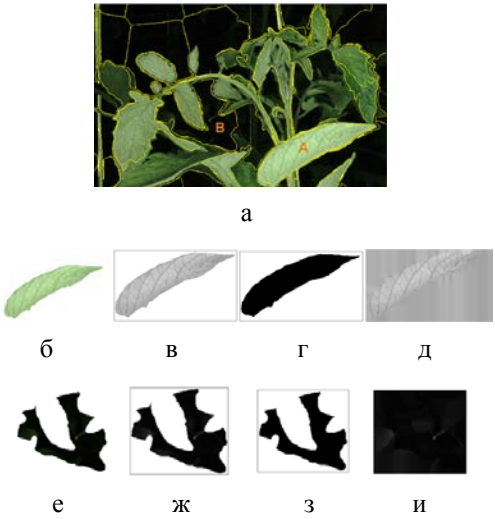
Рис. 3. Результат работы сегментации на основе сверточных нейронных сетей и прямоугольных фрагментов в качестве входных данных

2. Алгоритм сегментации объектов с использованием суперпиксельной сегментации и сверточных нейронных сетей

Второй алгоритм основан на суперпиксельной сегментации и сверточных нейронных сетях. Его достоинством является высокая точность распознавания вдоль границ объектов, а недостатками – относительно большое время работы и сложность в формировании обучающих данных. В качестве суперпиксельного алгоритма я использовала алгоритм SLIC [2]. Для формирования обучающих паттернов загружаются растровые изображения кустов, затем проводится суперпиксельная сегментация, которая значительно уменьшает количество информации на исходных изображениях, сохраняя при этом сложные контуры отдельных частей кустов.

После получения суперпикселей на изображениях происходит ручная разметка по классам для формирования обучающей выборки. Делается это, как и в первом методе при помощи нажатия левой или правой кнопки мыши. Однако неправильная форма суперпикселей не

позволяет напрямую подавать их на вход классификатора из сверточной нейронной сети. Поэтому необходимо каждый отдельный суперпиксель вписать в минимально возможный прямоугольник. Недостающие пиксели изначально заполняются белым цветом. Затем формируется бинарная маска заполнения, где объектом будут считаться именно недостающие белые пиксели. Завершительным этапом подготовки входных данных в рамках данного подхода является подача вписанного в прямоугольник суперпикселя и его бинарной маски на вход алгоритма интерполяции. Весь процесс изображен на рис. 4.



а - пример работы алгоритма SLIC; б, е - исходные суперпиксели объекта и фона; в, ж - вписанные в градациях серого суперпиксели объекта и фона; г, з - бинарные маски объекта и фона; д, и - результат работы интерполяционного алгоритма для суперпикселей объекта и фона

Рис. 4. Работа алгоритма

После обучения нейронного классификатора производится тестирование работы алгоритма на новом растровом изображении, которое также изначально сегментируют на суперпиксели. После этого каждый из них приводится к прямоугольной форме при помощи интерполяции и подаётся на вход обученной нейронной сети. Выходом является ответ относительно каждого суперпикселя, затем в

соответствие с ответом суперпиксели одного класса будут закрашены в один цвет. Результат работы алгоритма представлен на рис. 5.

слева – сегментированное на суперпиксели исходное изображение, справа – результат после классификации нейронной сети

Рис. 5. Результат работы алгоритма сегментации объектов с использованием суперпиксельной сегментации и сверточных нейронных сетей

3. Алгоритм на основе сверточных нейронных сетей класса U-Net

Данный алгоритм основывается на применении сверточных сетей архитектуры U-Net и размеченным по классам изображениям. Достоинством данного подхода является высокая точность распознавания вдоль границ объектов, а недостатком – сложность в формировании данных для обучения. U-Net отличается от типовых сверточных архитектур тем, что на выходе она имеет изображение с цветовой разметкой по классам того же размера, что и входное растровое изображение.

Сначала U-Net строится по сужающемуся пути, типичному для сверточной нейронной сети. После этого идёт расширение, на каждом этапе которого происходит сначала деконволюция, а затем объединение с соответствующим образом обрезанной карты свойств из сужающего пути [3]. В результате работы такой нейронной сети мы получаем изображение, на котором различным цветами показаны области принадлежности пикселей различным классам. В нашем случае это будут бинарные изображения, содержащее в себе 2 класса: объект и фон.

Чтобы заполнить хранилище паттернов, необходимо выбрать исходные растровые изображения и для каждого из них сформировать эталонную бинарную разметку. После этого пользователь выбирает

произвольные фрагменты на исходном изображении при помощи щелчка мыши с двигающимся окном, тот же фрагмент вырезается и с размеченного изображения. Этот процесс можно провести и без помощи пользователя: окно будет скользить по растровому и размеченному изображениям и вырезать одинаковые по величине и расположению фрагменты, которые и пойдут в качестве обучающих данных входа и выхода соответственно. Для более качественного обучения можно задавать шаг скольжения меньше, чем размер окна, тогда фрагменты будут частично перекрываться между собой.

При проведении тестирования изображение разделяется на прямоугольные непересекающиеся части, которые подаются на вход обученного классификатора. Итогом работы будет размеченный по классам фрагмент. Затем все части последовательно соединяются в целое изображение и отображаются в качестве результата. Итог работы данного алгоритма представлен на рис. 6.

Рис. 6. Результат работы алгоритма сегментации объектов с изначальной разметкой данных по классам и сверточных нейронных сетей архитектуры U-Net

Тестирование разработанных алгоритмов сегментации

Для оценки точности распознавания была разработана программа, реализующая рассмотренные выше алгоритмы. Формула для вероятности ошибки распознавания реальных объектов изображения I имеет следующий вид:

$$\alpha(I) = \frac{S(I_V \oplus I_B)}{\max(S(I_V), S(I_B))} \quad (1)$$

где I_B – результат сегментации изображения I одним их тестируемых алгоритмов, I_V – корректный результат сегментации, точно передающий пространственные характеристики растительных объектов,

S – совокупная площадь областей принадлежащих объектам на сегментированном изображении.

Общие результаты тестирования первого алгоритма сегментации на полных растровых изображениях показали, что точность распознавания может достигать 91%. Среднее время, необходимое для сегментации одного изображения, около 50 секунд. Однако многие биометрические признаки (ширина стебля, количество плодов и тп.) нельзя посчитать, так как форма объектов имеет очень грубое приближение.

Следующий алгоритм с использованием суперпиксельной сегментации и сверточных нейронных сетей работает дольше других алгоритмов (около 9 минут в среднем) и позволяет достичь точности лишь в 85-87%. Ошибки увеличиваются за счёт объединения в суперпиксели, которые содержат в себе значительно больше пикселей нежели один фрагмент предыдущего алгоритма. Поэтому при распознавании точность уменьшается. Алгоритм часто пропускает тонкие фрагменты стеблей кустов.

Заключительный алгоритм на основе сверточных нейронных сетей класса U-Net и изначальной разметкой данных по классам также позволяет получать более точные результаты распознавания вдоль границ объектов, нежели первый алгоритм. При наличии качественных эталонных сегментированных изображений и хорошо обученной сети точность распознавания в отдельных случаях достигает 95%. Время работы алгоритма для одного тестируемого изображения в среднем примерно 1 мин 50 секунд.

Интегрированные результаты распознавания и время работы трёх алгоритмов сегментации показаны в таблице ниже.

Таблица

Средние ошибки и среднее время работы алгоритмов сегментации полезного объекта от фона

	Алгоритм на основе сверточных нейронных сетях и прямоугольных фрагментов	Алгоритм с использованием суперпиксельной сегментации и сверточных нейронных сетей	Алгоритм на основе сверточных нейронных сетей класса U-Net
Средняя ошибка	9-11%	13-15%	5-6%
Среднее время работы (для одного изображения)	50 сек	9 мин	1 мин 50 сек

Заключение

В данной работе рассмотрены возможности использования глубоких нейронных сетей в задаче визуального фенотипирования тепличных растений кустов томатов. Также была разработана программная реализация, позволяющая получить численные значения ошибок распознавания. Результаты показали, что максимальная точность сегментации может достигать 95%. Выбор конкретного алгоритма сегментации полезного объекта от фона зависит от специфики конкретной задачи. Например, в случае необходимости подсчёта общей площади объекта рациональнее использовать наиболее быстроработающий алгоритм на основе сверточных нейронных сетей и прямоугольных фрагментов. Если же нужно определить высоту стебля, диаметр стебля под цветущей кистью, количество плодов и тд., то лучше использовать алгоритм с классификатором сети класса U-Net. Для определения точных контуров объектов, в особенности листьев, и дальнейшего анализа стоит воспользоваться алгоритмом с применением суперпиксельной сегментации.

Список литературы

1. Гудфеллоу, Я. Глубокое обучение / Я. Гудфеллоу, И. Бенджио, А. Курвилль. – 2-е изд., испр. – М. : ДМК Пресс, 2018. – 652 с.

2. Саввин, С. В. Методы суперпиксельной сегментации и их приенение для анализа изображений с разнородной текстурой / С. В. Саввин, А. А. Сирота // Вестник ВГУ, Серия: системный анализ и информационные технологии. – 2016. – № 4.

3. Применение нейронной сети архитектуры U-Net для сегментации СТМ-изображений / Е.Ю. Шелковников [и др.] // Химическая физика и мезоскопия. – 2019. – Т. 21. – № 2. – С. 330-336.

Жадный алгоритм поиска наиболее значимых трехмерных моделей лица

А. Д. Борисов

Студент магистратуры

Д. И. Соломатин

Старший преподаватель

Введение

В наше время очень важной частью компьютерной графики является создание цифровых дублеров для кино и игр. Эта потребность порождает ряд задач в сфере компьютерной графики, направленных на автоматизацию и улучшение качества цифровых дублеров. Большая часть этих задач строится на алгоритмах, для работы которых требуется отсканированные ключевые выражения лиц конкретного актера. Эти кадры получаются посредством фотограмметрической съемки и последующей обработки трехмерных сканов.

Чем больше ключевых кадров, тем решение правдоподобнее, но одновременно с этим падает скорость работы. Поэтому остро стоит вопрос уменьшения количества ключевых кадров, но при этом сохранения самых значимых из них. Этой проблеме и посвящена данная работа. В ее основе лежит поиск наиболее значимых трехмерных моделей лица из набора с помощью жадного алгоритма.

1. Существующие решения

Для решения данной задачи мог бы подойти метод главных компонент [1], который позволяет уменьшить размерность с сохранением ключевой информации. Однако, такой подход актуален, если допускается возможность потери согласованности с изначальными данными, из которых извлекались наиболее значимые компоненты.

К примеру, метод главных компонент может использоваться для создания линейной комбинации ключевых кадров, чтобы понизить размерность задачи. Такого рода проблемы возникают в задачах реконструкции лица по изображению или трехмерному скану [2]. Изначально число кадров, которые участвуют в линейном смешивании велико, и, чтобы упростить задачу, оставляют только самые главные

компоненты с помощью PCA. Но стоит учитывать, что из-за большого количества изначальных данных, этот метод может работать медленно.

Но в задачах, где важна семантика данных, этот подход не работает. К примеру, поиск самых значимых моделей лица, чтобы извлечь из них соответствующие текстуры для последующей работы. Если сжать методом главных компонент серию геометрий, то невозможно получить соответствия между новыми геометриями их оригинальными текстурами. Это допустимо, только при аналогичном сжатии самих текстур, но такие изображения будут слишком размытыми [3]. В предлагаемом же подходе исходные данные никак не изменяются, а только проходят сегрегацию по уровню значимости.

2. Описание жадного алгоритма

На вход алгоритму идет набор ключевых кадров. Для демонстрации используем геометрии человеческого лица, полученные после обработки трехмерных сканов. Эти сканы могут быть получены в результате фотограмметрического или же видеограмметрического сканирования.

Схематично работу алгоритма можно описать следующим образом:

– Извлечение нейтральной модели из набора данных. Эта модель зачастую представляет собой нейтральное выражение лица человека, который не отображает никаких эмоций. Такую модель всегда записывают в момент сканирования или съемки актерской игры.

– Подача нейтральной модели в качестве входа алгоритма. При отсутствии таковой, можно ее вычислить как математическое средние по всем вершинам всех моделей, которые идут на вход алгоритму. Так же на вход идут все остальные модели, из которых необходимо найти ключевые.

– Получение с помощью вычитания нейтральной модели из всех геометрий дельт смещения, которые при добавлении к нейтральной геометрии позволяют попасть обратно в каждую из геометрий.

– Формирование некого базиса, в который входит нейтральная модель и дельты геометрии, имеющие максимальную амплитуду. На основе проверок, зачастую, этой геометрии соответствует агрессивный крик человека.

– Описание всех оставшихся геометрий с помощью линейной комбинации сформированного базиса на основании метода наименьших квадратов, чтобы найти веса смешивания. Формула поиска весов выглядит следующим образом:

$$w = (d^t d)^{-1} d^T d_t,$$

где d – дельты смещения для геометрий базиса, d_t – дельты смещения для целевой геометрии.

– Вычисление ошибки реконструкции, как суммы евклидовых норм от разницы всех точек геометрий

– Добавление дельт геометрии с самой большой ошибкой к нынешнему базису.

Данная процедура повторяется, пока не сформируется базис нужного размера, или не будет достигнут нужный уровень точности.

3. Результаты

Данный алгоритм позволяет без больших вычислительных затрат осуществить поиск наиболее значимых кадров из набора.

Для проверки была проведена сортировка набора ключевых кадров с сессии сканирования по степени важности кадров. При сортировке учитывалась вся геометрия головы.

Визуальные результаты отображены на рисунке. Точность реконструкции всей головы для оставшихся вне базиса геометрий при размере базиса в 10 геометрий составляет 88.7831%.

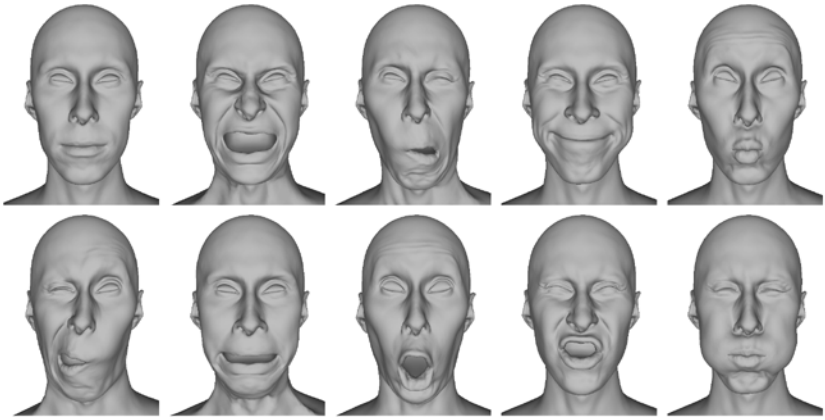


Рисунок. Внешний вид геометрий эмоций в порядке важности по результатам работы алгоритма

Также были проведены тесты с целью сортировки геометрий по значимости только в определенных областях, например, переносицы. Результаты оказались удовлетворительными. Точность реконструкции только на этом регионе на тех же данных составил 95.3032%

4. Применение

Рассмотрим теперь примеры применения данного алгоритма с целью упрощения и улучшения работы других алгоритмов.

Упомянутые ранее задачи реконструкции лица по изображению или скану могут требовать сохранения изначальной семантики базиса ключевых кадров, которые линейно смешиваются в процессе решения. В этих случаях не получится снизить размерность задачи с помощью метода главных компонент, что приведет к увеличению сложности решения. В таком случае можно применить данный подход для поиска наиболее значимого набора кадров. Однако, некоторая часть алгоритма реконструкции разбивает человеческое лицо на регионы, чтобы добавить гибкости модели деформации. При таком варианте сохраняется возможность отбирать наиболее важные кадры из набора для каждого региона в отдельности.

Зачастую имеется продолжительная анимация, в которой необходимо выбрать самые важные кадры. Эту работу делает трехмерный художник, основываясь исключительно на визуальном восприятии. В большинстве случаев он прекрасно с этим справляется, но это занимает значительное время. Представленный же подход позволяет делать это в автоматическом режиме. После проведенных тестирований было установлено, что большую часть кадров алгоритм выбирает так, как это сделал бы профессиональный художник.

Заключение

Таким образом, был разработан алгоритм, позволяющий находить наиболее значимые геометрии человеческого лица из набора с целью их дальнейшего использования в других алгоритмах. Он не требует сложных вычислений и позволяет производить поиск кадров для разных классов задач в автоматическом режиме.

Список литературы

1. Jolliffe, I. T. Principal Component Analysis / I. T Jolliffe // Springer Series in Statistics. – New York: Springer-Verlag, 2002. – 488 p.

2. Luan Tran. Nonlinear 3D Face Morphable Model / Luan Tran, Xiaoming Liu // CVPR. – N.Y, 2018. – 7346-7355

3. Heseltine, T. Face Recognition: A Comparison of Appearance-Based Approaches / T. Heseltine [et al.] // NCSIP. – Sydney, 2003. – P. 59-68.

Стерео ограничения для задачи не ригидной регистрации лица

А. Д. Борисов

Студент магистратуры

Д. И. Соломатин

Старший преподаватель

Введение

В наше время очень важной частью компьютерной графики является создание цифровых дублеров для кино и игр. Основной частью этого процесса является задача реконструкции человеческого лица по изображениям. Однако основной движущей силой решения в такого рода задачах является либо оптический поток, либо дифференцируемый рендер. В обоих случая на геометрию не накладываются какие-либо дополнительные ограничения по глубине.

Данная статья будет рассматривать эту проблему в контексте регистрации человеческого лица по стереокамерам.

1. Существующие решения

При наличии нескольких калиброванных камер, очевидным решением этой проблемы является получения трехмерного скана из изображений с целью последующей подгонки деформируемой геометрии под этот скан. Однако опыты и многочисленные наблюдения показывают, что скан человеческого лица, полученный по стереокамерам, не отличается высоким уровнем качества. Зачастую, получение скана лица осуществляется теми же алгоритмами, что и получения сканов других объектов. Алгоритм не знает ничего о том, скан какого объекта он пытается построить. Так как используется ограниченное количество камер, появляется большое количество артефактов. К тому же, расчет полноценного скана с нуля вычислительно затратен. Рассматриваемый подход предлагает меньшее количество вычислений, а также учитывание статистической модели конкретного человеческого лица.

Другой вариант получения глубинной информации был предложен Коки Нагано в его исследовательской работе [1]. В этой работе происходит поиск оптического потока между двумя ближайшими

камерами с помощью ряда штрафных функций, отсекающих неправильные соответствия. По полученным соответствиям строится триангуляция в трехмерном пространстве для каждого пикселя изображения. Затем ищется для каждой вершины геометрии такое смещение, чтобы вершина лежала на соответствующей триангулированной точке. В отличии от подхода с вычислением трехмерного скана, здесь может использоваться статистическая модель человеческого лица, а также не требуется расчет трехмерного скана. Однако у данного подхода есть существенные недостатки. Качество работы алгоритма очень сильно зависит от точности оптического потока, и если она недостаточна, а отсечения результатов не помогает убрать плохие соответствия, то нет никаких гарантий, что лучи, выпущенные из стерео-точек, пересекаются. А это условие – ключевое жесткое ограничение в построении сканов. Неудовлетворение этому условию влечет за собой искривление всего решения. Рассматриваемый в этой работе подход позволяет этого избежать, так как он использует только стерео-точки, полученные обратным путем – проецированием трехмерных точек деформируемой геометрии в пространства камер. Это гарантирует то, что лучи, выпущенные из стерео-точек, точно будут иметь одну точку пересечения.

2. Описание ограничения

Предполагается, что для реконструкции лица конкретного человека заранее были записаны и обработаны трехмерные сканы ключевых выражений лица.

В основе подхода может быть использована любая модель деформации. В данном случае воспользуемся для демонстрации линейной моделью [1], где происходит линейное смешивание дельт ключевых кадров с нейтральной эмоцией. После смешивания они прибавляются к нейтральной модели. В качестве основных невязок задачи оптимизации будет использоваться разница между проекциями заранее проставленных маркеров на геометрии и аналогичных маркеров, определенных на изображениях с камер. Решив задачу оптимизации получен результат отображенный на рис. 1.

Можно заметить, что финальное решение плохо попадает в эталонный результат. Чтобы улучшить качество решение с точки зрения попадания в эталонную геометрию, можно добавить информацию о глубине сцены.

Для решения этой задачи предложен следующий метод (он проиллюстрирован на рис. 2):

– Осуществляется трассировка лучами на нынешний вариант деформируемой геометрии через одну из двух используемых камер.

– Происходит отбор только тех точек геометрии, которые видны из первой камеры.

– Из полученного набора точек выбираются только те, которые одновременно наблюдаются и из второй камеры.

– Происходит проецирование отобранных точек в пространство обеих камер.

– Значения изображений в спроецированных точках, полученных с двух используемых камер, сравниваются между собой попиксельно.

Слева направо: целевая картинка, результат оптимизации по точкам, результат оптимизации по точкам со стерео ограничением

Рис. 1. Результаты оптимизации

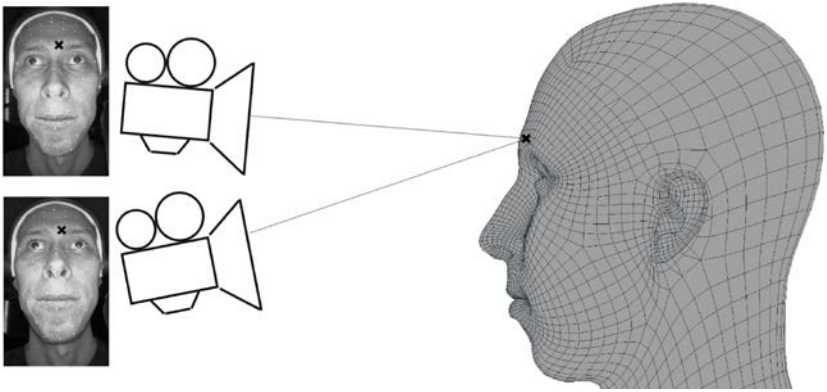


Рис. 2. Иллюстрация стерео ограничения

В результате решения задачи оптимизации геометрия будет пытаться деформироваться таким образом, чтобы каждая стерео-точка геометрии, при проецировании в одну из камер, попадала в пиксель, похожий на пиксель, полученный при проецировании стерео-точки в другую камеру. По своей сути, данная задача является аналогичной задаче, которая решается в момент построения трехмерного скана, только другими средствами. Главное достоинство данного подхода заключается в том, что в основе метода лежит модель деформации человеческого лица, имеющая априорную информацию о возможных изменениях геометрии лица конкретного человека, которая учитывается при решении.

Однако необходимо учитывать тот факт, что в рассматриваемой задаче происходит попиксельное сравнение значений двух изображений в стерео-точках, спроецированных в пространства этих изображений через камеры. Поэтому, как и в случае задачи построения трехмерного скана, которая решает схожую проблему, лучше всего применять подход, основанный на использовании пирамид изображений, где оптимизация начинает решение задачи с небольшого размытого изображения и постепенно его увеличивает до исходного [3]. Пример пирамид изображений отображен на рисунке (рис. 3)

Рис. 3. Пример набора пирамид изображений

3. Результаты

Данное ограничение накладывает дополнительные вычислительные сложности на решение, однако, при тщательной реализации, способно значительно улучшить решение без сильных просадок по производительности.

Было проведено тестирование на нескольких больших секвенциях, чтобы выяснить влияние этого ограничения на общее решение.

Данное ограничение очень сильно улучшает финальное качество решения по сравнению с тривиальным подходом (рис. 1), основанным на реконструкции геометрии исключительно по информации о маркерах на человеческом лице.

В данной статье было акцентировано внимание на данных, полученных с двух камер, однако, предлагаемое решение способно масштабироваться для большего количество камер.

К примеру, оно может быть использовано в стандартном подходе по обработке трехмерных сканов с фотограмметрических установок для того, чтобы исключить необходимость построения трехмерных сканов и осуществлять обработку изначально на оригинальных изображениях с камер.

Заключение

Таким образом, было реализовано стерео-ограничение, позволяющее получать информацию о глубине для более точного реконструирования человеческого лица.

Оно может быть использовано как для задач реконструкции на данных с двух камер шлема, так и на данных, полученных с фотограмметрических установок.

Список литературы

1. Nagano, K. Multi-View Stereo on Consistent Face Topology / K. Nagano, G. Fyffe // ACM SIGGRAPH computer graphics. – N.Y.: ACM, 2017 – P. 1-2.

2. Lewis, J. P. Practice and Theory of Blendshape Facial Models / J.P. Lewis, Ken Anjyo // ACM SIGGRAPH computer graphics. – N.Y.: ACM, 2014 – P. 1-2.

3. Andelson, E.H. Pyramid methods in image processing / E.H. Andelson, C.H. Anderson // ACM SIGGRAPH computer graphics. – N.Y.: ACM, 1983 – P. 1-2.

Модернизация сетевой и серверной инфраструктуры университета в новых условиях

К. О. Верещагин

Студент магистратуры

А. П. Толстобров

Доцент

1. Введение

В предыдущие годы в Воронежском государственном университете была создана развитая сетевая и серверная инфраструктура, удовлетворительно обеспечивающая поддержку университетской информационной системы управления образовательным процессом и контингентом студентов. Внедрение в образовательный процесс электронного обучения и дистанционных образовательных технологий обеспечивалось образовательным порталом «Электронный университет ВГУ» (https://edu.vsu.ru), созданным для этих целей на базе популярной LMS-системы Moodle.

Ситуация принципиально изменилась два года назад в связи с началом пандемии и возникшей необходимостью тотального перевода учебного процесса на дистанционный формат. Наличие образовательного портала и многолетний опыт работы в его электронной среде, обеспечивало необходимый для этого функционал. При многократном увеличении количества активных пользователей системы и интенсивности их работы в электронной среде из-за нагрузки на порядки возросшей, университет столкнулся с тем, прежняя сетевая и серверная инфраструктура оказалась не в состоянии обеспечивать бесперебойную эффективную работу преподавателей и студентов в электронной среде. В частности, обнаружилось, что при проведении лекций в онлайн-режиме пиковое количество пользователей, участвующих в видеоконференциях, выросло более, чем в 5-6 раз, а на сам электронный портал нагрузка выросла в 11 раз (см. рис. 1 и 2) комплекс ВКС BigBlueButton перестал справляться с такой нагрузкой.

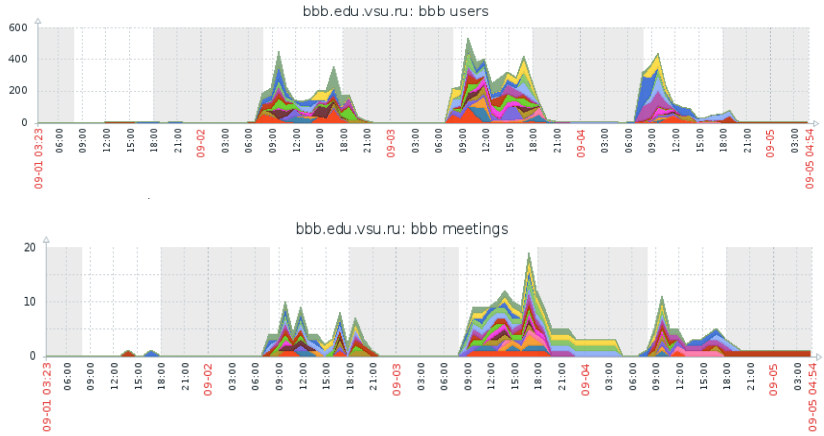


Рис. 1. Нагрузка и количество видеоконференций до пандемии

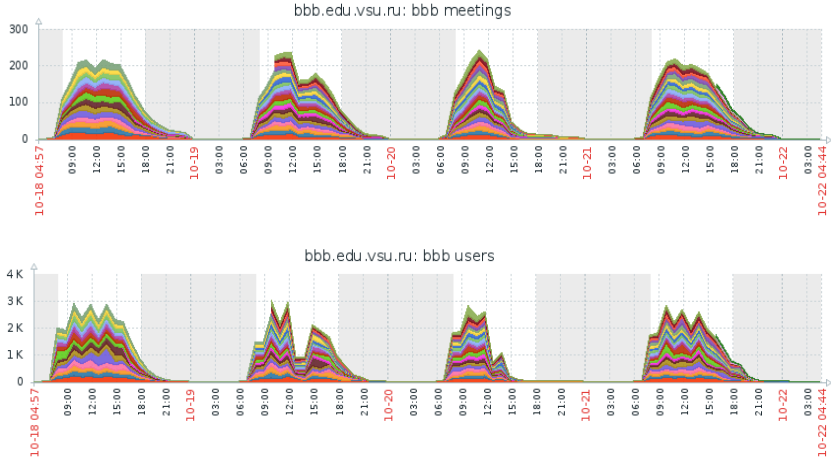


Рис. 2. Нагрузка и количество видеоконференций после начала пандемии

Из-за неготовности системы к такому объему единовременных пользователей и кратно возросшему трафику возникли частые сбои в работе системы, приводящие к усложнению, а зачастую невозможности проведения занятий для преподавателей и полноценной работе студентов в электронной образовательной среде.

2. Формулировка задачи

Проблемы с оборудованием, возникшие из-за возросшей нагрузки.

– Проблемы с проведением занятий в среде ВКС BigBlueButton из-за неспособности сетевых карт серверов обработать множественные пакеты, генерируемые видеоконференциями.

– Из-за огромного количества пакетов происходит засорение сетевого канала и как следствие нарушение связи между элементами кластера.

– Наличие уязвимостей, вызывающих сбои в работе портала из-за быстрого расширения ресурсов системы Moodle для работы на дистанционный формат обучения.

– Появились множество проблемных мест в плагинах самого Moodle, которые до этого оставались не замеченными.

Для решения этих разнородных проблем целесообразно разделили весь объем работ на отдельные подзадачи, а именно:

1. Аппаратное изменение сети и замена физического оборудования.
2. Изменений настроек и конфигурации сети, для увеличения ее пропускной способности и защиты от различных атак.
3. Перестройка виртуального кластера, который, как оказалось, из-за быстрого расширения был неверно сконфигурирован и настроен.

3. Аппаратное изменение сети

Среда видеоконференций и электронного портала LMS Moodle находится в кластере виртуализации oVirt, виртуальные машины располагаются на шести физических серверах, место для данных (БД, системные диски, общие шары) берется с СХД (Система хранения данных) Raidix.

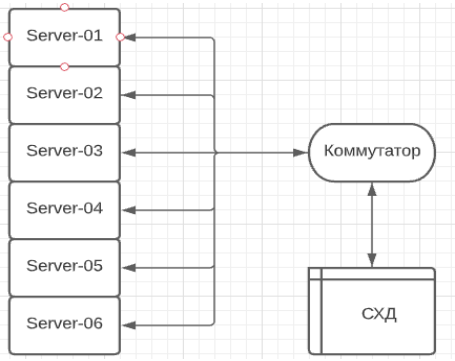


Рис. 3. Упрощённая аппаратная схема LMS Moodle и BBB

В шести физических серверах были установлены сетевые карты Intel X550 10Gb с 2 портами RJ45 формата PCIe 3.0 x4.

На каждом из шести физических сервере содержится по четыре BM с BBB. Первоначальные сетевые карты были не способны обрабатывать больше 60 тысяч пакетов, при том, что BM BBB и элементов кластера Moodle генерируют свыше 75 тысяч пакетов на один физический сервер. В результате получалась очередь из пакетов, которая замедляляла работу BBB и других участников сети.

Было принято решение обновить сетевые карты и коммутатор на Mellanox ConnectX-5 PCIe 3.0 x16 SFP+ подключаемые по оптическим кабелям, что увеличило количество обрабатываемых сервером пакетов и уменьшило системные прерывания.

4. Изменение конфигурации сети

Помимо обновления сетевых карт, также были изменены сетевые настройки BM, сервера которых были подключены по 10Gb.

Файл /etc/sysctl.conf:

```
kernel.core_uses_pid = 1
fs.suid_dumpable = 1
vm.swappiness = 2
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.ipv4.tcp_rmem = 4096 131072 16777216
net.ipv4.tcp_wmem = 4096 16384 16777216
net.core.netdev_max_backlog = 30000
net.core.somaxconn = 65535
net.ipv4.conf.default.rp_filter=1
net.ipv4.conf.all.rp_filter=1
net.ipv4.conf.all.accept_redirects = 0
net.ipv6.conf.all.accept_redirects = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv6.conf.all.accept_source_route = 0
```

Из важных параметров стоит отметить, следующие:

– kernel.core_uses_pid и fs.suid_dumpable с заданными параметрами выключает дамп ядра, чтобы не создавать копию памяти, если программа работает с suid;

– vm.swappiness – говорит системе о том, сколько % должно быть свободной памяти, чтобы начинался свопинг – механизм виртуальной памяти, при котором часть данных из ОЗУ перемещается на на HDD/SSD;

– net.core.rmem_max – максимальный размер буфера приема данных для всех соединений;

– net.core.wmem_max – максимальный размер буфера передачи данных для всех соединений;

– net.ipv4.tcp_rmem и net.ipv4.tcp_wmem – векторная переменная в файле tcp_rmem/tcp_wmem (минимум/по умолчанию/ максимум) содержит три целых числа, определяющих размер приемного буфера сокетов TCP.

Минимум - каждый сокет TCP имеет право использовать эту память по факту своего создания. Размер минимального буфера по умолчанию составляет 4 Кбайт (4096).

По умолчанию – количество памяти, допустимое для буфера приема/передачи сокета TCP по умолчанию. Значение используемого по умолчанию буфера составляет 87830 байт/16384 байт.

Максимум – максимальный размер буфера, который может быть автоматически выделен для приема/передачи сокету TCP. При «статическом» выделении памяти с помощью SO_RCVBUF/SO_SNDBUF этот параметр не имеет значения.

– net.core.netdev_max_backlog – параметр определяет максимальное количество пакетов в очереди на обработку, если интерфейс получает пакеты быстрее, чем ядро может их обработать.

– net.core.somaxconn – максимальное число открытых сокетов, ждущих соединения.

Оставшиеся параметры помогают защититься от IP-спуфинга и запрещают маршрутизацию от источника.

Таким образом была произведена конфигурация настроек сети, которая способствовала улучшению ее производительности и снижению простоев пакетов, что привело к значительному улучшению в работе как самого Moodle, так и сервису видеоконференций BBB.

5. Изменение виртуального кластера

В связи с внезапным наступлением дистанционного обучения, LMS Moodle была быстро расширена “на горячую”, что привело к многим проблемам.

На приведенной на рис. 4 схеме можно сразу отметить несколько уязвимых мест.

2. Все запросы обрабатываются только одной ВМ, без обеспечения какой-либо отказоустойчивости, то есть выход её из строя означает полный отказ кластера.

3. Каждая ВМ для обработки скриптов общается со своей базой данных, то есть если база moodle-db-01, перестает работать, мы так же теряем и moodle-web-01, что ведет к значительному увеличению нагрузки на оставшиеся web сервера, а 01 будет простаивать.

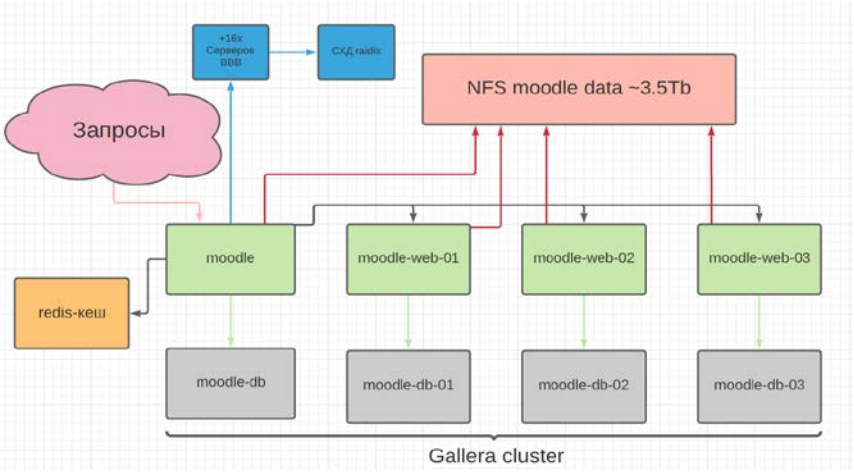


Рис. 4. Прежняя схема кластера LMS Moodle

4. Отсутствие балансировщика нагрузки для баз данных, приводило к тому, что некоторые БД работали на 10-15% выделенных мощностей, в то время как одна из БД могла «захлебываться», тратя все выделенные ей ресурсы.

5. Помимо обработки запросов, главный сервер отвечал еще и за общение с серверами кластера видеоконференций BBB. Это приводило к его полной перегрузке, «торможению» и простоям в работе всего учебного процесса.

Для исправления всех уязвимых мест пришлось переделать виртуальный кластер LMS Moodle (Рис. 5).

В кластер были добавлены 2 сервера балансировки запросов, работающие в связке для обеспечения отказоустойчивости системы, а также 2 балансировщика запросов к базам данных. Это позволяло планомерно распределить нагрузку по всем 5 серверам БД. Была устранена проблема, когда веб-серверы общались каждый со своей базой, что зачастую приводило к зависаниям в работе портала, с основного сервера была снята значительная часть нагрузки.

Заключение

Зачастую, в силу больших вычислительных мощностей и сжатых сроков реализации задач, не всегда задумываются над правильной конфигурацией и настройкой кластеров, сетей, виртуальных машин и т.д. В данной работе мы показали какие погрешности и излишние нагрузки может создавать не точная настройка.

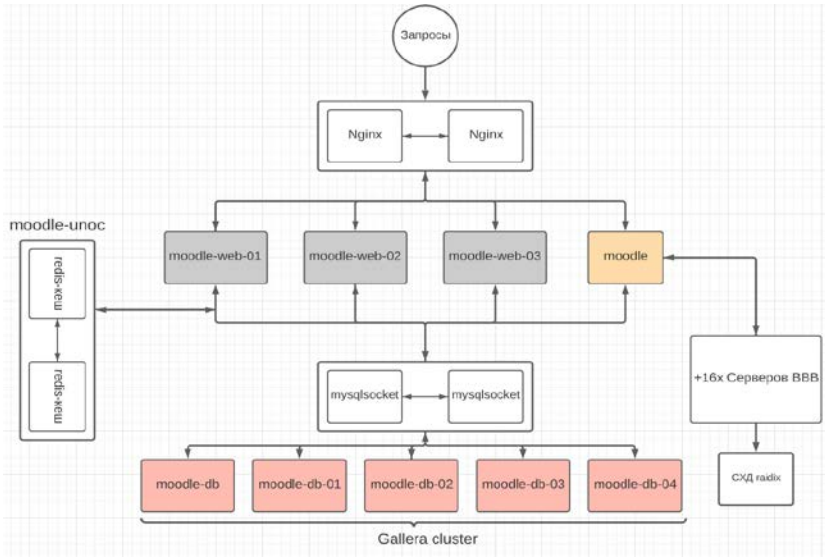


Рис. 5. Новая схема кластера LMS Moodle

Благодаря настройке сети на уровне ядра и смене оборудования удалось добиться значительного прироста пропускной способности сети. Изменения в схеме виртуального кластера уменьшило его нагрузку и количество сбоев системы, что привело к уменьшению времени его простоев при увеличении числа активно работающих с порталом пользователей (рис. 6-7).

Выяснилось, что три тысячи человек в пике — это еще не конечная нагрузка, а в различные дни в пиках в системе сидят свыше четырех тысяч пользователей, только в одних видеоконференциях BBB, и более 20 тысяч человек работают с самим порталом Moodle, также видим увеличение входящего трафика после изменений сетевых конфигураций.

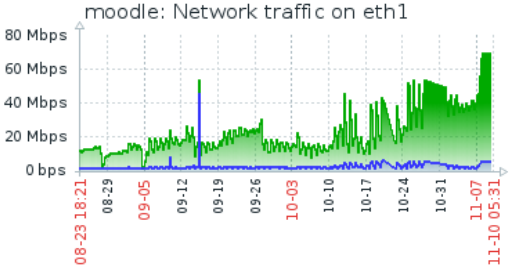


Рис. 6. Изменения трафика

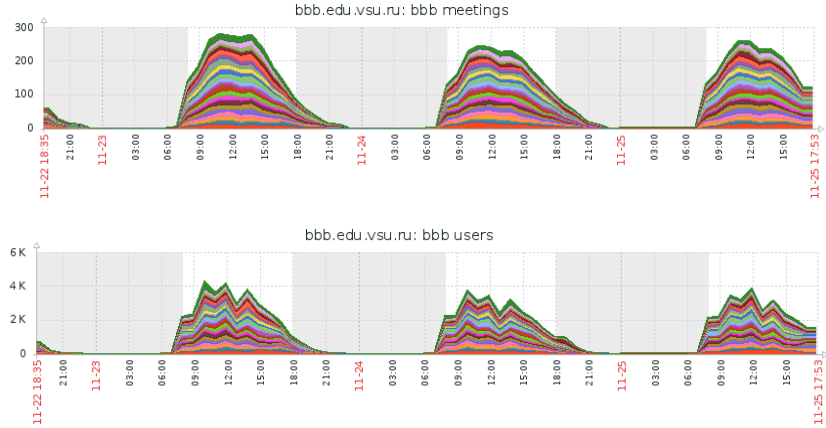


Рис. 7. Количество пользователей и видеоконференций после изменений

Хотя это оказалось не единственным уязвимым местом в производительности высоконагруженных систем. В дальнейшем целесообразно провести большую работу над оптимизацией кода самих плагинов LMS Moodle, по созданию выделенных серверов под задачи, которые сильно нагружают сам портал (например, плагин для Latex), так же необходимо грамотно подстраивать расписание под дистанционное обучение, потому что на всех графиках видно, что очень большое количество пользователей использует видеоконференции в промежуток с 9:00 до 14:00, после чего нагрузка резко спадает.

Список литературы

1. Evolution of Software-Defined Networking [Электронный ресурс] : – Режим доступа : https://advance.biz-tech-insights.com/whitepaper/CS-0052_Evol_SDN_Dummies_fin.pdf

2. Software-defined networking [Электронный ресурс] : – Режим доступа : https://en.wikipedia.org/wiki/Software-defined_networking

3. Шматко, А. Современные подходы построения SDN сетей А. Шматко . – LAP, 2017. – 84 с.

4. Nadeau, T. D. SDN: Software Defined Networks: an authoritative review of network programmability technologies / T. D. Nadeau, K. Gray. –" O'Reilly Media, Inc.", 2014. – 384 с.

5. Олифер, В. Г. Компьютерные сети. Принципы, технологии, протоколы : учебник для вузов / В. Г. Олифер, Н.А. Олифер. – Юбилейное издание. – СПб. : Питер, 2020. – 1008 с.

6. Куроуз, Д. Компьютерные сети: Нисходящий подход / Д. Куроуз, К. Росс. – 6-е изд. – М. : «Э», 2016. – 912 с.
7. Настройка Linux для высоконагруженных проектов и защиты от DDoS [Электронный ресурс] : – Режим доступа : https://romantelychko.com/blog/1300/

Анализ поведения клиентов в сфере B2C на основе технологии Data Mining

М. С. Верещагина

Студент бакалавр

В. В. Гаршина

Доцент

Введение

Большинство крупных корпоративных компаний, вынужденных в ходе своей работы обрабатывать большие объемы клиентских данных, обращаются к технологиям машинного обучения и анализу больших данных. При таких объемах разнородной (и далеко не всегда «чистой») информации компании не всегда имеют возможность следить за ситуацией на рынке и вовремя реагировать на негативные события без использования дополнительных средств анализа. Так, один из самых популярных примеров использования технологий Data Mining в реальной жизни – это прогнозирование оттока клиентов [1].

Задача прогнозирования оттока клиентов (churn prediction) заключается в том, чтобы заранее найти и выявить категорию пользователей, склонных через определенный промежуток времени отказаться от использования услуг и продуктов компании. Точное и заблаговременное обнаружение таких клиентов позволяет более эффективно бороться с их оттоком – например, выявлять причины их ухода и принимать соответствующие меры по их удержанию. Эта задача особенно актуальна для организаций, работающих в сегменте B2C («business to consumer» или «бизнес — потребителю»), а в особенности для компаний, работающих в областях, где распространение предлогаемых компанией услуг близко к 100%. В такой ситуации задача удержания клиентов в компании и налаживания с ними взаимоотношений выходит на первый план, поэтому в данной работе будет рассмотрена методика прогнозирования оттока клиентов в B2C сегменте.

1. Обзор предметной области

Приступая к составлению плана исследования, необходимо провести анализ предметной области.

Массовый отток клиентов из компании оказывает сильное влияние на компании с транзакционной моделью бизнеса, т.е. подразумевающих регулярные платежи от клиентов, где доля распространения услуг и продуктов уже близка к полному насыщению. К ним относятся банки, телеком-операторы, интернет-провайдеры, интернет-магазины и т.д.

B2C сегмент рынка отличается от B2B (модель «business to business» или «бизнес — бизнесу») аудиторией продукта — конечным клиентом. В B2C потребителем выступают обычные люди, т.е. физические лица, которые представляют собой нестабильную аудиторию: клиентская база может вырасти в 10 раз за один сезон и практически сразу же уменьшиться после окончания периода скидок. В B2B сегменте клиентами выступают другие компании, т.е. юридические лица – они гораздо стабильнее, изменение их объема на рынке замедленно, поэтому переходы одного крупного клиента к другому происходят реже. Именно поэтому наиболее важной сферой для прогнозирования оттока клиентов является B2C сегмент рынка.

Чем обусловлена выгода повышения лояльности существующих клиентов?

1. Много мелких клиентов обеспечивают более высокую стабильность компании и прогнозируемость получаемой прибыли, так как уход одного из клиентов не является критичным для бизнеса, в отличие от компаний с малым количеством крупных клиентов. Именно поэтому многие компании пытаются совмещать работу на два сектора: B2C и B2B.

2. Большая клиентская база и стабильная работа повышают статус и имидж компании, что дает определенное маркетинговое преимущество для привлечения новых клиентов - это приводит к возможности позволить себе снизить бюджет на рекламу.

Именно в силу этих причин компании часто внедряют CRM-системы в свой рабочий процесс для управления взаимоотношениям с клиентами, одной из задач которых является повышение лояльности существующих и возврат ушедших клиентов. Для этого компании разрабатывают всевозможные методы, одни из них - специальные формы договоров (с наличием штрафов за досрочное расторжение) и особые программы лояльности.

Показательный пример такой области – рынок телекоммуникационных услуг, где насыщение клиентов товарами и услугами уже произошло. Как следствие – происходит постепенное снижение прироста клиентской базы и ее перераспределение между существующими операторами, что повышает значимость

существующих клиентов. В таких условиях задача удержания и налаживания отношений с клиентом выходит на первый план.

Таким образом, анализ поведения абонентов в период предшествующий отказу от услуг компании позволяет выявить типичные для них признаки и среди общей массы пользователей выделить тех, кто с наибольшей вероятностью перестанут пользоваться услугами компании в ближайшее время с целью предложить им в рамках программы лояльности персонализированные тарифные планы и специальные предложения, чтобы предотвратить их уход. [1]

2. Алгоритмы обработки данных Data Mining

Существует множество различных подходов анализа оттока клиентов. Большая часть этих подходов основана на интеллектуальном анализе данных [2], который демонстрирует высокую эффективность и производительность в современном мире. Если речь идет о традиционных – статистических – методах анализа данных или оперативной аналитической обработке данных (OLAP), то они в большинстве своем ориентированы на проверку заранее сформулированных гипотез. Методы Data Mining, напротив, ставят своей целью нахождение неочевидных закономерностей, которые ранее были скрыты и позволяют самостоятельно стоить гипотезы о взаимосвязях. Если речь идет о клиентских данных, подразумевающих большое количество переменных, то в работе анализа самой сложной частью является именно формулировка гипотезы относительно их зависимостей, поэтому очевидно, что в данной задаче предпочтение необходимо отдать именно методам Data Mining.

На рис. 1 показаны различные алгоритмы обработки данных, которые используются для нахождения и извлечениях скрытых закономерностей из информации.

Первые две группы «Трансформация» и «Предобработка» показывают набор алгоритмов для первоначальной обработки исходных данных для их дальнейшего использования в алгоритмах Data Mining. Применяются такие методы, как слияние нескольких таблиц в одну, заполнение пропусков, сэмплинг, внутреннее обогащение данных и другие. К примеру, для большинства компаний величина CLV (Customer Lifetime Value) является непрерывным атрибутом, который надо дискретизировать, чтобы получить категориальный атрибут для дальнейшей классификации.

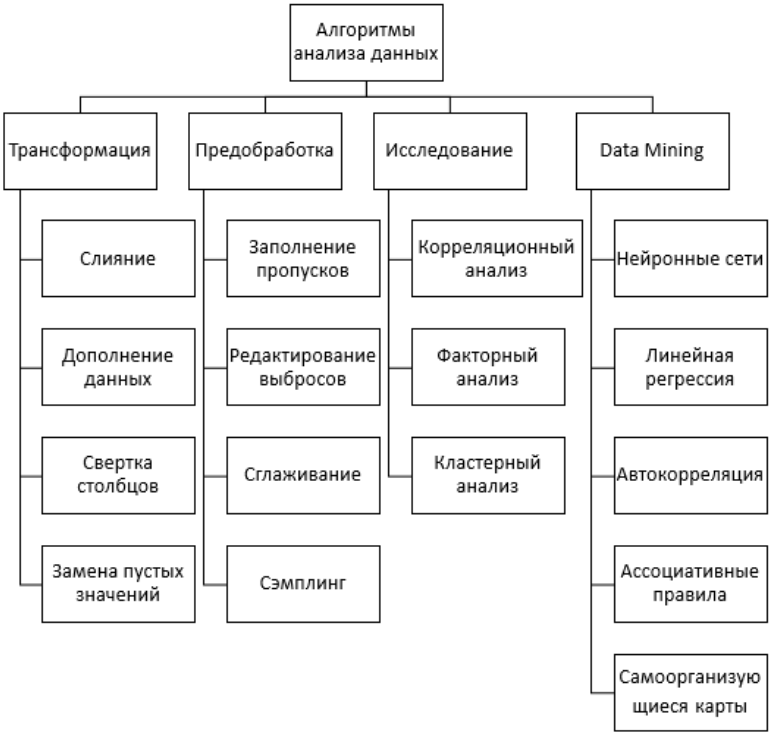


Рис. 1. Классификация алгоритмов анализа данных

Само же исследование будет проводиться с помощью факторного и кластерного анализа, благодаря которым можно оценить и визуализировать структуру и статистические характеристики данных для проведения разведочного и описательного анализа. Далее будет проведена сравнительная характеристика различных методов Data Mining для определения конечных проведения исследования.

3. План исследования

В результате вышеописанного анализа предметной области и подходов к анализу оттока клиентов был составлен план исследования для разработки сценария проведения бизнес-анализа модели поведения расторгнувших контракты для задачи управления отношениями с клиентами телекоммуникационной компании.

Для достижения поставленной цели предполагается решение следующих задач:

1. Определить признаки классификации поведения абонентов.
2. Сформировать дата сеты из истории взаимодействия компании с клиентами. Для оценки вероятности отключения того или иного абонента компании используют большое количество метрик: количество совершенных звонков в месяц, пользовательские подписки, давность последней покупки и многие другие. Поэтому мы хотим проклассифицировать отключившихся за определенный период абонентов по ряду признаков - это будет происходить в в рамках итерационного процесса с корректировкой классифицирующих признаков в зависимости от полученных результатов на последующих шагах.
3. Выбрать набор алгоритмов Data Mining для решения поставленных задач.
4. Разработать сценарии для Системы Анализа Данных для выявления клиентов “в группе риска” (т.е. тех клиентов, которые в ближайшем будущем способны покинуть компанию) и наиболее влиятельных классифицирующих признаков и их значений, наибольшим образом способствующих оттоку абонентов разных групп.
5. На основе полученных результатов сформировать предложения для удержания клиентов (в первую очередь наиболее значимых для компании групп).

4. Система анализа данных

В качество основного инструмента, использующего методы DataMining, после обзора рынка инструментов было решено выбрать аналитическую платформу Loginom [3]. Данное решение отчасти обусловлено доступностью бесплатной полнофункциональной версии программы.

Loginom — интегрированная Low-code платформа для реализации всех аналитических процессов: от подготовки данных до моделирования и визуализации результатов. Эта платформа позволяет пользователям самостоятельно создавать сценарий анализа данных из уже готовых компонентов - при этом компоненты такие компоненты обработки способны реализовывать как простейшие операции сортировки и вычисления, так и разнообразные методы Data Mining, включая использование нейросетей. Сценарий анализа данных настраивается визуально, аналитику, поэтому пользователь видит, как обрабатываются данные на любом этапе, что упрощает поиск ошибок в расчётах. [3]

Разработанные сценарии анализа можно многократно использовать в разных проектах, превратив их в компоненты и поделившись с другими пользователями.

Соответственно, существует 2 метода разработки сценариев, которые продемонстрированы на рис. 2.

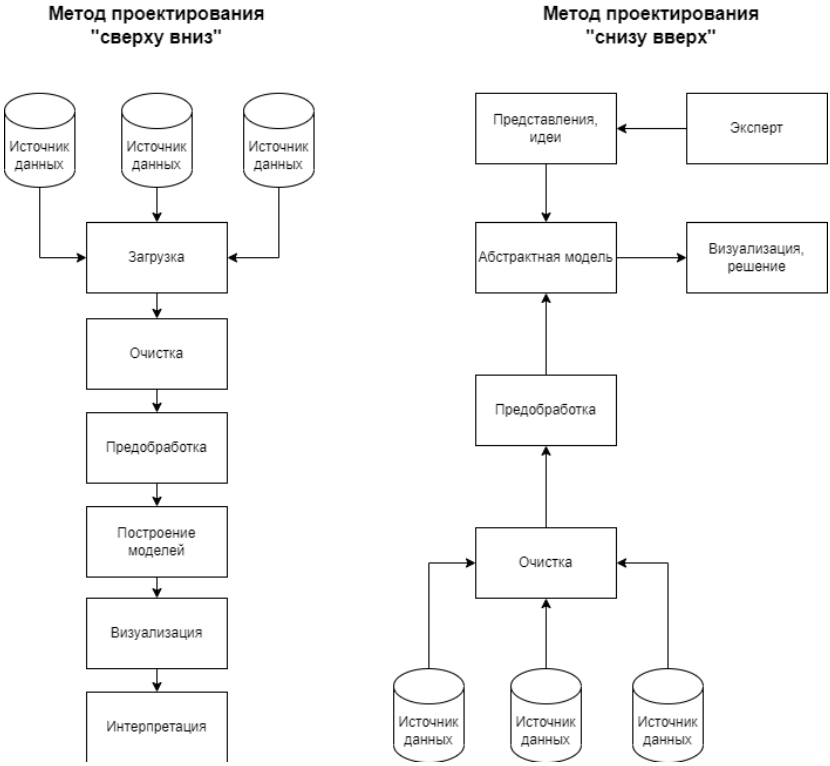


Рис. 2. Методы проектирования сценариев в Loginom

При методе проектирования “снизу вверх” любой сценарий анализа всегда будет начинаться с загрузки источников данных, где на каждом следующем уровне сценария будет происходить абстрагирование от данных путем применения к результатам предыдущего уровня тех или иных компонент. При этом оказывается, что на верхних уровнях сценария мы оперируем в основном условиями задачи, почти абстрагировавшись от исходных данных. Такой подход, ориентированный на единичные задачи и привязанный к данным, делает невозможным повторное использование сценария в аналогичных задачах.

При проектировании «сверху вниз», то есть в отсутствии исходных данных, сначала строятся максимально абстрагированные от данных сценарии, а разработка производится по нисходящей – к всё более конкретным действиям. Пройдя весь путь разработки, в конце ясно, какой тип данных потребуется и в каком виде они должны быть представлены.

В данной работе мы будем стремиться к реализации подхода «сверху вниз», чтобы получить максимально обобщенный сценарий, готовый к внедрению в разные организации.

Заключение

Данная статья посвящена разработке методики проведения анализа клиентских данных в среде Loginom для предотвращения оттока клиентов. Разработанный сценарий после тестирования на реальных обезличенных данных от одной из компаний, предоставляющей услуги в данном сегменте рынка, затем можно будет многократно применять к однотипным данным и использовать для анализа поведения и прогнозирования оттока клиентов в B2C сфере.

Список литературы

1. Кудинов, А. Т. CRM: российская практика эффективного бизнеса / А. Т. Кудинов, Е. Голышева, М. Сорокин. – М. : Издательство 1С-Паблишинг, 2009. – 374 с.

2. Методы и модели анализа данных: OLAP и Data Mining / А. А. Барсегян [и др.]. – СанктПетербург : БХВ Петербург, 2012. – 335 с.

3. Loginom – Руководство пользователя [Электронный ресурс]. – Режим доступа: https://help.loginom.ru/userguide/

Новый пример однородной поверхности в $\mathbf{R}^4$

М. С. Верещагина

Студент магистратуры

А. В. Лобода

Профессор

Введение

В данной работе обсуждается процесс построения и интегрирования 4-мерных представлений 3-мерных алгебр Ли. Такая задача связана с описанием однородных поверхностей, каждая из которых восстанавливается по соответствующей ей алгебре Ли. В относительно простом случае однородности относительно линейных преобразований 4-мерного вещественного пространства мы рассматриваем 3-мерные алгебры Ли, состоящие из матриц 4-го порядка. Задача описания вещественных гиперповерхностей пространства R^4 в своей неизученной части [1] сводится именно к таким рассмотрениям.

Постановка задачи

3-мерная (вещественная) алгебра Ли – это линейное 3-мерное пространство g с дополнительной билинейной антикоммутативной операцией, называемой коммутатором (или «скобкой»): если $A, B, C \in g$, то

$$[A, B] = [B, A] \in g,$$

$$[\alpha A + \beta B, C] = \alpha [A, C] + \beta [B, C],$$

$$[[A, B], C] + [[B, C], A] + [[C, A], B] = 0.$$

Всего 3-мерных алгебр Ли имеется 9 типов [2]. В данной работе будет рассмотрена разложимая алгебра. Ее описание в некотором базисе e_1, e_2, e_3 дается следующими коммутационными соотношениями:

$$g2 \oplus g1 : [e_1, e_2] = e_1.$$

Коммутаторы $[e_1, e_3]$ и $[e_2, e_3]$ полагаются равными 0.

Необходимо построить примеры матричной реализации этой алгебры в пространстве квадратных матриц 4-го порядка. Везде далее e_1, e_2, e_3 – это базисные (4×4) -матрицы искомой алгебры.

В качестве e_1 изначально берётся произвольный тип жордановой нормальной формы матрицы. Вообще говоря, вид жордановой формы связан с количеством различных (вещественных и комплексных) собственных значений такой матрицы, а также с наличием у нее присоединенных векторов. В данной работе были рассмотрены только случаи, имеющие в спектре лишь вещественные собственные значения.

По полученной реализации обсуждаемой алгебры Ли требуется выписать и решить систему уравнений в частных производных для получения интегральной поверхности алгебры векторных полей.

В силу необходимости большого количества технических (но точных!) вычислений при решении этой задачи используется пакет символьной математики MAPLE.

Выбор базиса

Из условия (4) известно, что e_3 одновременно коммутирует с e_1 и e_2:

$$e_1e_3 = e_3e_1, e_2e_3 = e_2e_1.$$

Известно [2], что алгебр с таким «двойным коммутированием» базисных элементов имеется достаточно много. Поэтому для упрощения вычислений примем e_3 равной единичной матрице. Отметим, что условие (4) накладывает ограничения и на матрицу e_1.

Предложение 1. Пусть e_1 имеет одну жорданову клетку с 4-кратным вещественным собственным значением и удовлетворяет соотношению (2). Тогда диагональ e_1 может быть только нулевой.

Доказательство. Пусть

$$e_1 = \begin{pmatrix} \lambda & 1 & 0 & 0 \\ 0 & \lambda & 1 & 0 \\ 0 & 0 & \lambda & 1 \\ 0 & 0 & 0 & \lambda \end{pmatrix}, e_2 = \begin{pmatrix} b_1 & b_2 & b_3 & b_4 \\ b_5 & b_6 & b_7 & b_8 \\ b_9 & b_{10} & b_{11} & b_{12} \\ b_{13} & b_{14} & b_{15} & b_{16} \end{pmatrix}.$$

Тогда $[e_1, e_2]$ будет равно

$$[e_1, e_2] = \begin{pmatrix} b_5 & b_6 - b_1 & b_7 - b_2 & b_8 - b_3 \\ b_9 & b_{10} - b_5 & b_{11} - b_6 & b_{12} - b_7 \\ b_{13} & b_{14} - b_9 & b_{15} - b_{10} & b_{16} - b_{11} \\ 0 & -b_{13} & -b_{14} & -b_{15} \end{pmatrix}.$$

В силу равенства (4) этот коммутатор должен равняться матрице e_1. Приравняем все диагональные элементы полученной матрицы (7) собственному значению λ исходной матрицы и решим полученную систему четырех уравнений относительно b_5, b_{10}, b_{15}, λ. Все эти величины, как легко видеть, оказываются нулевыми.

Несложно понять, что, аналогично, все собственные значения всех жордановых клеток, составляющих матрицу e_1, должны быть нулевыми.

За счет компьютерных экспериментов были получены несколько видов матричных базисов, в которых e_1 представлена в виде комбинаций жордановых клеток различных размеров с нулевыми собственными значениями. Приведем два примера базисов искомых представлений разложимой алгебры.

Пример 1. Матрица e_1 содержит две клетки 2-го порядка.

$$e_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, e_2 = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & a-1 & b \\ 0 & 0 & 0 & a \end{pmatrix}, e_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & l & 0 \\ 0 & 0 & 0 & l \end{pmatrix}.$$

где a, b, l – вещественные числа.

Пример 2. Матрица e_1 содержит две клетки 2-го порядка.

$$e_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, e_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 3 \end{pmatrix}, e_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Интегрирование алгебры из примера 2

Выпишем систему линейных дифференциальных однородных уравнений в частных производных первого порядка, соответствующую матричному базису (9):

$$\begin{cases} x_2\dfrac{\partial\Phi}{\partial x_1}+x_3\dfrac{\partial\Phi}{\partial x_2}=0, \\ x_1\dfrac{\partial\Phi}{\partial x_1}+2x_2\dfrac{\partial\Phi}{\partial x_2}+3x_3\dfrac{\partial\Phi}{\partial x_3}+(x_3+3x_4)\dfrac{\partial\Phi}{\partial x_4}=0, \\ x_1\dfrac{\partial\Phi}{\partial x_1}+x_2\dfrac{\partial\Phi}{\partial x_2}+x_3\dfrac{\partial\Phi}{\partial x_3}+x_4\dfrac{\partial\Phi}{\partial x_4}=0. \end{cases}$$

Искомым решением данной системы уравнений будет функция, определяющая некоторую гладкую поверхность, которая называется интегральной поверхностью исходной алгебры векторных полей [3]:

$$\Phi(x_1, x_2, x_3, x_4)=0.$$

Решать систему (10) предлагается следующим образом: выбрав одно из уравнений – решить его, найденное решение подставить в остальные уравнения системы и аналогичным образом действовать, пока не будет найдено решение последнего уравнения.

Запишем сначала уравнения характеристик для третьего уравнения системы (10):

$$\frac{dx_1}{x_1}=\frac{dx_2}{x_2}=\frac{dx_3}{x_3}=\frac{dx_4}{x_4}.$$

Все уравнения из (12) являются обыкновенными дифференциальными уравнениями с разделяющимися переменными [4]. Для всех уравнений получили:

$$\frac{x_1}{x_2}=C_1,\frac{x_2}{x_3}=C_2,\frac{x_3}{x_4}=C_3.$$

Выразим теперь частную производную по x_1 через новую функцию F:

$$\frac{\partial\Phi}{\partial x_1}=\frac{\partial F}{\partial t_1}\frac{\partial t_1}{\partial x_1}+\frac{\partial F}{\partial t_2}\frac{\partial t_2}{\partial x_1}+\frac{\partial F}{\partial t_3}\frac{\partial t_3}{\partial x_1}=\frac{\partial F}{x_2\partial t_1}.$$

Таким образом, получаем следующее соотношение:

$$\Phi(x_1, x_2, x_3, x_4)=F(t_1, t_2, t_3),$$

где $t_1(x_1, x_2, x_3, x_4)=\dfrac{x_1}{x_2}, t_2(x_1, x_2, x_3, x_4)=\dfrac{x_2}{x_3}, t_3(x_1, x_2, x_3, x_4)=\dfrac{x_3}{x_4}$.

Выразим остальные частные производные аналогичным способом, подставим в (10) и получим новую систему уравнений:

$$\begin{cases}(t_2-t_1)\dfrac{\partial F}{\partial t_1}+t_2\dfrac{\partial F}{\partial t_2}=0,\\ -t_1\dfrac{\partial F}{\partial t_1}+t_2\dfrac{\partial F}{\partial t_2}-t_3^2\dfrac{\partial F}{\partial t_3}=0.\end{cases}$$

Запишем уравнения характеристик для первого уравнения системы и найдем 2 первых интеграла системы:

$$\frac{dt_1}{t_2-t_1}=\frac{dt_2}{t_2}=\frac{dt_3}{0}.$$

Аналогично решаем (17) и находим:

$$F(t_1,t_2,t_3)=f(\xi_1,\xi_2),$$

где $\xi_1(t_1,t_2,t_3)=2t_1t_2-t_2^2=2\dfrac{x_1}{x_3}-\dfrac{x_2^2}{x_3^2}, \xi_2(t_1,t_2,t_3)=t_3=\dfrac{x_3}{x_4}$.

Выразим частные производные через новую функцию f и подставим их в систему (16):

$$-2\xi_1\frac{\partial f}{\partial \xi_1}-\xi_2^2\frac{\partial f}{\partial \xi_2}=0.$$

После решения (19) получаем:

$$\frac{\xi_2\ln(\xi_1)+2}{2\xi_2}=C.$$

Перейдем к исходным переменным x_1, x_2, x_3, x_4, используя соотношение (18), и получаем первый интеграл:

$$\frac{x_4}{x_3}+\frac{\ln(x_2^2-2x_1x_3)}{2}-\ln(x_3)=C.$$

На основании первого интеграла (21) можно записать в итоге общее решение системы (10).

$$\Phi(x_1,x_2,x_3,x_4)=F\_1(\frac{x_4}{x_3}+\frac{\ln(x_2^2-2x_1x_3)}{2}-\ln(x_3)).$$

Полученная функция $F\_1(\varphi)$ из (22) была подставлена в каждое из уравнений исходной системы (10) – все равенства выполняются, поэтому функция $F\_1(\varphi)$ действительно является решением системы (10).

Запишем теперь искомое уравнение поверхности, полученное

для рассмотренного частного случая:

$$x_4 = x_3(\ln(x_3) - \frac{1}{2}\ln(x_2^2 - 2x_1x_3)).$$

Заключение

В работе найдены 4-мерные представления разложимой 3-мерной алгебры Ли (2) и получена интегральная поверхность (23) для одного из этих представлений. Найденное уравнение представляет собой новый пример однородной поверхности в R^4, отсутствующий в [1].

Список литературы

1. Можей, Н. П. Однородные подмногообразия в четырехмерной аффинной и проективной геометрии / Н. П. Можей // Изв. вузов. Матем. – 2000. – № 7. – С. 41–52.

2. Мубаракзянов, Г. М. О разрешимых алгебрах Ли / Г. М. Мубаракзянов // Изв. вузов. Матем. – 1963. – № 1. – С. 114–123.

3. Лобода, А. В. О больших семействах аффинно однородных поверхностей в R^4 / А. В. Лобода, Б. М. Даринский // Материалы Международной конференции «Воронежская зимняя математическая школа С. Г. Крейна – 2022». – Воронеж, 2022. – С. 147–150.

4. Понтрягин, Л. С. Обыкновенные дифференциальные уравнения / Л. С. Понтрягин. – 4-е изд. – М. : Наука, 1974. – 331 с.

Тонкая настройка NER модели bert-mult

Т. С. Гаршин

Студент магистратуры

А. Ю. Иванков

Доцент

Введение

В технологиях TextMining одной из актуальных задач является извлечение фактов и данных из массивов неструктурированных текстовых документов. Решения этой задачи в настоящее время строятся на основе двух подходов:

– использования обученных нейросетей для поиска именованных сущностей в тексте без учета синтаксиста;

– применение алгоритмов разбора предложений (как нейросетевых, так и основанных на правилах), учитывающих морфологию и синтаксис.

При использовании нейросетевого подхода модель требует настройки на предметную область, с которой будет работать в дальнейшем. Настройка, представляет собой, дообучение модели для правильного распознавания специфических слов и терминов, используемые в анализируемых текстах. Одной из основных задач в рамках настройки модели является сбор и разметка датасета, на котором будет проводиться дообучение.

1. Процесс сбора и разметки датасета для дообучения модели

Необходимый для настройки модели датасет был собран вручную из текстов новостных статей. Объем размеченного датасета составил 10000 тегов. Для реализации задачи распознавания именованных сущностей было выделено 27 классов распознавания (рис.1). Тегирование производилось в системе разметки bio (b-голова, i-тело, o-не значимый тег) [1].

В рамках нашей задачи мы настраивали модель на излечение фактов из новостных статей, связанных с заключением контрактов между организациями и людьми. Это означает, что наиболее важным при выделении фактов для нас было найти теги:

– Персона (B-PER);

– Организация (B-ORG);
– Продукт (B-PRODUCT);
– Страна (B-GPE);
– Денеги (B-MANEY);
– Контракт (B-LAW).

	Sentence#	Word	Tag
10405	449	продвижению	O
10406	449	этих	O
10407	449	продаж	B-MONEY
10408	449	утверждает	O
10409	449	аналитик	B-PER
10410	449	ГК	B-ORG
10411	449	'Финам'	I-ORG
10412	449	Леонид	B-PER
10413	449	Делицын	I-PER
10414	449	Спортивная'	B-ORG

Рис. 1. Датасет, подготовленный для обучения

В типичной структуре предложения двумя сторонами контракта могут быть организации, люди, страны. Фактом контракта должен быть соответствующий тег (это может быть договор, контракт, соглашение ит.д.).

Для упрощения процесса набора обучающей выборки было реализовано приложение разметки (рис.2). Приложение реализовано на Python. В процессе набора датасета приложение позволило избежать ошибок и повысить скорость разметки предложений.

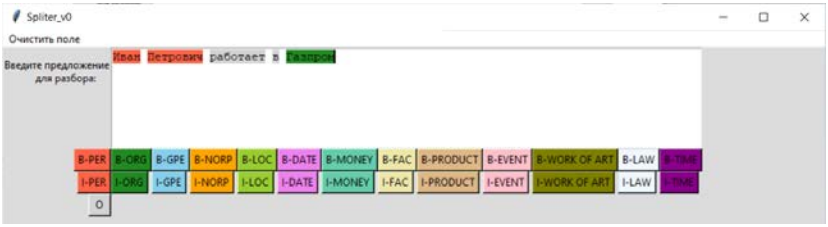


Рис. 2. Приложение для разметки датасета

2. Описание используемой модели

В качестве базовой модели НС для дообучения была выбрана модель bert-mult [3,6]. Модель bert-mult предобученна на двух задачах без учителя — моделирование языковых масок и предсказание

следующего слова в предложении. Это двунаправленная, нейросетевая модель от google на основе архитектуры transformer (Рис.3). Отличительной особенностью такой архитектуры является механизм внимания (self-attention). Это специальный новый слой, который дает возможность каждому входному вектору взаимодействовать с другими словами через attention mechanism, вместо передачи hidden state как в RNN или соседних слов как в CNN .

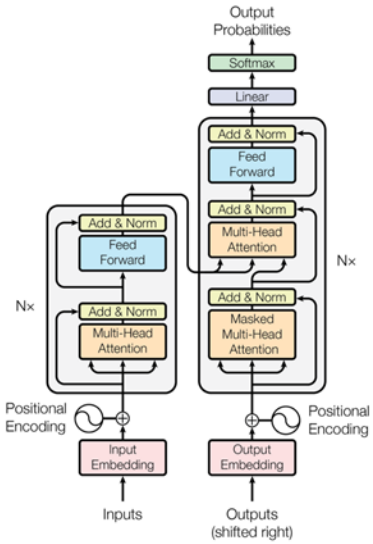


Рис. 3. Архитектура НС на базе Transformer

3. Выбор инструментов и средств разработки

В качестве языка программирования был выбран Python, так как именно на нем развивается анализ данных, под него существует большое количество библиотек и для него написаны NER модели BERT. В качестве IDE был выбран Google Collab, так как он предоставляет доступ к серверу с GPU, что просто необходимо при обучении модели.

На основе руководства к моделям и в ходе ряда дополнительных экспериментов подобранны оптимальные аргументы обучения: batch size=32, max_grad_norm = 10.0, 15 эпох.

4. Подходы к обучению и предобработке данных

Для дальнейшего обучения модели нам нужно было определиться с предобработкой данных для обучения. Для этой задачи была найдено несколько готовых решений:

– Библиотека tokenaizer[4], решение от haggin face. Помогает разбивать текст на набор токенов (на предложения и слова). После ряда экспериментов это решение в проект включено не было, так как регулирование уровня токенизации было невозможно, что приводило к разбиению не только предложений на слова, но и слова на слоги, что в процессе обучения вводило модель в заблуждение.

– Библиотека seqeval эффективный инструмент для предиктивного анализа данных. Была использована в проекте для оценивания модели после обучения.

5. Обучение модели и результаты

После того как нами выбраны инструменты, подключены и настроены все требуемые библиотеки, набран датасет, было проведено обучение модели и оценка результатов, полученных в ходе обучения.

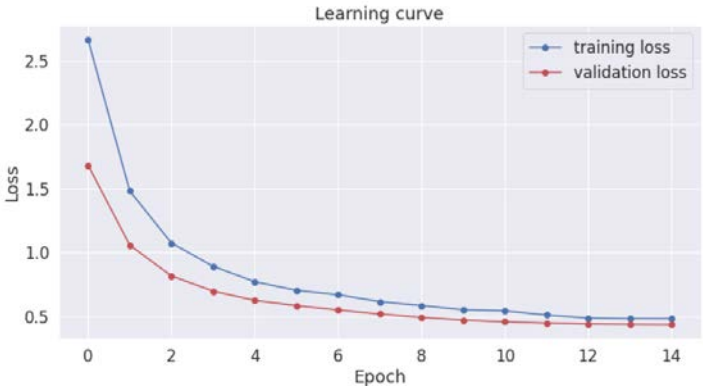


Рис. 4. График потерь тестирования и валидации

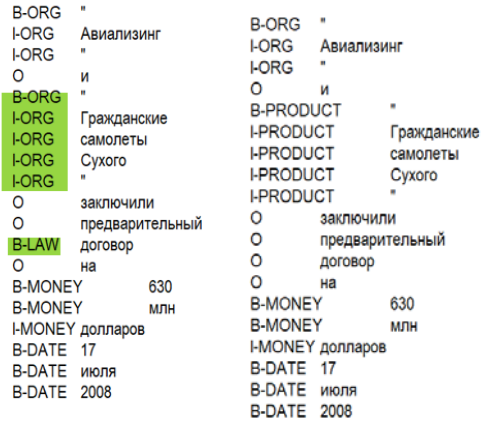
```
Average train loss: 0.6476007406254448
Epoch: 100%|██████████|15/15[08:27<00:00, 101.58s/it]Validation loss: 0.47649709632526344
Validation Accuracy: 0.9408052884615384

Validation F1-Score-weighted: 0.943301909406215
```

Рис. 5. Оценки модели после обучения

B-ORG	Посольство
B-GPE	России
O	в
B-GPE	Таллине
O	потребовало
O	немедленного
O	освобождения
B-PER	шеф
B-PER	-
B-PER	редактора
B-NORP	эстонского
I-PER	информационного
I-PER	портала
B-ORG	Sputnik
I-ORG	Meedia
B-PER	Елены
I-PER	Черышевой
I-PER	.

O	Посольство
B-GPE	России
O	в
B-GPE	Таллине
O	потребовало
O	немедленного
O	освобождения
O	шеф
O	-
O	редактора
B-NORP	эстонского
O	информационного
O	портала
B-ORG	Sputnik
I-ORG	Meedia
B-PER	Елены
I-PER	Черышевой

Рис. 6. Результаты тестирования
(дообученная модель- слева, базовая- справа)



B-ORG	"
I-ORG	Авиализинг
I-ORG	"
O	и
B-ORG	"
I-ORG	Гражданские
I-ORG	самолеты
I-ORG	Сухого
I-ORG	"
O	заключили
O	предварительный
B-LAW	договор
O	на
B-MONEY	630
B-MONEY	млн
I-MONEY	долларов
B-DATE	17
B-DATE	июля
B-DATE	2008

B-ORG	"
I-ORG	Авиализинг
I-ORG	"
O	и
B-PRODUCT	"
I-PRODUCT	Гражданские
I-PRODUCT	самолеты
I-PRODUCT	Сухого
I-PRODUCT	"
O	заключили
O	предварительный
O	договор
O	на
B-MONEY	630
B-MONEY	млн
I-MONEY	долларов
B-DATE	17
B-DATE	июля
B-DATE	2008

Рис. 7. Результаты тестирования
(дообученная модель- слева, базовая- справа)

Заключение

В статье описан подход к реализации задачи дообучения модели для распознавания именованных сущностей (NER) из текста. Результатом дообучения модели стало:

– Повышение качества распознавания уже существующих классов.

– Добавление нового класса распознавания LAW(документ, контракт), определяющий связи между организациями и персонажами.

– Программно реализовано приложение для разметки текста и с его помощь набран объемный датасет.

В дальнейшем это позволит в упрощенной форме перенастраивать NER модель на интересующую нас предметную область.

Список литературы

1. Открытый фреймворк для анализа текста и создания диалоговых систем. [Электронный ресурс]. – Режим доступа : https://deeppavlov.ai/

2. Burtsev M. DeepPavlov: Open-Source Library for Dialogue Systems / M. Burtsev [et al.] // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics-System Demonstrations (Melborne, Australia, 15-20 июля 2018 г.) – 2018. – С. 122-127.

3. Руководство для начинающих по BERT для мультиклассифиции. [Электронный ресурс]. – Режим доступа : https://www.machinelearningmastery.ru/beginners-guide-to-bert-for-multi-classification-task-92f5445c2d7c/

4. Классификация текста с помощью BERT Tokenizer и TF 2.0 в Python [Электронный ресурс]. – Режим доступа : https://pythobyte.com/text-classification-with-bert-tokenizer-and-tf-2-0-in-python-44cafd87/

5. Учебник по BERT WORD IMBEDINGS [Электронный ресурс]. – Режим доступа : https://russianblogs.com/article/60231465736/

6. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [Электронный ресурс]. – Режим доступа : https://arxiv.org/pdf/1810.04805.pdf

Анализ вариантов использования редакторов онтологий

В. С. Господарикова

Студент магистратуры

В. В. Гаршина

Доцент

Введение

В современном мире количество данных растёт с каждым днём. Это приводит к необходимости в обработке и структуризации данных для дальнейшего использования.

Онтология - формализованные с помощью концептуальной схемы данные о предметной области. Также такая структура представления информации значительно облегчает её использование. Именно поэтому в последнее время онтологии становятся все более популярными. Однако их составление на данный момент не является простой задачей.

Для создания онтологии на данный момент необходимы знания о предметной области и навыки управления специальным программным обеспечением. Поскольку существующие на данный момент редакторы онтологий имеют достаточно сложные и запутанные интерфейсы, то создание даже простой базы знаний требует одного из двух подходов. Первый – это долгое изучение самим экспертом принципов построения онтологий вне приложения, а также дополнительное изучение самой программы. Второй подход заключается в привлечении дополнительного посредника, который будет переводить рассказ эксперта предметной области в формализованный вид.

Эта трудность сильно затрудняет процесс создания и использования онтологий. Именно поэтому возникает необходимость в программе, которая сможет заменять посредника-человека и сама приведёт знания к необходимой структуре.

Самое главное в таком приложении – понять, какие именно функции она должна выполнять. В данной статье мы займёмся именно анализом необходимых редактору онтологии вариантов использования.

1. Постановка задачи

Главные задачи разрабатываемого приложения можно разделить на два типа:

1Предоставление интерфейса взаимодействия с человеком;

2Реализация взаимодействия с онтологией.

В данной статье мы считаем, что первая задача включает в себя распознавание команд на изменение онтологии и решается самыми комфортными для человека способами, например, текстовым или голосовым описанием.

Вторая же часть может быть реализована разными способами и включать в себя самые разные виды изменений узлов создаваемой базы знаний. По сути, взаимодействие с онтологией должно иметь функции соответствующие всем вариантам её модификаций. Таким образом, рассматривая варианты использования редактора такой структуры данных нужно учесть все варианты её изменений.

Поскольку онтологии, как понятие в информационных технологиях, используются с 1980-х годов, логично обратиться в анализе способов работы с данным представлением знаний к уже существующим аналогам.

Безусловно, существующее на данный момент программное обеспечение покрывает большую часть потребностей пользователя и может показать нам основные необходимые сценарии использования таких приложений.

2. Анализ существующих аналогов

Самым распространённым, лёгким инструментом, который позволяет удобно создавать небольшие онтологии является Protégé (by BMIR). Экран интерфейса для ознакомления представлен далее на рисунке 1.

Безусловно, этот редактор предоставляет все необходимые возможности, но имеет очень существенный недостаток: чем больше становится онтология, тем неудобнее с ней становится работать.

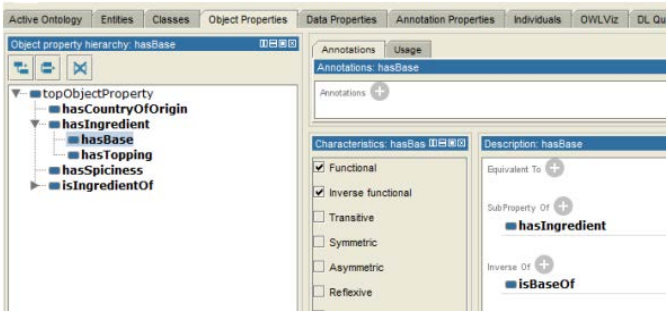


Рис. 1. Интерфейс Protégé

Следующим рассматриваемым аналогом будет GraphDB (by ontotext). Это не просто редактор, а графовая база данных, которая позволяет загружать в неё онтологии, редактировать их, искать в них данные и многое другое. Одной из главных особенностей такого инструмента является, конечно, возможность доступа к данным сред разработки, программ и приложений. Для предоставления удобного доступа к онтологии у GraphDB присутствует локальный веб-интерфейс. Примеры экранов интерфейса показаны на рисунках 2 и 3.

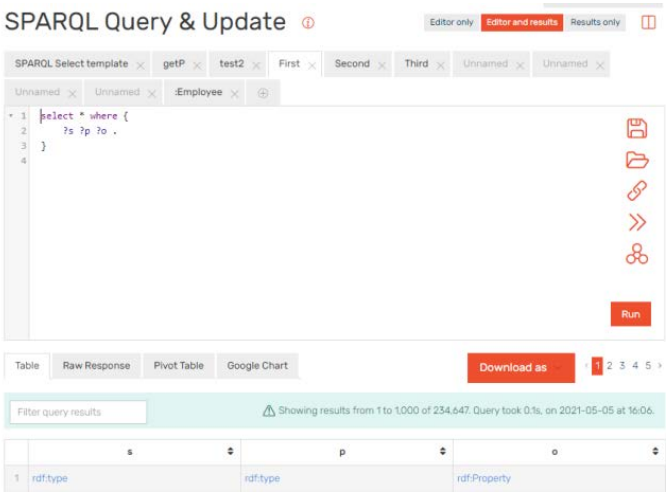


Рис. 2. Пример интерфейса запросов в GraphDB

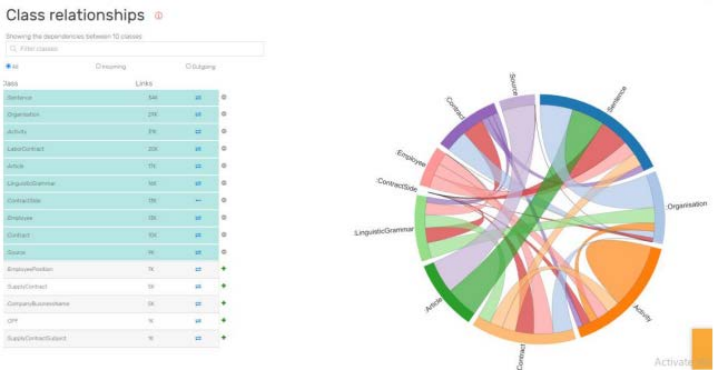


Рис. 3. Пример интерфейса взаимосвязей классов в GraphDB

На основании представленных выше редакторов были выявлены основные функции, необходимые для полноценного создания онтологии. Далее они будут перечислены и объяснены.

3. Варианты использования

Перед описанием самих классов введём определение. Инстанс – это конкретная реализация класса.

Теперь перейдём к вариантам использования. Приложение, создающее формализованную базу знаний, как правило, обладает следующими возможностями:

1. создание классов;
2. задание классовой иерархии;
3. добавление классам свойств;
4. определение для свойств типов данных;
5. создание инстансов;
6. редактирование инстансов или классов;
7. удаление инстансов или классов;
8. выборка инстансов или классов.

Рассмотрим по очереди подробнее каждую из функций.

Создание сущностей типа класс предполагает описание формальной модели объекта. Эксперт при описании класса должен определить его имя. Таким образом, изначально пустой класс лишь обозначает существование такого типа объектов, но не задаёт их характерные особенности.

Следующая функция – создание иерархии классов. Так эксперт может облегчить базу знаний, не создавая каждый тип объектов с нуля, а разделив их на большие группы со схожими свойствами. Например, можно разделить все в мире на два больших класса: живое и неживое. И задать живому такие характеристики как тип дыхания, питания и т.д.

Добавление свойств классам заключается в описании возможных характеристик объектов данного типа. Также каждому свойству необходимо указать тип данных, который можно выбирать из стандартных, либо создавать дополнительно. Конечно, эта функция в дальнейшем помогает не допустить лишних ошибок.

Создание инстансов – это одна из самых главных задач онтологии. Именно это действие наполняет базу знаниями. Каждый инстанс – это конкретный факт, существующий в предметной области и на его основе в дальнейшем можно проводить анализ.

Редактирование и удаление можно объединить в один пункт. Всё это касается изменений уже существующей онтологии. Также стоит упомянуть, что на уровне хранения и класс и его реализация являются

узлами графа с какими-то свойствами, поэтому их изменение для пользователя практически не имеет отличий.

Выборка инстансов также является очень важной задачей. После записи всех знаний в базу нам необходимо воспользоваться ими. Для этого нужно обратиться к онтологии и вывести конкретные факты по каким-то свойствам или названиям. Поэтому выборка является финальной из списка необходимых задач, завершая цикл использования онтологии пользователем.

Как мы видим, у редакторов графовых баз знаний есть много функций. Конечно, здесь перечислены самые необходимые из них, а не всё. Такие возможности, как например выведение правил на основании онтологии или их выполнение были опущены, поскольку они относятся к процессу не столько редактирования, сколько дальнейшего анализа.

Заключение

Хотелось бы сказать, что упрощённое редактирование онтологии важно для дальнейшего развития данной сферы обработки и представления знаний. Без эксперта предметной области любой редактор онтологии не приносит практической пользы. Также и эксперт без специальных знаний и умений не может применить онтологию для получения выгоды для себя. Поэтому снижение порога разделяющего две составляющих формализованной базы знаний – это очень важная задача.

Также хочу упомянуть, что представленные в данной статье редакторы онтологий, естественно, не единственные и существует множество других редакторов. Однако, они являются одними из самых известных и распространённых инструментов и предоставляют пользователю все основные и некоторые дополнительные функции по работе с графовыми базами данных.

Список литературы

1. Митрофанова, О. А. Онтологии как системы хранения знаний / О. А. Митрофанова, Н.С. Константинова. – СПбГУ, 2008. – 54 с.

2. GraphDB documentation [Электронный ресурс] : документация – Режим доступа : https://graphdb.ontotext.com/documentation/free/

3. Муромцев, Д. И. Онтологический инжиниринг знаний в системе PROTÉGÉ / Д.И. Муромцев. – СПбГУ ИТМО, 2007. – 62 с

Преимущества гексагональной архитектуры в микросервисном приложении

В. В. Денисов

Студент магистратуры

Н. К. Самойлов

Старший преподаватель

Введение

В мире разработки программного обеспечения с накоплением опыта стали образовываться эффективные архитектуры. Компьютеры эволюционировали, развивались операционные системы, увеличивались вычислительные мощности, удешевлялись вычислительные устройства — все это приводило к появлению новых типов программного обеспечения и, соответственно, новых архитектур приложения.

Один из самых современных типов организации программной системы является микросервисная система, где каждое приложения является небольшим по размеру и по зоне ответственности сервисом. К особенностям таких сервисов можно отнести большое количество взаимодействий с другими микросервисами, внешними системами, базами данных.

С развитием подходов к разработке, тестированию, публикации микросервисов появились различные виды архитектур. Одна из них - гексагональная, по мнению сообщества программистов является подходящей практически для всех видов микросервисов.

Но данная система является относительно молодой и слабоизученной. Многие современные архитекторы приложений, разрабатывая новые системы, продолжают использовать устаревшие подходы к разработке микросервисных систем. Изучение гексагональной архитектуры на данный момент является очень актуальной задачей как для разработчика, так и для архитектора программного обеспечения.

1. Использование гексагональной архитектуры

Гексагональную архитектуру используют преимущественно в микросервисных системах, хотя её применение возможно и в монолитном приложении. Последнее сопряжено с рядом трудностей,

одно из которых – работа с унаследованной кодовой базой. Дело в том, что гексагональная архитектура представляет собой определенный подход к разработке, включающий в себя разделение приложения на слои. В Java, например, разделение на слои можно выполнить с помощью определения иерархии пакетов, где определенные классы сгруппированы вместе. Но в монолитных приложениях исторически складывается очень объемная кодовая база, которую сгруппировать очень сложно, как и в принципе сложно выполнять её рефакторинг.

Но тем не менее, гексагональную архитектуру можно применить и в монолитном приложении. Очень хорошо она себя проявляет вместе с паттерном «вырождающееся приложение» [1]. Этот паттерн позволяет медленно уходить от монолитной архитектуры, заменяя монолит системой микросервисов.

Исходя из вышесказанного, гексагональная архитектура может быть лучше всего применена в микросервисном приложении. Именно в микросервисах можно наблюдать за полным раскрытием всего потенциала этого подхода.

Кроме того, гексагональную архитектуру используют и просто для обозначения наличия архитектуры. Любой подход к разработке, будучи где-то написанным, зафиксированным, или просто обговоренным в команде разработчиков, уже приносит полезные улучшения в продукт – стандартизацию. Стандартизация подхода для написания программ является хорошим способом для решения проблемы возрастания технического долга. Гексагональная архитектура является отличным кандидатом для этой роли. Тем не менее, этим плюсом можно было бы отметить любую архитектуру.

2. Построение гексагональной архитектуры

Гексагональная архитектура основывается на принципе разделения слоёв или модулей, называемых гексагонами. Данная архитектура имеет множество вариантов реализации, большинство из которых обычно подразумевают наличие трёх гексагонов – доменной модели, приложения и фреймворка. Расположение элементов гексагональной архитектуры относительно гексагонов представлено на рис. 1.

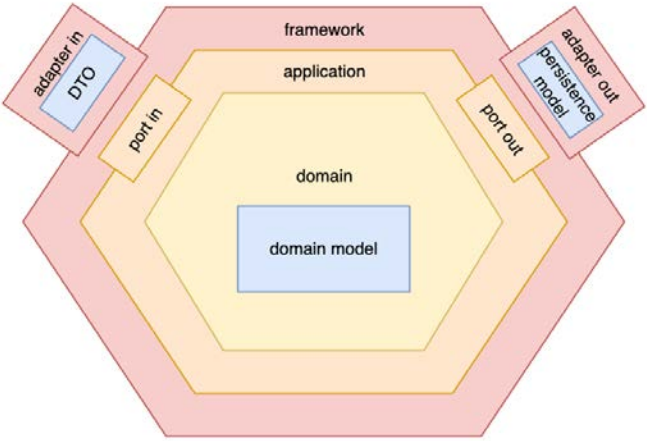


Рис. 1. Схема микросервиса с использованием гексагональной архитектуры

Разделение на слои повышает уровень абстракции, например - с помощью использования интерфейсов или абстрактных классов. В Java подобное разделение можно представить следующим образом (далее основные элементы гексагональной архитектуры в порядке их вызова):

1. Использование входных адаптеров. В современном микросервисном мире это чаще всего http-контроллеры.
2. Использование сценариев использования. Позволяют описать варианты использования гексагона приложения. Являются интерфейсами.
3. Использование входных портов. Реализуют сценарии использования, неявно внедряются во входные адаптеры.
4. Использование выходных портов. Являются интерфейсами, используются входными портами.
5. Использование выходных адаптеров. Реализуют выходные порты, используются неявно во входных портах, гексагон приложения не знает о конкретных реализациях.

Визуализация взаимосвязей элементов гексагональной архитектуры представлена на рис. 2.

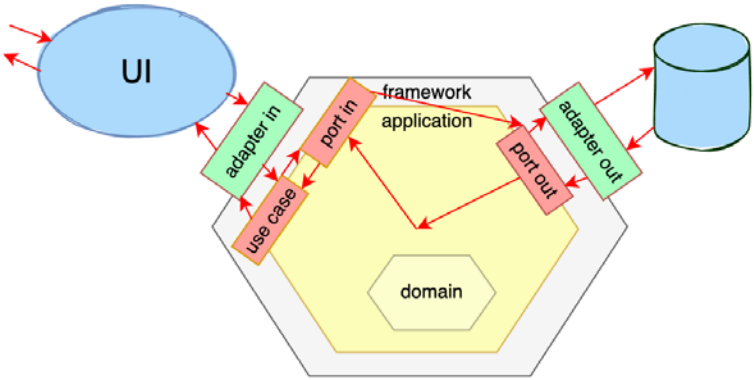


Рис. 2. Последовательность вызовов
элементов гексагональной архитектуры

Данная цепочка вызовов является для многих проектов оптимальной. Изучая различные варианты реализации сервисов, основанных на использовании этой цепочки вызовов, были активно применены такие паттерны, как инверсия зависимостей и принцип единой ответственности. В ходе изучения гексагональной архитектуры на практике было установлена, что данная последовательность позволяет определить несколько основных её преимуществ.

3. Преимущества гексагональной архитектуры

Преимуществами гексагональной архитектуры отчасти являются преимущества и других, смежных архитектур. Все они стремятся организовать кодовую базу, замедлить появление технического долга, и повысить общее качество продукта [2]. Но можно и выделить явные преимущества гексагональной архитектуры.

К основным преимуществам гексагональной архитектуры относят:

– устойчивость к изменениям;
– легкость в обслуживании;
– легкость в тестировании.

4. Устойчивость к изменениям

Если изменение не затрагивает фундаментальные бизнес-процессы, реализованные в приложении, то его реализовать гораздо проще, чем без использования данной архитектуры. Самые простые изменения включают в себя изменение реализации адаптеров, то есть интерфейса взаимодействия с пользователем или со внешними системами. Или же может возникнуть потребность в изменении логики других элементов

системы, что приведет к изолированным изменениям соответственных модулей. Качество устойчивости к изменениям является противоположным по отношению к объёму спагетти-кода, который поддерживать сложно и имеется тенденция к ещё большему усложнению [3].

Требование устойчивости к изменениям является нефункциональным и его практически невозможно отследить или измерить на практике, но эта устойчивость присутствует и к ней стоит стремиться, в том числе применяя подход гексагональной архитектуры.

Очень многие нефункциональные требования, такие как это, являются важными для проектов. Говоря о следующих преимуществах, стоит понимать, что они тоже являются решением для соответственных нефункциональных требований – к обслуживанию и тестируемости. В некотором смысле, использование какой-либо архитектуры, в частности гексагональной, подразумевает борьбу с возникновением технического долга и контроль реализаций нефункциональных требований.

5. Легкость в обслуживании

Разделение приложения на определенные модули облегчает разработчикам добавление нового функционала, причем легче становится как добавление новых сценариев использования, так и новых путей взаимодействия (API). Это достигается благодаря тому, что в каждом модуле все элементы находятся на одном уровне в иерархии, при этом каждый модуль реализует определенный функционал, имеет определенную ответственность.

Добавление нового функционала зачастую сопряжено с проблемами в программах, не придерживающихся определенного архитектурного стиля. К таким проблемам можно отнести множество точек входа в программу, не связанных между собой, в том числе эти точки могут находиться в одном месте с бизнес-логикой приложения; запутанные интерфейсы; «божественные» классы и другие анти-паттерны. С использованием гексагональной архитектуры приходит упорядоченность в кодовой базе – уменьшается связанность (англ. cohesion) модулей друг с другом, увеличивается его связность (англ. coupling) [3].

Легкость в обслуживании – тоже нефункциональное требование. Иногда оно формулируется как отдельное требование, но чаще – нет, существует лишь общее требование к качеству продукта, где легкость в обслуживании измеряется лишь косвенно.

6. Легкость в тестировании

Каждый модуль можно протестировать отдельно от вышестоящих модулей. Это исходит из низкой связанности модулей, описанной ранее.

В случае с гексагоном приложения можно использовать заглушки вместо портов, так как реализаций для них в этом модуле нет. Кроме того, именно с использованием гексагональной архитектуры облегчается тестирование сценариев в гексагоне приложения. Тестирование сценариев в обычном приложении, без явного разделения на модули или слои, превращается в интеграционное, или даже компонентное тестирование, требующее наличие систем для интеграции, которые в большинстве случаев для данного типа тестов не нужны, а в некоторых случаях и практически невозможны (как, например, очень сложно предоставить интеграцию с внешним сервисом для тестового приложения).

В случае гексагонов предметной области и приложения тестирование происходит без использования технологического кода, который определяется в гексагоне фреймворка, что также облегчает тестирование. Существует несколько подходов к написанию этих гексагонов таким образом, чтобы гексагон фреймворка не имел никаких трудностей во взаимодействии с нижестоящими гексагонами [3].

Легкость в тестировании является одним из важнейших преимуществ гексагональной архитектуры. Благодаря изоляции модулей друг от друга, разработчики или тестировщики пишут тесты гораздо быстрее, покрывая большую часть кода тестами, создавая более качественный продукт.

Легкость в тестировании, иногда также называемую тестируемостью приложения, является важным нефункциональным требованием. Сложно при разработке проекта использовать какие-либо метрики для контроля тестируемости, т.е. скорости написания тестов, их сложности и возможности покрытия кода.

Заключение

Данная статья посвящена исследованию преимуществ гексагональной архитектуры. Гексагональная архитектура, как и другие архитектуры, приобретающие популярность, обладает рядом преимуществ. В результате исследования было выявлено 3 основных преимущества, а также некоторые замечания к данному архитектурному стилю. Эти преимущества очень ценны, а во многих проектах они критичны. Например, банковские приложения, выполняющие денежные операции, просто не могут быть написаны без архитектурного подхода (т. е. некачественно) по определению.

Кроме того, были сделаны выводы о роли гексагональной архитектуры в удовлетворении нефункциональных требований приложения.

В современных микросервисных системах гексагональная архитектура занимает свое место одной из самых популярных архитектур.

Список литературы

1. Шаблоны корпоративных приложений / М. Фаулер [и др]. – Москва : Вильямс, 2016. – 544 с.

2. Viera, D. Designing Hexagonal Architecture with Java. / D. Vieira. – Amazon Kindle, 2022. – 220 p.

3. Martin, R. The Clean Coder: A Code of Conduct for Professional Programmers / R. Martin. – Boston : Pearson Education, 2011. – 205 p.

Распределение канальных ресурсов в сетях когнитивного радио

М. И. Игнатов

Студент магистратуры

А. Ю. Савинков

Профессор

Введение

В настоящее время большая часть частотного диапазона вещания занята и активно используется, но некоторые каналы связи используются не все время и из-за этого потенциальная эффективность использования частотного спектра уменьшается [1]. Чтобы рационально распределить нагрузку на каналы, оптимизировать работу сети и повысить коэффициент использования спектра необходимо применить в радиосистеме особую технологию, которая сможет обеспечить постоянную занятость каналов связи таким образом, чтобы не пострадало качество связи абонентов, которые привязаны к определенному частотному диапазону. Для решения этой задачи в 2011 году вводится в эксплуатацию стандарт связи IEEE 802.22.

Вместе с тем необходимо отметить, что из-за стремительного развития технологий беспроводной связи и существующей необходимости создания большой радиосети состоящей из множества станций, которая будет способна оказывать широкий спектр абонентских сервисов таких как: поддержание видеосвязи хорошего качества, передача сообщений через электронную почту.

1. Определение когнитивного радио

В настоящее время определение термину "когнитивное радио" можно объяснить следующим образом: «Система когнитивного радио- это совокупность модулей связи различного назначения»[2]. В ней присутствует контролирующий модуль, в который пользователь или администратор системы, ввел алгоритмы работы. С помощью этого модуля система собирает массив различных данных со всех остальных подключенных элементов радиостанции в базу данных, после проведения вычислений оптимальных значений она постоянно отслеживает изменения на приемнике и автономно корректирует свои

рабочие параметры согласно полученным данным [3]. Главной целью работы модуля и всей системы в целом является поиск свободного диапазона частот и таким образом обеспечение помехоустойчивого качество приема и передачи. Система должна обладать интуитивно понятным пользовательским интерфейсом для облегчения ее настройки, ввода в эксплуатацию и своевременного обслуживания и обновления ее составляющих.

2. Устройство когнитивного радио

Для того, чтобы создать устройство для реализации технологии когнитивного радио (элемента самоорганизующейся сети), предлагается следующий комплекс устройств:

- Средства, обеспечивающие работу сети
- Средства предоставления услуг.
- Средства приема и передачи каналов связи и их обработки.
- Пользовательский интерфейс взаимодействия с системой.
- Устройства, снабженные модулем Wi-Fi приемника.

Представленное устройство или совокупность из нескольких таких устройств, образующих полноценную когнитивную радиосеть, обладает следующими свойствами:

- при помощи постоянного сканирования частотного диапазона происходит постоянный анализ радиоэлектронной обстановки;
- во время работы алгоритма поиска каналов связи система определяет наилучшие из оптимальных частотных каналов, происходит автоматический выбор рабочей частоты и других параметров физического канала;
- автоматическое постоянное отслеживание параметров канала, управление остальными подключенными устройствами связи через интерфейс управления с помощью ЭВМ;
- формирование перечня услуг связи сети таких как: телефония, поддержка видеосвязи, обмена короткими сообщениями, услуг электронной почты;
- обеспечение постоянного доступа к перечню сервисов самоорганизующейся сети на основе процедуры авторизации абонента;

Таким образом, рассмотрен основной способ построения работоспособного адаптирующегося устройства сети когнитивного радио, который может стать основой для одного устройства или комплекса таких изделий, объединенных в сеть.

3. Основные алгоритмы построения когнитивного радио

Работу когнитивного приемопередающего устройства можно сделать еще более эффективной с помощью внедрения технологий

искусственного интеллекта для проведения более быстрого и точного этапа анализа параметров окружающего частотного диапазона, проведения расчетов для выявления наиболее оптимальных каналов связи и принятия решений.

Алгоритмы когнитивной радиосвязи предусматривают следующие необходимые этапы:

– поиск свободного фрагмента частотного спектра;
– выбор из доступных лучший канал для обеспечения связи;
– работа абонента на данном диапазоне частот;
– поиск другого свободного диапазона, если привязанный к этому каналу абонент начинает им пользоваться.

Работу когнитивного приемопередающего устройства можно сделать еще более эффективной с помощью внедрения технологий искусственного интеллекта для проведения более быстрого и точного этапа анализа параметров окружающего частотного диапазона, проведения расчетов для выявления наиболее оптимальных каналов связи и принятия решений.

Элемент самоорганизующейся сети Ej выбирает физический наилучший по измеренному соотношению сигнал / шум на соответствующей частоте канал ki из множества K\ K' случайным образом. Если в процессе построения множества K' были обнаружены служебные сообщения алгоритма кластеризации, то они обрабатываются соответствующим образом сразу после выбора канала ki. При получении нового перечня сигналов канал выбирается заново. За счет наличия блока навигационного приемника возможно закреплять определенные географические зоны за отдельными подмножествами K, так что элемент самоорганизующейся сети выбирает каналы из некоторого подмножества K, только находясь в закрепленных за этим подмножеством зонах, в результате чего можно достичь экономии частотного ресурса за счет повторного использования одних и тех же каналов в достаточно удаленных друг от друга зонах, а также – оптимизировать алгоритм кластеризации.

Рассмотрим алгоритм кластеризации. Пусть N – максимальное количество приемопередатчиков в кластере. На рисунке ниже стрелками изображена рассылка сообщений между радиостанциями E11, E21, ..., EN1, ..., EM1, ..., EQ1, входящими в состав элементов самоорганизующейся сети E1, E2, ..., EN, ..., EM, ..., EQ соответственно (см., например, рисунок) в процессе кластеризации, причем под обозначением каждой радиостанции расположены оси времени, на которых римскими цифрами I, II, III, IV обозначены этапы кластеризации. Предполагается, что станции E11, E21, ..., EN1 войдут в

один кластер. На рисунке процесс кластеризации изображен с помощью диаграммы, в которой под названием каждого узла размещена ось времени, на одной из осей отмечены этапы кластеризации, а стрелки обозначают передачу узлом сообщения в некоторый момент времени, отмеченный на оси времени, соответствующей данному узлу, началом стрелки, и прием этого сообщения другим узлом сети в некоторый другой момент времени, отмеченный на соответствующей оси времени концом стрелки.

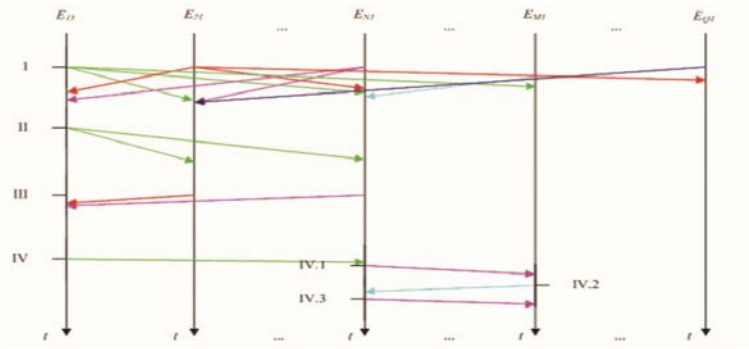


Рисунок. Кластеризация

Заключение

Данная статья посвящена разработке и реализации программного модуля для устройства когнитивного радио. Описаны алгоритмы распределения и выбора каналов и устройства реализующего данную технологию. Использование данной технологии позволяет оптимизировать использование частотного спектра радиовещания. В дальнейшем предполагается реализация одного из алгоритмов, основанных на данной технологии, построение моделей сетей.

Список литературы

1. Николашин, Ю. Л. SDR радиоустройства и когнитивная радиосвязь в декаметровом диапазоне частот / Ю. Л. Николашин. – 2015. – 20 с.

2. Doyle, L. Essentials of Cognitive Radio / L. Doyle – Cambridge University Press, 2009. – 252 p.

3. Авдонин, Д. В. Интеллектуальные радиосистемы: когнитивное радио / Д.В. Авдонин // Информационные технологии. Системы, средства связи и управления: Информационно-аналитический сборник Под ред. С. В. Ионова; ОАО «Концерн «Созвездие». – 2012. – №1. – С. 115-117.

Регуляризация скелетов бинарных изображений

А. Д. Ильина

Студент магистратуры

В. В. Фертиков

Доцент

Введение

Срединной осью, или скелетом, фигуры называют связный граф, центрированный относительно её границ и сохраняющий топологические свойства исходного объекта. Скелет является альтернативой классическому граничному представлению формы, поэтому любые его преобразования эквивалентны преобразованию самого объекта. Это качество срединных осей широко используется в задачах визуализации, сегментации изображений, распознавании образов, оптимизации и многих других.

Существует немалое количество способов получения скелетов, как дискретных, так и непрерывных. К сожалению, все они в той или иной степени порождают проблему регуляризации, то есть потребность в устранении коротких ветвей, которые не вносят существенного вклада в описание формы фигуры. Данная статья посвящена рассмотрению способов решения этой задачи и их сравнению. В качестве алгоритма скелетизации используется модификация непрерывного подхода на основе диаграмм Вороного.

1. Проблема нестабильности скелетов

Шумовые ветви возникает в срединных осях при любом алгоритме скелетизации. Связано это с тем, что скелет фигуры крайне чувствителен к малейшим нерегулярностям границы объекта. На рис. 1 показано, как даже незначительное изменение контура влияет на результирующий скелет. Без дополнительной обработки такие скелеты невозможно было бы использовать для решения прикладных задач.

Это свидетельствует о необходимости получения «чистого» скелета, без лишних ветвей. Привести скелет к упрощённому виду можно через удаление (стрижку) ветвей, несущих меньшее количество информации о структуре фигуры по сравнению с остальными. При этом

в общем случае задаётся некоторый параметр регуляризации a, ограничивающий набор ветвей, пригодных для обрезки.

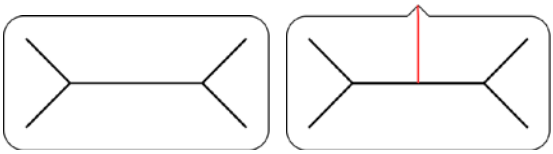


Рис. 1. Изменение границы фигуры приводит к появлению новой ветви скелета

Например, можно в качестве регулирующего оператора использовать операцию морфологического размыкания. В этом случае a определяет размер структурного элемента. Такой наивный подход, однако, зачастую требует ручной настройки и подгонки параметров стрижки под конкретное изображение. К тому же, необходимо задать критерий остановки процесса стрижки, — в противном случае есть риск потерять существенную топологическую информацию.

Следовательно, в результате регуляризации мы хотим получить некоторую «укороченную» версию полного скелета, основываясь на автоматизированном критерии принадлежности каждой ветви к наиболее устойчивой части графа. Множество таких ветвей называется базовым скелетом [1]. Иными словами, регуляризация эквивалентна задаче поиска базового скелета.

2. Методы регуляризации

Алгоритм обрезки срединных осей должен искать компромисс между простотой полученной структуры и сохранением топологии. Ниже приведены три метода [2-4], каждый из которых включает в себя процесс последовательной стрижки ветвей. Ключевое различие описываемых подходов состоит в выборе критерия, по которому удаляется следующая ветвь.

1. Эпсилон-критерий (ЭК)

Положим, что терминальная ветвь скелета имеет внутреннюю вершину A и конечную вершину B. Пусть от некоторой внутренней вершины отходит k терминальных ветвей b_1, …, b_k. Очевидно, что точка A будет являться смежной для каждой из терминальных вершин B_i.

Предположим, что одна из ветвей была удалена в ходе обрезки, что повлекло скругление некоторых из углов фигуры. Под скруглением понимается сопряжение двух линейных сегментов границы дугой вписанной окружности радиуса r_A с центром в A. Тогда критерий удаления терминальной ветви определим следующим образом:

$$\lambda(A) = \max d_e(A, B_i) - r_A \le \varepsilon, \quad (1)$$

где λ(A) – расстояние от границы исходной фигуры до скруглённого контура после удаления ребра; $d_e(A,B_i)$ – евклидово расстояние между внутренней и терминальной вершинами; ε – фиксированная величина.

В начальный момент времени в список кандидатов на отсечение попадают все терминальные ветви полного скелета. Если при удалении ветви её внутренняя вершина становится терминальной, то связанная с ней ветвь также попадает в список. Операция повторяется до тех пор, пока список кандидатов не опустеет.

2. Критерий аппроксимации длины (КАД)

Представим каждую терминальные ветвь скелета набором точек $b=\{p_0, \ldots, p_n\}$ (p_0 – точка разветвления, p_n – терминальная вершина). Тогда ветвь b необходимо удалить, если справедливо условие:

$$\sum_{i=0}^{n-1} d_e(p_i, p_{i+1}) \le S \cdot r_0, \quad (2)$$

где S – параметр, определяющий расстояние между границей фигуры и контуром восстановленной из скелета фигуры; r_0 – радиус максимальной вписанной окружности в точке разветвления.

Оговаривается, что в случае наличия у точки разветвления более одной терминальной ветви, удовлетворяющей данному критерию, за одну итерацию удаляется только самая короткая ветвь. Это позволяет сохранить последнюю ветвь, и таким образом избежать потерю полезной структурной информации.

3. Критерий реконструкции (КР)

Под реконструкцией понимается восстановление фигуры из скелета при условии, что известны радиусы всех вписанных пустых кругов. В этом варианте регуляризации предлагается использовать жадный алгоритм, который строит последовательность из укороченных скелетов $\{S_1, \ldots, S_n\}$ путём отсечения терминальных ветвей до тех пор, пока в скелете не останется лишь одно ребро. На каждой итерации для устранения выбирается ребро с минимальным весом w_i, где:

$$w_i = \alpha \frac{\Lambda(F - R(S_i))}{\Lambda(F)} + \log(L(S_i) + 1), \quad (3)$$

где α – параметр, определяющий соотношение точности реконструкции к сложности структуры скелета; F – исходная фигура; S_i – укороченный скелет фигуры после удаления i-ой ветви; Λ(*) – площадь в пикселях; R(*) – реконструкция фигуры по скелету; L(*) – нормированная длина кривой.

Параметр α определяется:

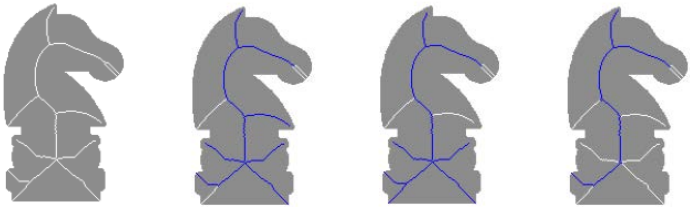
$$\alpha = \beta \log \frac{L(S_i)}{\omega(S_i)} \qquad (4)$$

Здесь β=9 – константа [4], $\omega(S_i)$ – средняя длина всех конечных путей в скелете. Получив таким образом множество скелетов-кандидатов, остаётся выбрать тот скелет S^*, при построении которого вес w_i имел глобально наименьшее значение.

3. Результаты экспериментов

Описанные выше алгоритмы были реализованы с помощью языка Python. Целью экспериментов является качественное и количественное сравнение результатов регуляризации. Набор входных данных состоит из 110 бинарных изображений размером до 512x1024 пикселей.

На рис. 2 представлен пример результатов стрижки. Качественный анализ показал, что на простых объектах (таких как буквы естественных языков и цифры) скелеты, полученные разными методами, практически не отличаются друг от друга. Более того, по критериям КАД и ЭК регуляризация зачастую не производится совсем. Для фигур более сложной формы можно отметить, что скелеты со стрижкой по КР получаются заметно более «простыми», с меньшим числом терминальных ветвей. Обрезка по ЭК даёт, как правило, средний результат между КАД и КР, с приоритезацией в пользу точности восстановленного контура.



а б в г

а – срединная ось, б – ЭК (ε=7), КАД (S=1,25), г – КР (β =9)

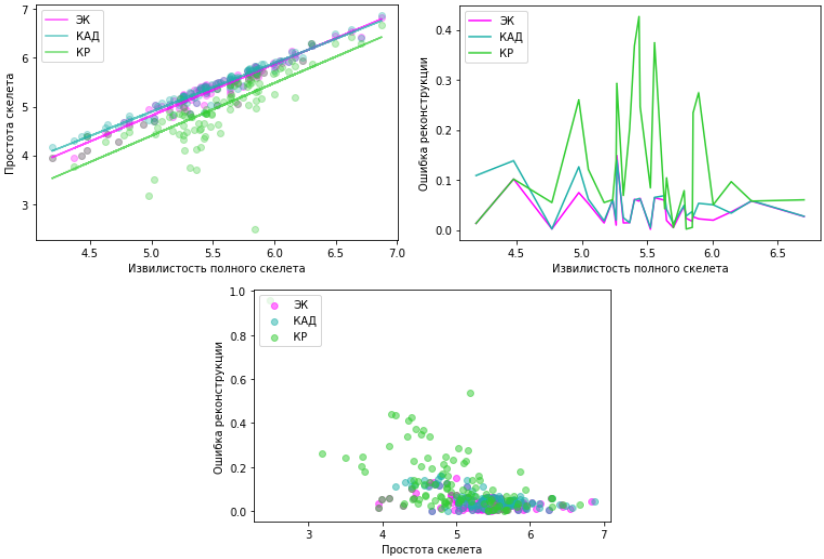
Рис. 2. Регуляризация скелета:

Количественная оценка осуществлялась с помощью значений ошибки реконструкции RER и простоты структуры скелета SS [3,4]. Ошибка реконструкции является показателем точности скелета. Простота структуры важна в прикладных задачах, где требуется сравнивать скелеты друг с другом.

$$RER(S) = \frac{\Lambda(F - R(S))}{\Lambda(F)}, \tag{5}$$

$$SS(S) = \log(L(S) + 1). \tag{6}$$

На графиках рис. 3 представлена зависимость этих значений от извилистости полного скелета, а также их взаимное соотношение. Извилистость определяется как отношение длины скелета к площади фигуры. Кроме того, для каждого метода было рассчитано среднее отклонение в пикселях при вращениях – показатель устойчивости скелета к геометрическим преобразованиям. Результат расчётов представлен в таблице.



соотношение между простотой регуляризованного скелета, ошибкой реконструкции и извилистостью полного скелета

Рис. 3. Количественная оценка стрижки

Таблица

Среднее отклонение скелета от срединной оси при вращении в зависимости от критерия регуляризации

Критерий	**Среднее отклонение**		
	π/6	**π/3**	**π/2**
ЭК	0,51	0,63	0,22
КАД	1,0	1,0	0,77
КР	0,65	0,7	0,44

Заключение

В данной статье был проведён обзор трёх подходов к регуляризации скелетов. Как показали результаты экспериментов, каждый из методов пригоден для решения поставленной задачи, так как ни один из них значительно не нарушает топологию скелета при вращении. Тем не менее, присутствуют различия в ошибке реконструкции и простоте базового скелета фигуры в зависимости от критерия обрезки. Наибольшую ошибку дал критерий реконструкции, тогда как эпсилон-критерий и критерий аппроксимации длины оказались более склонными к сохранению точности.

Таким образом, при выборе типа регуляризации необходимо брать во внимание требования конкретной предметной области. И хотя каждое из рассмотренных здесь решений нельзя назвать универсальным в отдельности, сочетание разных подходов способно дать эффективный баланс между точностью и простотой скелета.

Список литературы

1. Местецкий, Л. М. Непрерывная морфология бинарных изображений: фигуры, скелеты, циркуляры / Л. М. Местецкий – М.: ФИЗМАТЛИТ, 2009. – 288 с.

2. Местецкий, Л. М., Рейер, И. А. Непрерывное скелетное представление изображения с контролируемой точностью / Л. М. Местецкий, И. А. Рейер // Труды XIII Международной конференции по компьютерной графике и зрению «ГрафиКон» (Москва, 5-10 сентября 2003 г.). – Москва, 2003. – С. 92-96.

3. A new method for skeleton pruning / Philia-Buitrago, L.A. [et al.] // Lecture Notes in Computer Science, vol. 8495 : Pattern Recognition, 6th Mexican Conference (2014). – P. 301-310.

4. Skeleton pruning as trade-off between skeleton simplicity and reconstruction error / W. Shen [et al.] // Sci. China Inf. Sci. – 2013. – № 56. – P. 1–14.

Разработка инструментария автоматизированного тестирования сетевого стека беспроводных устройств

М. А. Карпова

Студент магистратуры

Д. И. Соломатин

Старший преподаватель

Введение

В настоящее время люди активно используют беспроводные устройства – беспроводные наушники, фитнес-браслеты, умные пылесосы, если говорить о проект «Умный дом», то нельзя не отметить умные лампочки, датчики движения, термостаты и т.д. Гарантией того, что пользователи будут получать качественный продукт – это тестирование беспроводных устройств и сетевого стека. Хотелось бы отметить, что тестирование сетевого стека сильно отличается от тестирования сайтов, мобильных приложений, отличительными чертами является отсутствие графического интерфейса и невозможность использования отладчика.

Единственными способами проверки качества сетевого стека служат чтение лог-файлов и анализ трафика, который может быть получен с помощью устройства сниффер.

Каждая из компаний, которые разрабатывает сетевой стек, используют свои разработки, и программного обеспечения, предназначенного для автоматизации тестирования сетевого стека, в свободном доступе нет.

В ходе данной работы был разработан инструментарий для системы автоматизированного тестирования сетевого стека. Указанная система является проприетарным решением. Проверка работы разработанного инструментария проводилась на устройствах, которые используют протокол Zigbee [1].

Для достижения цели, для разработки инструментария к системе автоматизированного тестирования сетевого стека, были поставлены следующие задачи:

– Изучение системы автоматизированного тестирования сетевого стека;

- Сбор требований к инструментарию;
- Проектирование и разработка инструментария.

1. Изучение системы автоматизированного тестирования сетевого стека беспроводных устройств

Принцип работы данной системы можно описать следующими шагами:

1. Прошивка устройств необходимыми бинарными файлами тестируемых приложений.
2. Запуск приложений на плате.
3. Перехват трафика и по возможности лог-файлов.
4. Валидация пакетов, запись результатов в лог-файл.

Первый пункт осуществляется программистами, которые пишут приложения-прошивки.

Второй пункт осуществляется системой автономно, приложения могут быть запущены как на платах, так и на симуляторе.

Третий пункт включает в себя работу сниффера, который ловит пакеты, являющиеся результатом общения устройств во втором пункте. Для перехвата трафика используется программное обеспечение Wireshark [2]. Программа может записывать трафик в файлы с расширениями PCAP (Packet Capture), PCAPNG (Packet Capture Next Generation) и PDML (Packet Description Markup Language). PDML соответствует стандарту XML и содержит подробную информацию о разбиении пакетов. На рис.1 показан пример PDML файла.

```
</proto>
  <field name="" show="Link Quality Indicator: 0" size="8" pos="20" value="0a00010000000000">
    <field name="wpan-tap.tlv.type" showname="TLV Type: Link Quality Indicator (10)" size="2" pos="20" show="10
    <field name="wpan-tap.tlv.length" showname="TLV Length: 1" size="2" pos="22" show="1" value="0100"/>
    <field name="wpan-tap.lqi" showname="Link Quality Indicator: 0" size="1" pos="24" show="0" value="00"/>
    <field name="wpan-tap.tlv.padding" showname="Padding: 000000" size="3" pos="25" show="00:00:00" value="0000
  </field>
  <field name="" show="Channel assignment: Page: Default (0), Number: 21" size="8" pos="28" value="030003001500
    <field name="wpan-tap.tlv.type" showname="TLV Type: Channel assignment (3)" size="2" pos="28" show="3" valu
    <field name="wpan-tap.tlv.length" showname="TLV Length: 3" size="2" pos="30" show="3" value="0300"/>
    <field name="wpan-tap.ch_num" showname="Channel: 21" size="2" pos="32" show="21" value="1500"/>
    <field name="wpan-tap.ch_page" showname="Page: Default (0)" size="1" pos="34" show="0" value="00"/>
    <field name="wpan-tap.tlv.padding" showname="Padding: 00" size="1" pos="35" show="00" value="00"/>
  </field>
</proto>
```

Рис. 1. Пример PDML файла

Особое внимание следует уделить четвертому пункту, поскольку разработанный инструментарий позволяет улучшить и оптимизировать процесс валидации. Валидация осуществляется с помощью валидационный скриптов, написанных на языке Python.

Валидационный скрипт включает в себя набор проверок, которые заключаются в том, чтобы найти в PDML файле нужное поле, получить

значение указанного атрибута и сравнить с ожидаемым. Проверки такого рода являются простыми и не удовлетворяют требованиям работы с беспроводными устройствами. Представим ситуацию, когда координатор шлет сначала один запрос, на который несколько устройств должны ответить. Нельзя угадать, в какой последовательности будут получены эти ответы. Или представим ситуацию, когда координатору должен ответить только одно устройство, но какое – неизвестно. С этой целью в системе были реализованы сложные условия И (and), ИЛИ (or), ПОКА (until), ПОСЛЕДОВАТЕЛЬНО (sequential), ПАРАЛЛЕЛЬНО (parallel).

В ходе выполнения валидационного скрипта все результаты проверок записываются в лог-файл, представленный на рис. 2.

```
ource]      2022-04-28 03:20:09,006:        received a packet, page: 0, channel: 21
]   2022-04-28 03:20:09,006:        ================================ Packet 6 ================================
]   2022-04-28 03:20:09,006:        page: 0 channel 21
]   2022-04-28 03:20:09,006:        get(allPackets)
]   2022-04-28 03:20:09,006:        [[1,4] is_permit_join_request (tp-additional-5.py:67)]
]   2022-04-28 03:20:09,006:        ============================== [1,4] is_permit_join_request (tp-additional-5.py:67) ======
]   2022-04-28 03:20:09,006:        checking until condition
ity]        2022-04-28 03:20:09,006:        = = = = = = = = = = is_permit_join_request = = = = = = = = = =
ity]        2022-04-28 03:20:09,006:        args:
es] 2022-04-28 03:20:09,006:        checking item 'zbee_aps.zdp_cluster', attr 'value', itemNum '0'
es] 2022-04-28 03:20:09,006:        item value: field zbee_aps.zdp_cluster 3600
es] 2022-04-28 03:20:09,006:        is_exactly_match(3600, 3600)
es] 2022-04-28 03:20:09,006:        item matched
]   2022-04-28 03:20:09,006:        is_permit_join_request returns True
es] 2022-04-28 03:20:09,006:        get_field(name=frame.number, attr=show, itemNum=0)
ity]        2022-04-28 03:20:09,007:        # # # # # # # # # # get_dst_pan_id # # # # # # # # # #
ity]        2022-04-28 03:20:09,007:        args: TDSK(pan_id)
es] 2022-04-28 03:20:09,007:        get_field(name=wpan.dst_pan, attr=value, itemNum=0)
ity]        2022-04-28 03:20:09,007:        ==================== check args types for set_var(validation_entity.py:68) =======
ity]        2022-04-28 03:20:09,007:        ---------- argument 'varName' ----------
ity]        2022-04-28 03:20:09,007:        Argument is TDSK, check is postponed until runtime
ity]        2022-04-28 03:20:09,007:        correct arg
ity]        2022-04-28 03:20:09,007:        ---------- argument 'value' ----------
ity]        2022-04-28 03:20:09,007:        parameter type should be 'typing.Any'
ity]        2022-04-28 03:20:09,007:        type matched
ity]        2022-04-28 03:20:09,007:        correct arg
ity]        2022-04-28 03:20:09,007:        ==================== check done for set_var ====================
ity]        2022-04-28 03:20:09,007:        # # # # # # # # # set_var # # # # # # # # # #
ity]        2022-04-28 03:20:09,007:        args: TDSK(pan_id), d6b5
]   2022-04-28 03:20:09,008:        ================================ End of Packet 6 ================================
]   2022-04-28 03:20:09,008:        __getitem__(allPackets)
]   2022-04-28 03:20:09,008:        __getitem__(allPackets)
ource]      2022-04-28 03:20:09,462:        received a packet, page: 0, channel: 21
]   2022-04-28 03:20:09,462:        ================================ Packet 7 ================================
]   2022-04-28 03:20:09,462:        page: 0 channel 21
]   2022-04-28 03:20:09,462:        get(allPackets)
]   2022-04-28 03:20:09,462:        [[1,5] or_predicate (tp-additional-5.py:70)]
```

Рис. 2. Пример лог-файла, который записывается в ходе выполнения валидационного скрипта

Поскольку запуск тестов может включать в себя несколько тестов, то лог-файл соответствующего теста кладется в папку с названием теста. Файловая структура, которая создается в ходе выполнения валидации тестов, показана на рис. 3.

Рис. 3. Файловая структура для хранения лог-файлов при валидации

2. Сбор требований к инструментарию

Для разработки инструментария были опрошены инженеры, работающие с этой системой, и сформулированы главные критерии:

– Графический интерфейс

– Компактное представление лог-файла валидации

Лог-файл как результат валидации может включать в себя около 100000 строк, в котором искать нужный пакет и проверку бывает сложным.

После мозгового штурма были определены не такие важные, но тем не менее полезные функции, а именно:

– Автоматическое редактирование списка тестов в конфигурационном файле для их запуска.

– Представление результатов прогона тестов в виде списка тестов и их статуса.

– Открытие pcap файла, который соответствует тесту, в программе Wireshark.

– Запуск тестов с помощью инструментария.

3. Разработка инструментария

Для реализации был выбран язык Python [3], поскольку разработанная система написана именно на нем. В качестве среды

разработки была выбрана среда PyCharm Community, для реализации графического интерфейса была выбрана библиотека PyQT5 [4].

На рис.4 можно увидеть основное окно, которое обеспечивает следующий функционал:

– загрузка папку с результатами тестирования и смотреть результаты выполнения тестового прогона;

– редактирование конфигурационный файл;

– выбор с помощью проводника валидационных скриптов и автоматическое редактирование набора тестов в конфигурационном файле;

– запуск прогона теста;

– открытие окна с деталями результатов тестирования конкретного теста;

– открытие папки с лог-файлами в проводнике.

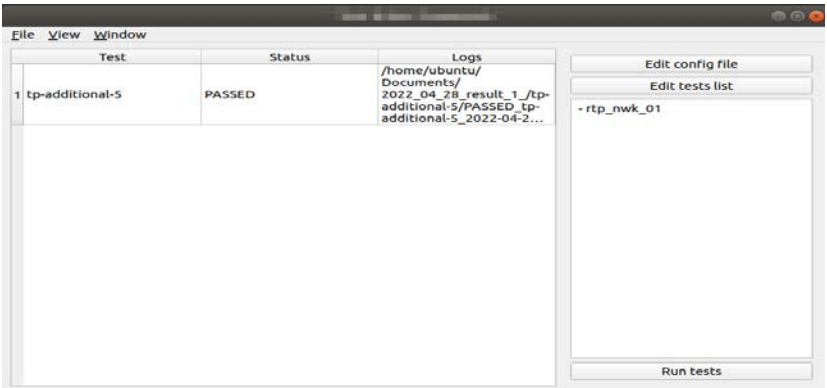


Рис. 4. Основное окно инструментария

На рис. 5 представлено окно, которое представляет лог-файл в компактном виде.

Было решено, что для представления лог-файла наиболее подходящим решением можно считать древовидную структуру. Для реализации решения был использован виджет QTreeWidget [4].

При клике на кнопку «Open PCAP» открывается программа Wireshark с соответствующим для теста PCAP файлом. При клике на кнопку «Open log» открывается окно файлового проводника, которое позволяет выбрать лог-файл. При клике на кнопку «Open test script» открывается окно, в котором показан валидационный скрипт.

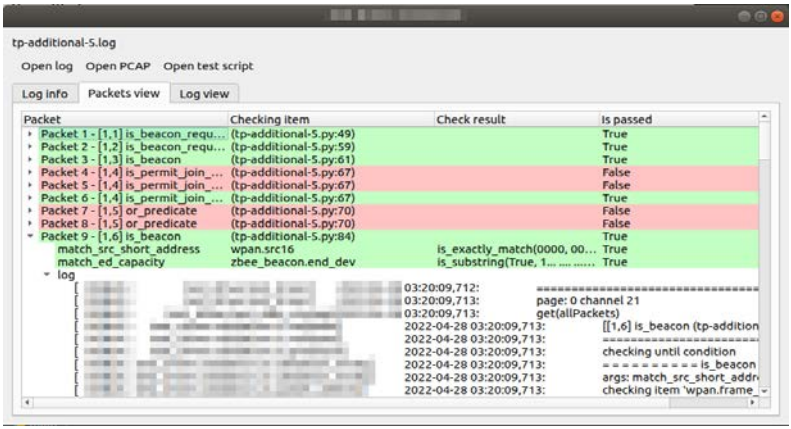


Рис. 5. Окно для просмотра лог-файла в компактном виде

Выводы

В ходе работы был разработан инструментарий к системе автоматизированного тестирования сетевого стека беспроводных устройств.

Разработанный инструментарий включает в себя две основные функции: графический интерфейс и компактное представление лог-файла как результата работы валидационного скрипта. Помимо основного функционала было реализовано автоматическое редактирование конфигурационного файла с помощью выбора валидационных скриптов из окна проводника, открытие программы Wireshark с соответствующим файлом трафика, компактное представление результатов тестирования нескольких приложений.

В дальнейшем планируется усовершенствовать алгоритм построения дерева лог-файла.

Список литературы

1. Zigbee Wireless Networking / Drew Gislason // Newnes. – 2008. – P. 2-44

2. Изучение программы Wireshark [Электронный ресурс] : база данных. – Режим доступа : https://www.wireshark.org/#learnWS

3. Документация по языку программированию Python [Электронный ресурс] : база данных. – Режим доступа : https://www.python.org/

4. Документация по библиотеке PyQT5 [Электронный ресурс] : база данных. – Режим доступа : https://doc.qt.io/qtforpython/

Бинарная сериализация на языке Java

Е. А. Кольченко

Студент магистратуры

В. С. Тарасов

Ассистент

Введение

В настоящее время современные клиент-серверные решения используют Json формат данных для передачи сообщений. Главными достоинствами такого формата является более простая отладка взаимодействия (вследствие их читаемости), простая реализация (наличие готовых парсеров) и их универсальность. Однако, очевидно, что такой формат является крайне избыточным, мизерная доля полезной информации находится в массивной, неэффективной обёртке. В следствии этого объем данных и время её передачи становится так же крайне избыточным, что негативно сказывается на скорости работы программы. При передаче любой числовой информации приходится заниматься её конвертацией в стоковое представления и обратно.

1. Постановка задачи

Главная идея – это отправлять бинарные данные в готовом виде, что позволит избавиться от избыточности данных JSON формата и избавиться от конвертации данных. Это даст прирост к скорости передачи и к скорости последующей обработки этих данных. Данная работа посвящена разработке и программной реализации на языке Java комплекса алгоритмов для сериализации и десериализации примитивов и объектов бинарное представление [1].

2. Сериализация примитивных типов данных

Для реализации данной задачи будем использовать библиотеку Guava от компании Google, она позволяет реализовывать порядок чтения и записи байт от младшего к старшему (little-endian) и имеет хорошие показатели в скорости работы [2]. Классы LittleEndianDataInputStream и LittleEndianDataOutputStream из пакета com.google.common.io реализуют интерфейсы DataInput и DataOutput, которые отвечают за чтение и запись примитивных типов в Java.

Однако, чтобы реализовывать свою логику чтения и записи примитивов и добавлять методы для работы с более сложными объектами и структурами данных, необходимо между 2-мя этими классами добавить ещё один уровень абстракции: BinaryDataInput и BinaryDataOutput классы, в которых и будет происходить вся дальнейшая работа. Как известно в Java есть 8 примитивных типов, которые делятся на 4 группы:

- Целые числа – byte (8 бит), short (16 бит), int (32 бита), long (64).
- Числа с плавающей точкой – float (32 бита), double (64 бита).
- Логический – boolean (8 бит).
- Символьный – char (16 бит).

При этом, в реальных задачах очень редко числа достигают своих пограничных значений, в результате этого основная часть байт заполнена 0, и лишь меньшая доля несет в себе полезную информацию. Так как основная цель работы — это создание инструмента для уменьшения объема данных, то передавать пустые байты нецелесообразно [3]. Поэтому был разработан алгоритм, который упаковывает только нужные байты, и затем корректно распаковывает и преобразовывает до нужного типа данных.

3. Упаковка данных

Так как в Java 1 байт имеет диапазон от -128 до 127. Но для начала нужно проверить, если число меньше 127 и больше или равно 0, то значит можно передать всего один байт. Но предварительно мы инвертируем старший бит применив дизъюнкцию на маску 0x80, чтобы при чтении понимать, что значение занимаем всего один байт. Если же значение выходит за рамки этого диапазона, то тогда нужно посчитать количество байт, которое занимает это значение, записать получившейся число, а затем побайтно разбить и записать уже само значение. Создадим переменную i, которая будет является текущим номером байта, который был уже записан. Затем после конъюнкции на маску 0xFF, получим то количество бит, на которые надо сместить влево исходное значение. Затем приведя полученное значение к типу byte, мы обрежем первые 8 бит и запишем их в буфер. Для того, чтобы правильно посчитать количество необходимых байт будем разбирать число со старшего байта, накладывая маску 0xFF, смещенную на количество бит текущего байта. Затем проверять, если полученное число рано 0, то двигаться дальше к младшему байту, если же не равно 0, значит дальше идут уже полезные байты и цикл можно останавливать (рис. 1, рис. 2).

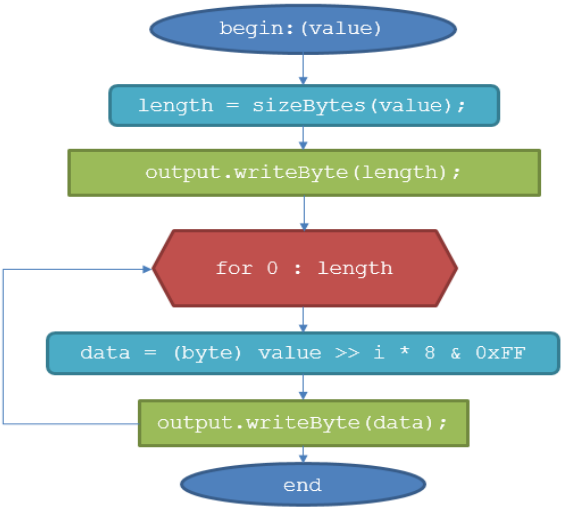


Рис. 1. Упаковка значения

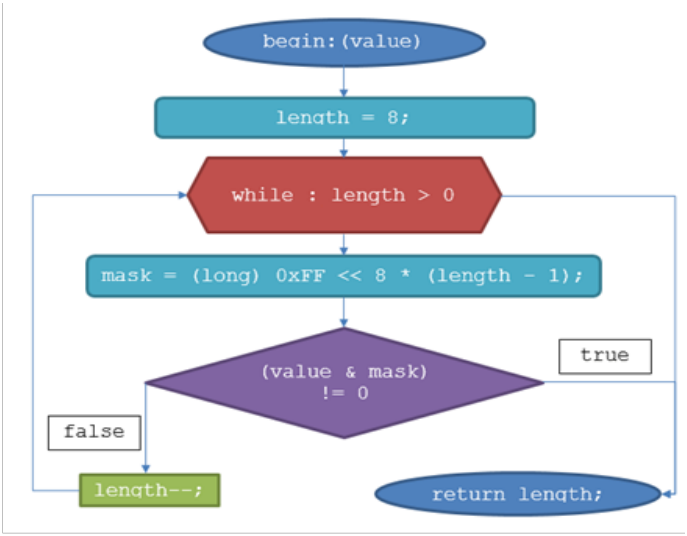


Рис. 2. Подсчет количества байт

4. Распаковка данных

После того, как данные были запакованы, необходимо уметь их корректно считывать, а значит обратным алгоритмом их нужно распаковать. Первый байт, который мы считаем может быть либо уже самим значением из диапазона [0, 128) с инвертированным старшим битом, либо же количеством байт, которые надо считать и собрать в одно значение. Чтобы проверить является ли считанный байт всего одним значением, нужно произвести конъюнкцию с маской 0x80. Если после этого число стало равным 128, то значит, чтобы получить значение осталось инвертировать старший бит, наложив маску 0x7F (число 127). Если же этого не произошло, значит считанный байт является количеством байт, занимаемых значением. Затем мы в цикле считываем все байты, расширяя их до типа Long и вставляем в соответствующие места результирующего числа. Полный алгоритм представлен ниже (рис. 3).

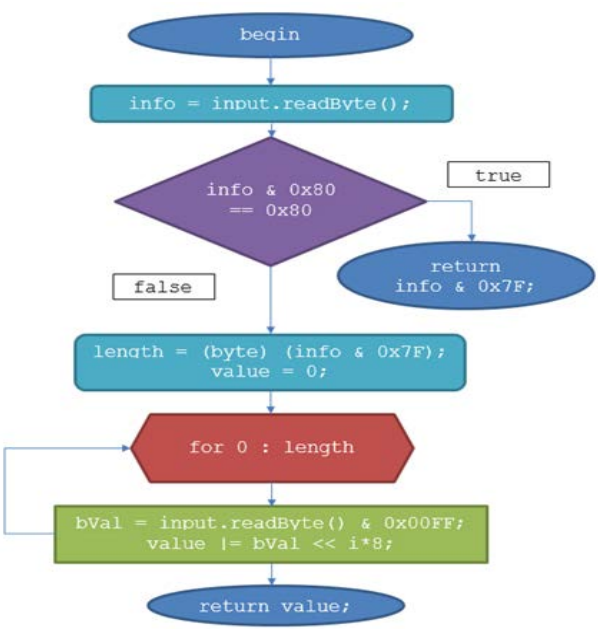


Рис. 3. Распаковка значений

Стоит упомянуть, что отрицательные числа будут записывать по такому алгоритму в полном их размере, так как их старшие биты будут инвертированы в единицу. Однако в реальных прикладных задачах и

бизнес-приложениях отрицательные числа редко передаются от клиента к серверу и обратно, так как в реальной жизни мы в основном оперируем только областью натуральных чисел N.

5. Сериализация объектов

После того, как сериализации и десериализация примитивных типов данных была реализована, можно усложнять текущий сериализатор. Необходимо добавить возможность сериализовать готовые объекты классов. Все классы, которые мы хотим сериализовать, должны реализовывать интерфейс BinarySerializable, в котором будет определен метод serialize с входящим параметром BinaryDataOutput output (листинг 1).

Листинг 1

```
@FunctionalInterface
interface BinarySerializable extends Serializable {
  void serialize (BinaryDataOutput output) throws IOException;
}
```

Этот интерфейс должен быть наследником маркера Serializable, чтобы JVM понимала, что объект, реализующий его, может быть сериализован. Далее в этом методе мы сериализуем все поля класса по порядку, вызывая соответствующие методы у объекта output (листинг 2).

Листинг 2

```
@Override
  public void serialize (BinaryDataOutput output) throws
IOException {
    output.writeInt(valueInt);
    output.writeLong(valueLong);
    output.writeFloat(valueFloat);
  }
```

Сам метод serialize будет вызываться в методе writeObject класса BinaryDataOuput. Метод будет использовать Generic type, так как мы заранее не знаем какой тип данных будет использоваться. Однако, мы знаем, что он гарантированно должен реализовывать внутри себя метод serialize. Внутри метода writeObject мы его вызываем, передав туда контекст класса BinaryDataOutput (листинг 3).

Листинг 3

```
public <T extends BinarySerializable> void writeObject(T obj)
throws IOException {
    obj.serialize(this);
}
```

6. Десериализация объектов

Разберем теперь десериализацию объектов. Для этого будем использовать конструктор с параметром BinaryDataInput input и вызывать его рефлексивно при чтении. Конструктор для примера описан ниже (листинг 4).

Листинг 4

```
public TestClassA (BinaryDataInput input) throws IOException {
    this.vInt = input.readInt();
    this.vLong = input.readLong();
    this.vFloat = input.readFloat();
  }
```

Важно то, что порядок чтения и записи полей должен быть обязательно одинаковым, иначе можно считать не те данные и полностью нарушить весь протокол, так как могут возникнуть смешения в массиве байт. Далее нужно вызвать конструктор рефлексивно, поэтому добавим метод readObject в BinaryDataInput с параметрами класса, который мы хотим считать и набором параметром для конструктора (листинг 5).

Листинг 5

```
public <T> T readObject(Class<T> type,
List<ConstructorParameter> parameters) throws IOException {
    final List<ConstructorParameter> params = new ArrayList<>
(parameters);
    params.add(new ConstructorParameter(BinaryDataInput.class,
this));
    return instance(type, params)
        .flatMapFailure(err -> instance(type, parameters))
        .orElseThrow(IOException::new);
  }
```

Если же класс является финальным, то нужно сначала попробовать сделать instance с теми параметрами, которые мы указали изначально, а затем в случае ошибки сделать instance с объектом класса BinaryDataInput. Класс ConstructorParameter, описывает тип и объект параметра для конструктора. В методе instance мы рефлексивно пытаемся найти и вызвать конструктор, если же произошла ошибка, то возвращаем объект Result.fail с описанием ошибки (листинг 6).

Листинг 6

```
private <T> Result<T> instance (Class<T> type,
List<ConstructorParameter> params) {
    try {
      final T instance = type
```

Окончание листинга 6

```
.getConstructor (array (params, param ->
param.type,Class[]::new)).newInstance (array (params, param ->
param.value, Object[]::new));
      return Result.success(instance);
    } catch (InstantiationException | IllegalAccessException |
      InvocationTargetException | NoSuchMethodException e)   {
     return Result.failure(e);}
  }
```

Если поиск, вызов конструктора и сериализация всех его полей произошел успешно, то возвращаем Result.success с готовым объектом.

7. Тестирование разработанного программного модуля

Предварительное тестирование разработанного программного модуля производилось с использованием фреймворка JUnit. Для демонстрации будем работать с тестовым классом (листинг 7) с полями разных типов, и построим график результатов размера объекта в разных форматах представления (рис. 4).

Листинг 7

```
public final class TestClazz implements BinarySerializable {
  private final short shortField;
  private final int intField;
  private final long longField;
  private final float floatField;
  private final double doubleField;

  public TestClazz(BinaryDataInput input) throws IOException {
    this.shortField = input.readShort();
    this.intField = input.readInt();
    this.longField = input.readLong();
    this.floatField = input.readFloat();
    this.doubleField = input.readDouble();
  }

@Override
  public void serialize(BinaryDataOutput output) throws
IOException {
    output.writeShort(shortField);
    output.writeInt(intField);
    output.writeLong(longField);
    output.writeFloat(floatField);
    output.writeDouble(doubleField);
  }
}
```

Далее создадим объект этого класса с значениями разных типов данных (листинг 8).

Листинг 8

```
var obj = new TestClass (128, 12543, 9223357L, 1e10f, 3.4e38);
```

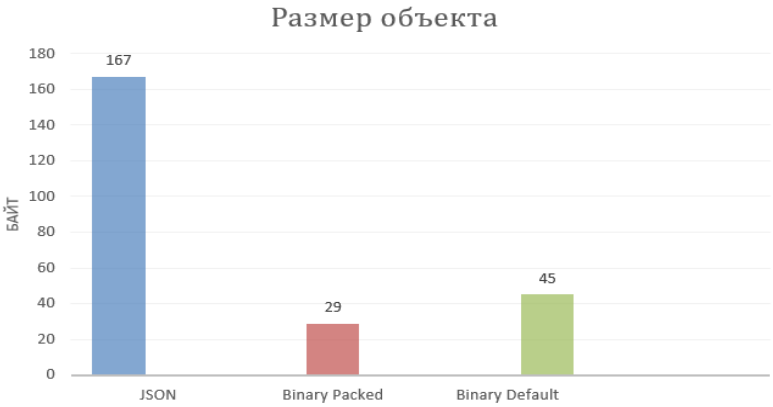


Рис. 4. Размер объекта.

Заключение

Данная статья посвящена разработке и реализации программного модуля для сериализации и десериализации объектов и примитивных типов данных в бинарное представление. Описаны алгоритмы для упаковки и распаковки данных. В итоге данной разработки получился полностью рабочий программный модуль, реализованный в виде библиотеки и тестового приложения. Использование данного решения позволит заменить json формат передачи данных на бинарный, что существенно уменьшит размер передаваемых данных и увеличит скорость их обработки. В дальнейшем предполагается расширение данного модуля для сериализации и десериализации массивов и коллекций различных типов данных.

Список литературы

1. JavaScript Object Notation [Электронный ресурс]: официальная документация. – Режим доступа: https://json.org/json-en

2. Java Development Kit 11 [Электронный ресурс]: официальная документация. – Режим доступа: https://docs.oracle.com/en/java/javase/11

3. Бурдинский, И. Н. Системы счисления и арифметика ЭВМ: учеб. пособие / И. Н. Бурдинский. – Хабаровск: Тихоокеан. гос. ун-та, 2008. – 79 с.

Алгоритм переноса информации о деталях геометрии в текстуры для низкополигональных моделей с применением метода трассировки лучей

Д. А. Королёв

Студент магистратуры

Д. И. Соломатин

Старший преподаватель

Введение

Для создания реалистичной 3d-модели, используют тысячи и даже миллионы полигонов. При этом виртуальные миры игр зачастую содержат тысячи подобных объектов. Системы с низкой производительностью, в частности многие мобильные устройства, с нужной скоростью визуализировать такие сцены не способны.

Для уменьшения размерности задачи с целью сохранения визуальной схожести конечного результата в 3d-графике активно применяется подход, связанный с проецированием информации о поверхности высокополигональной модели на низкополигональную (запекание).

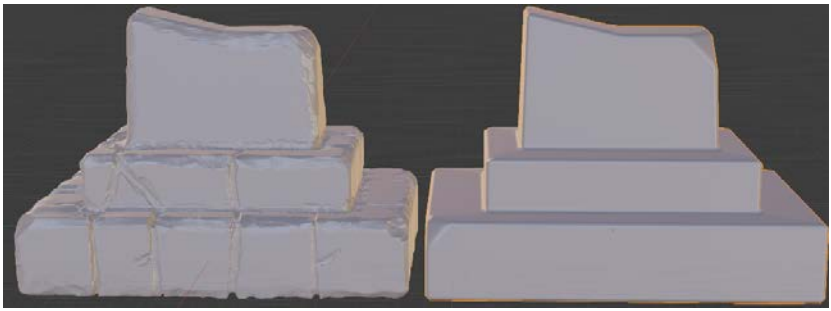
В данной работе рассматривается алгоритм такого проецирования с применением метода трассировки лучей, реализованный с применением API библиотеки OpenGL.

1. Постановка задачи

Входными данными для алгоритма выступают две трёхмерные модели, одна из которых – высополигональная, вторая – низкополигональная. Пример таких моделей приведен на рис. 1.

Результатом работы алгоритма является изображение, которое содержит информацию о нормалях высокополигональной модели относительно низкополигональной – текстура нормалей. Пример такого изображения приведен на рис. 2.

При рендеринге низкополигональной модели с учетом наложенной текстуры нормалей получается результат, внешне схожий с результатом визуализации исходной высокополигональной модели. Сравнительный результат с учетом освещения приведен на рис. 3.



слева – высокополигональная модель, справа – низкополигональная

Рис. 1. Модели

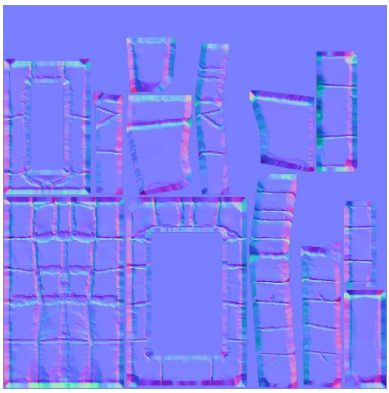


Рис. 2. Изображение с информацией о нормалях высокополигональной модели (текстура нормалей)

Визуализация высокополигональной модели (слева) и низкополигональной с наложенной текстурой нормалей (справа)

Рис. 3. Визуализация моделей

Как можно заметить результаты визуализации очень похожи, однако количество полигонов в низкополигональной модели во много раз меньше.

2. Текстура нормалей

Текстура нормалей [1] – это изображение, в котором зашифрована информация о векторах нормалей в каждой точке поверхности. Вектора находятся в касательном пространстве или tangent space. Длина вектора нормали равна единице. Соответственно, в векторном пространстве компоненты нормали находятся в диапазоне [-1; 1], кодируются переменными в 4 байта с плавающей запятой; таким образом, вектор нормали занимает 12 байт в памяти. Для хранения векторной информации в изображении необходимо перевести компоненты нормали в компоненты изображения и нормировать в положительные градиенты серого, например в 8 бит. При этом точность, естественно, сильно падает, но ее хватает для данной задачи. Для кодирования информации о нормали в точке RGB-изображения каждой компоненте цвета сопоставляется соответствующая координата вектора нормали: X → R, Y → G, Z → B.

Формула пересчёта векторных нормированных компонент в положительные компоненты изображения осуществляются по формуле:

$$RGB_{пиксель} = (1 + XYZ_{координата}) * 128 .$$

3. Касательное пространство

Стоит подчеркнуть, что вектора нормалей заданы в касательном пространстве. Касательное пространство – это локальная система координат для плоскости треугольника (полигона). Чаще всего вектора нормали направлены в направление Z+ или в сторону наблюдателя, в независимости от направления нормали треугольника. Из-за этой особенности карты нормалей имеют ярко выраженный синий оттенок.

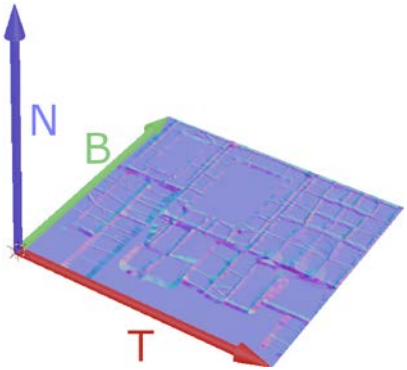
Для наложения карты нормали на 3d-модель необходимо для каждого треугольника рассчитать матрицу перехода в касательное пространство. Для этого необходимо рассчитать нормаль, тангент и битангент. С помощью трёх вектором можно получить матрицу перехода в касательное пространство, такая матрица называется TBN [2].

4. Запекание текстур

Для переноса информации с высокорполигональной модели на низкополигональную необходимо спроецировать точки с поверхности низкополигональной модели в направлении высокополигональной модели. Для этого потребуются текстурные координаты, которые будут

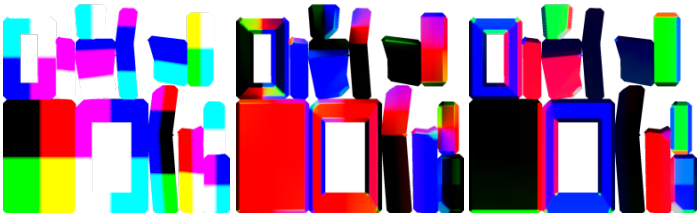
использованы для растеризации низкополигональной модели в пространство текстур или плоское представление 3д модели.

Так как, запекание текстур [2] происходит с использованием OpenGL, треугольники низкополигональной модели растрируются в пространство текстур, как показано на рис. 4. При этом используется аппаратная интерполяция для расчёта промежуточных значений между вершинами треугольника. Таким образом, растрируются координаты позиции, касательные координаты и нормали поверхности. Вся информация нормирована в диапазоне [-1, 1], из-за этого на изображениях на рис. 5 содержат чёрные области.



N – это нормаль треугольника, T – это тангент, B – это битангент

Рис. 4. Базис касательного пространства



а б в

а – позиция треугольников в трёхмерном пространстве, б – касательное пространство и в – нормаль

Рис. 5. Результат растрирования треугольников низкополигональной модели

На следующем шаге алгоритма, используя полученную на предыдущем шаге информацию, создаются лучи, для которых ищутся пересечения с высокополигональной моделью. При этом для одного луча может быть несколько пересечений, из которых нужно выбрать ближайшее к точке начала луча. Луч имеет всего две компоненты: позиция и направление, в качестве направления берётся нормаль треугольника.

5. Оптимизация трассировки лучей

Перебор каждого треугольника на наличие пересечения с лучом крайне затратно, учитывая огромное количество лучей и треугольников. Для решения этой проблемы применялись SAH BVH [3] деревья.

BVH (Bounding Volume Hierarchy) – иерархия ограничивающих объемов – подход к индексации трёхмерного пространства. На рис. 6 показан пример BVH-дерева.

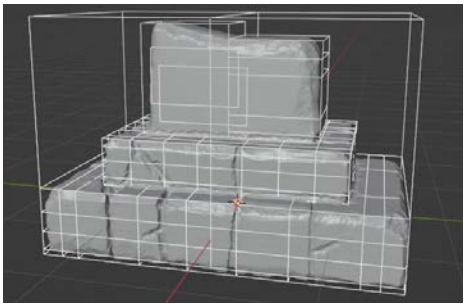


Рис. 6. Пример BVH для трёхмерной модели

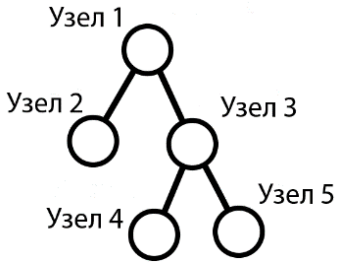
Для быстрого построения BVH дерева применяются SSE инструкции для векторных операция. SAH – это эвристика площадей поверхности, т.е. оптимизация для построения более качественных BVH-деревьев за счёт большего времени построения самого дерева.

Скорость трассировки лучей сильно зависит от качества BVH-дерева, однако необходимо соблюдать баланс между качеством дерева и скоростью его построения. Например, для запекания карты нормалей в разрешении 4098x4098 необходимо 16.7 миллионов лучей, дерево низкого качества обеспечивает пропускную способность в 10^7 мл/с (миллионов лучей в секунду), дерево низкого качества строится за 2 секунды, получается, что суммарное время запекания карты нормалей займёт ≈5 секунд. Дерево высокого качества имеет пропускную способность в $5*10^7$ мл/с, но при этом построение такого дерева займёт 30 секунд. В запекании текстур часто встречаются случаи, когда в

приоритете не скорость трассировки лучей, а скорость построения структур акселерации.

6. Упаковка и загрузка структуры акселерации в графический ускоритель

BVH-дерево обычно строится как односвязный список узлов, где каждый узел ссылается на узлы, находящиеся ниже, как на рис. 7.



Узел 1 ссылается на узел 2 и 3. Узел 3 ссылается на узел 4 и 5

Рис. 7. Пример дерева

Дерево в таком случае хранится в памяти не линейно, а кучей небольших участков. Для отправки дерева в память видеокарты необходимо скопировать дерево в линейную область памяти, в массив, а ссылки или указатели на другие узлы заменить индексом этого узла в линейном массиве. Предварительно необходимо выровнять структуру узла в памяти, данные должны быть кратны 16 байтам.

Загрузить большой объём данных в видеопамять по спецификации можно тремя способами:

1. Текстура с 4 компонентами по 32 бита каждая, формат GL_RGBA32F. Подобный буфер доступен на довольно старом оборудование поддерживающем спецификацию OpenGl 2.0 и старше. Текстура обладает кэшем, что позволяет уменьшить количество чтений из памяти видеокарты при обходе дерева. Главный минус – нелинейная адресация к данным, необходимо рассчитывать координаты u и v для получения значений.

2. Текстурный буфер. Так же, как и обычная текстура, кеширует данные, но адресация к памяти линейная. Текстурные буферы доступны на оборудовании, поддерживающим спецификацию OpenGL 3.1.

3. Шейдерный буфер. Имеет все плюсы текстурных буферов, однако обладает возможностью чтения в виде структуры. Чтение из текстуры и текстурного буфера осуществляется по 16 байт.

7. Трассировка лучей

Трассировка лучей [4] – это метод отправки лучей в определённом направлении с последующим пересечением луча с примитивом. Данный метод применяется для проецирования информации с высокополигональной модели на низкополигональную путём отправки лучей с поверхности треугольников низкополигональной модели в направлении нормали. В точке пересечения берётся нормаль высокополигональной модели и сохраняется как результат проецирования.

Заключение

После проецирование получаем текстуру, которую можно сохранить в файл. Эта текстура содержит информацию о нормали в локальном пространстве модели. Данная текстура может использоваться в любых игровых движках. Метод запекания текстур позволяет уменьшить нагрузку на видеоускоритель в среднем в 15 раз.

Список литературы

1. Learn OpenGL. Урок 5.5 – Normal Mapping [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/415579/

2. GPU Gems 3 [Электронный ресурс]. – Режим доступа: https://developer.nvidia.com/gpugems/gpugems3 Chapter 22

3. Modern C++17 header-only BVH library. [Электронный ресурс]. – Режим доступа : https://github.com/madmann91/bvh

4. Introduction to Ray Tracing: A Simple Method for Creating 3D Images. [Электронный ресурс]. – Режим доступа : https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/ray-tracing-practical-example

Математическое и программное обеспечение для оптимизации рекомендаций с использованием нейросетевого подхода

Д. А. Кравченко

Студент магистратуры

Е. Ю. Митрофанова

Доцент

Введение

В современной экономике огромную роль играет малое предпринимательство. Оно является наиболее «чувствительным» к потребностям потребителей, так как, кроме прибыли, не имеет других средств для саморазвития. С развитием информационных технологий стало появляться всё больше платформ для продвижения собственных услуг. Однако популярность предпринимателя и качество предоставляемых им услуг не всегда прямо пропорциональны из-за ряда причин: отсутствие клиентской базы, неадекватные цены, некорректная работа системы рекомендаций [1].

В связи с этим существует потребность в разработке механизма рекомендаций, который реализует необходимые для пользователя (клиента) возможности, а также учитывал их индивидуальные предпочтения. Существует ряд критериев оценки предпочтений клиента, с помощью которых можно построить предположение об услугах, которые могли бы его заинтересовать.

В данной статье рассматривается задача разработки программного обеспечения, реализующего механизм предсказания услуги на основе предпочтений пользователя с помощью LSTM – рекуррентной сети, способной к обучению долгосрочными зависимостями.

1. Алгоритм формирования рекомендаций

Система рекомендаций – это программное средство, которое, основываясь на данных о клиенте и услуге, делает предсказание об услуге, которая будет интересна пользователю. Данная система включает в себя весь процесс: от получения информации до её выдачи клиенту в виде рекомендации [2].

В настоящий момент можно выделить четыре основных типа рекомендационных систем:

– рекомендационные системы, базированные на контенте (Content base);

– коллаборативные рекомендационные системы (Collaboration);

– рекомендационные системы, базированные на знаниях (Knowledge base);

– гибридные рекомендационные системы (Gybrid) [3].

Принцип работы первого типа заключается в анализе услуги, составлении набора критериев (тип услуги, слова в названии и описании), определении критериев, которые нравятся клиенту, а далее сопоставлении этих данных и рекомендации.

В коллаборативных системах рекомендации рассчитываются на основе оценок других пользователей.

В системах, базированных на знаниях, рекомендации получаются на основе знаний, которые чаще всего добавляются вручную.

В гибридных рекомендационных системах используется несколько из выше перечисленных подходов [4].

Разработка данного программного обеспечения была основана на гибридном подходе, включающем в себя рекомендационные системы, основанные на контенте и коллаборативные. Данное сочетание подходов даёт рекомендации, основываясь и на критериях услуги, и на предпочтениях других клиентов, что в свою очередь, позволяет минимизировать субъективную составляющую в подборе услуги.

В качестве оценки точности предсказаний была выбрана одна из самых эффективных метрик RMSE – нормализованная средняя абсолютная ошибка:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}, e_i = origin_i - predict_i \quad (1)$$

Для реализации коллаборативного типа рекомендационных систем, в частности, определения сходства между клиентами был выбран коэффициент корреляции Пирсона. Данный алгоритм измеряет линейную зависимость между двумя клиентами как функцию их атрибутов (заказанные услуги, слова в поисковых запросах) [5].

$$r_{xy} = \frac{\sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m}(x_i - \bar{x})^2 \sum_{i}^{m}(y_i - \bar{y})^2}}, r_{xy} \in [-1,1] \quad (2)$$

2. Общий алгоритм работы системы рекомендаций

Для реализации механизма рекомендаций необходимо выполнить следующие действия:

– получить данные о клиентах (названия, описания и типы оказанных им услуг, поисковые запросы);

– обработать данные о клиентах с помощью коэффициента корреляции Пирсона, получить список схожих критериев;

– получить данные об услугах (названия, описания и их типы);

– получить рекомендацию на основании критериев схожих клиентов и данных об услугах с помощью LSTM;

– интерпретация ключевых слов, получение услуг на основании рекомендаций.

Общий алгоритм работы клиента с приложением для демонстрации системы рекомендаций представлен на рис. 1

Рис. 1. Общий алгоритм работы системы

3. Реализация программного средства

Разработанное программное средство является отдельным программным модулем. Данный модуль предназначен для улучшения системы рекомендаций клиент-серверного приложения, разработанного мною в рамках бакалаврской работы. Данное приложение каталогизирует организации, предоставляющие услуги, клиентов и

записи. В рамках его разработки было реализовано хранение данных в базе данных, система авторизации, возможность записаться на услугу, понятный интерфейс и быстрый поиск.

В качестве средств реализации используется:

– язык программирования Python;
– модуль для работы с искусственными нейронными сетями keras;
– модуль для работы с многомерными массивами - numpy;
– PostgreSQL система для работы с базами данных.

На рис. 2 представлена структура базы данных, на которой можно увидеть, что данные об услуге хранятся в таблицах Service и Interests, а информация о клиентах хранится в таблице User.

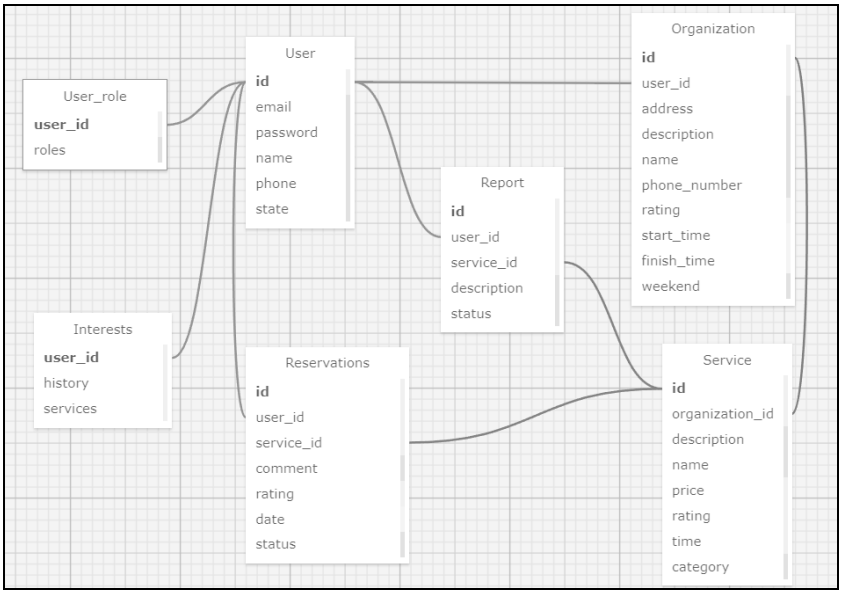


Рис. 2. Структура базы данных

Нейронная сеть в качестве датасета использует текст, состоящий из данных об услугах, представленных в виде текста. Слова выделяются в n1m с помощью токенизатора – готового класса для парсинга. Далее каждое из выделенных слов кодируется в соответствии с его номером в образованном словаре, после чего образуется двумерная матрица. Из этой матрицы формируется тензор обучающей выборки и набор выходных значений. После создаётся модель рекуррентной нейронной сети LSTM, которая обучается на датасете. Далее определяется список

критериев схожих юзеров и для него вместе с данными о клиенте (данные об услугах, которые были предоставлены клиенту и история поисковых запросов) формируется прогноз в виде текста.

Далее полученный текст интерпретируется в список услуг, которые будут рекомендованы пользователю после.

Для того, чтобы клиент мог увидеть список рекомендаций, необходимо успешно авторизоваться в приложении и войти на главную страницу. На рис. 3 представлено отображение списка рекомендаций. Как видно на изображении, рекомендации представлены в виде организаций, предоставляющих их с названием, адресом, рейтингом и кратким описанием.

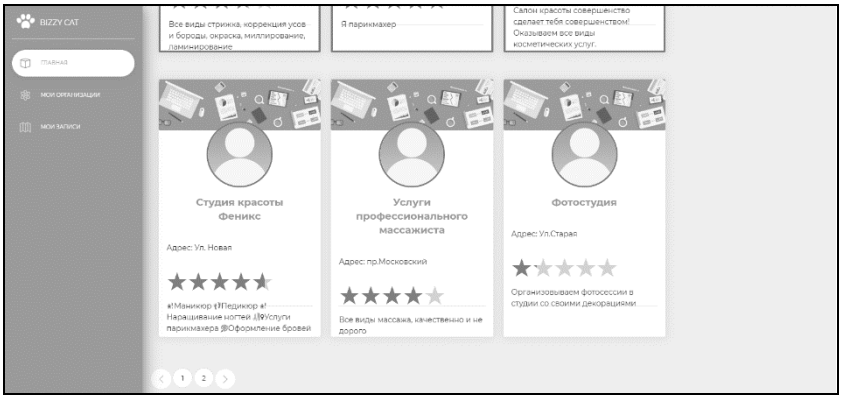


Рис. 3. Отображение рекомендованных услуг на главной странице

Заключение

В результате было разработано программное обеспечение, реализующее механизм предсказания услуги на основе предпочтений пользователя с помощью LSTM. В данной статье описан алгоритм формирования рекомендаций, основанный на гибридном подходе, включающем рекомендационные системы, базированные на контенте и коллаборативные. В итоге данной разработки получился полностью рабочий программный модуль, реализованный в виде подключаемого модуля. Использование данного решения позволяет усовершенствовать систему рекомендаций под индивидуальные предпочтения клиентов. В дальнейшем предполагается использование данного модуля в приложении для малого бизнеса в сфере рынка услуг, реализованного мною в рамках магистерской работы.

Список литературы

1. Алимова, Т. Диверсификация деятельности малых предприятий / Т. Алимова // Вопросы экономики. – СПб. : Питер, 1997. – С. 130-137.

2. Николенко, С. А. Рекомендательные системы / С.А. Николенко. – СПб. : Изд-во Центр Речевых Технологий, 2012. – 53 с.

3. Филиппов, С. А. Определение сходства информационных объектов на основе неявных пользовательских предпочтений в рекомендательных системах / С.А. Филиппов, В.Н. Захаров // Аналитика данных и управление в областях с интенсивным использованием данных. – 2016. – Вып. 12. – С. 169-174.

4. Recommending and Evaluating Choices in a Virtual Community of Use / W. Hill [et al.] // Proceeding Conference Human Factors in Computing Systems. – Canada, 1995. – С. 194-201.

5. Рекомендательные системы: You can (not) advise [Электронный ресурс]. – Режим доступа : https://habr.com/post/176549/

Разработка информационной системы учёта посещаемости факультета

В. К. Кушнеренко

Студент магистратуры

А. А. Вахтин

Доцент

Введение

На данный момент во многих учебных заведениях распространены различные документы по учету посещаемости. Они используют данные, которые каждую неделю отмечаются самими преподавателями или предоставляются старостами, что не способствует получению своевременной информации о посещаемости. Документы по ее учёту содержат информацию об отсутствии или присутствии человека на занятиях.

Проверка посещаемости требует немаленьких затрат: сопровождается значительным объёмом документов, отнимает много времени как у старост, так и у преподавателей.

Принимая во внимание вышеописанные факты, мы можем сформулировать ряд конкретных проблем:

– Необходимость затрат большого количества времени на заполнение документов;

– Неструктурированность документов по учету посещаемости;

– Старосты не всегда добросовестно заполняют посещаемость;

– Необходимо хранить все документы по учету в бумажном виде;

– Сложность в просмотре и ведении статистики по каждому студенту;

Отдельно рассмотрим учёт посещаемости через нашу систему.

Разработка системы для учета посещаемости позволит решить указанные проблемы процесса его проведения, который используется в настоящее время, сократить затраты времени на заполнение документов по учету, работать под разными ролями.

1. Постановка задачи

Целью данной работы является разработка системы учета посещаемости.

Конечными пользователями системы являются:

– Студенты, которые будут иметь возможность просматривать статистику по посещениям;

– Старосты, которые смогут заполнять посещаемость вместо преподавателей, если это будет необходимо;

– Преподаватели, которые смогут также следить за статистикой и заполнять данные;

– Администраторы, которые будут иметь возможность делать все действия в системе;

К разрабатываемому продукту предъявляются следующие требования:

– Удобный просмотр и редактирование учета посещаемости;

– Удобное взаимодействие с элементами системы;

– Работа под несколькими ролями в системе;

– Возможность работать с данными, связанными с посещаемостью;

Для создания качественного решения необходимо провести анализ существующих решений.

2. Анализ

2.1 Анализ предметной области

Процесс учета включает в себя множество действий, направленных на формирование документов о посещаемости студентов и сбора различной актуальной информации об этом.

При проверке посещаемости преподаватель получает список/журнал посещаемости. Затем он заходит в него и отмечает не пришедших на занятие студентов. После этого он может продолжать занятие. Имеется возможность для администратора изменять данные студентов, дисциплин, добавлять новые группы, занятия. Также можно провести поиск по конкретному студенту.

В данной работе будет проведён детальный разбор функционирования системы.

2.2 Анализ задачи

Рассмотрим задачу экспорта данных о посещаемости из файла. В настоящий момент имеются файлы с записями, общая структура которых представлена в таблице.

Таблица

Упрощенное представление списка студентов для посещаемости.

ФИО студента/Дата проведения занятия	01. 05. 02	02.05.02	03.05.02
Беркутов И. С.	+	-	+
Меркулов С. В.	-	+	+
Петров П. В.	+	+	+
Сидоров В. И.	+	-	-

В 1 ячейке хранится параметр, обозначающий то что находится в ячейках внизу (ФИО студентов) и справа (даты занятия), в остальных ячейках находятся отметки - плюс или минус в зависимости от того был ли студент на занятии или нет.

Соответственно, экспорт в PDF данных студентов по посещаемости должен быть разработан с учетом структуры файла с формализованными требованиями.

Теперь проанализируем задачу разработки системы.

Для работы в системе предусмотрены разные роли. Роли могут принимать следующие значения:

– Студент;
– Преподаватель;
– Староста;
– Администратор.

Отметки посещаемости могут иметь свои статусы:

– Плюс - студент был на занятии;
– Минус - студент не был на занятии.

Система должна упрощать процесс своевременного получения данных, а также отслеживание статистики посещаемости, с учетом текущих данных.

Существующие решения

При проведении анализа были рассмотрены следующие готовые решения по учету посещаемости, в скобках приведены ссылки на официальные сайты существующих аналогов:

– Moodle [1] Система позволяет не только отслеживать посещаемость, но и выставлять оценки за курсы, общаться с преподавателями онлайн. Внешний вид сайта, представляющего систему, представлен на рис. 1.

– iTeacherBook [2] - журнал преподавателя – рис. 2. Этот сайт позволяет хранить информацию о студентах, их посещаемости, оценках и других различных данных.

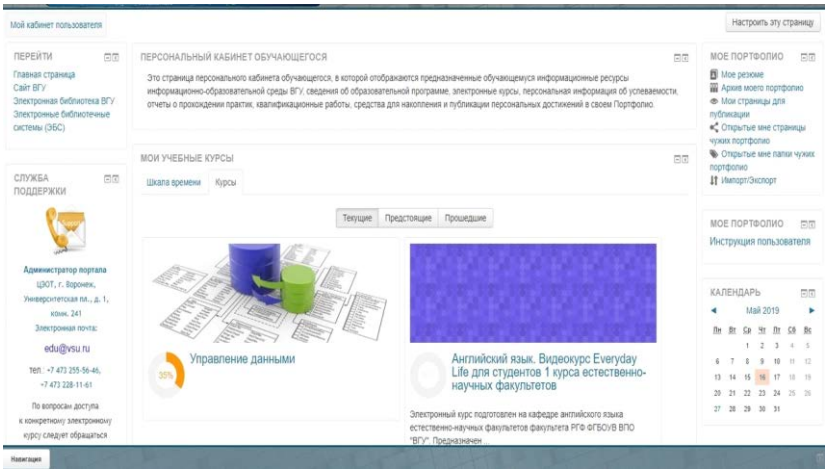


Рис. 1. Официальный сайт системы Moodle

Рис. 2. Официальный сайт системы iTeacherBook

3. Реализация

3.1 Средства реализации

Для реализации проекта были выбраны следующие технологии и программные средства:

– Язык программирования Python 3.8 - обладает простотой и функциональностью, широко используется для разработки веб-приложений.

– Библиотека Flask [3] из языка программирования Python – обеспечивает быстрое и легкое создание веб-приложений и различных необходимых для его функционирования модулей.

– Библиотека Flask-Login - для реализации авторизации и аутентификации в системе по ролям.

– Шаблонизатор Jinja [4] – для обработки данных, поступающих на HTML-страницу.

– Библиотека SQLAlchemy [5] - для создания прослойки в виде классов, связанных с БД.

– Система контроля версий Git – обеспечивает возможность сохранения состояний проекта, получения текущей версии проекта и извлечение старых версий. Более того, система позволяет отслеживать этапы разработки решения.

3.2 Реализация классов

В результате анализа было принято решение о разработке системы, описание которой приводится в данной главе.

Структура проекта имеет вид:

– Файл app.py – содержит в себе все маршруты веб-приложения. Также здесь идет печать в PDF и авторизация.

– Файл app_config.py – конфигурация и настройки всего проекта.

– Файл forms.py – предоставляет все формы, необходимые для функционирования приложения.

– Файл model.py – представляет собой модели БД в виде их классов.

– templates – представляет собой папку с шаблонами html-страничек.

3.3 Диаграмма вариантов использования

При разработке системы составлена диаграмма вариантов использования, изображенная на рис. 3.

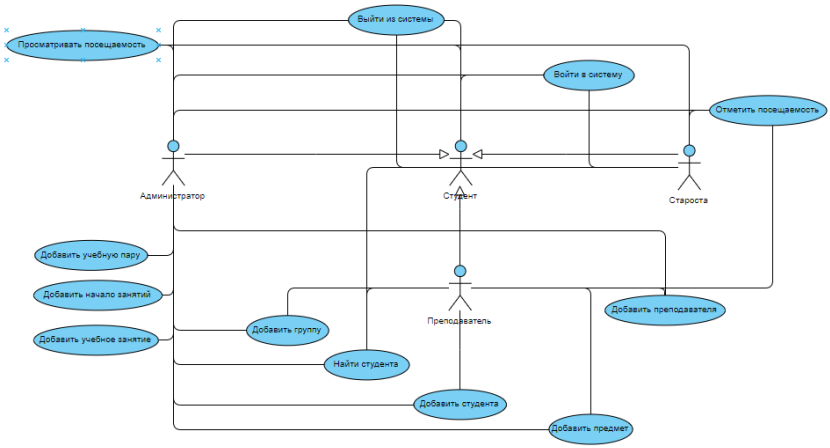


Рис. 3. Диаграмма вариантов использования

3.4 Диаграмма классов

При разработке системы составлена диаграмма классов, изображенная на рис. 4.

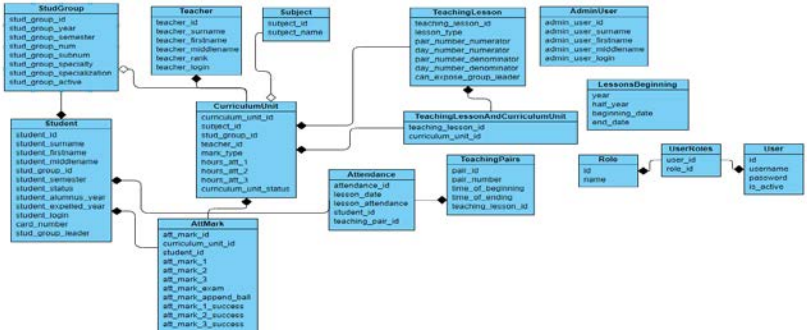


Рис. 4. Диаграмма классов для системы

3.5 Схема данных

При разработке системы составлена схема данных, изображенная на рис. 5.

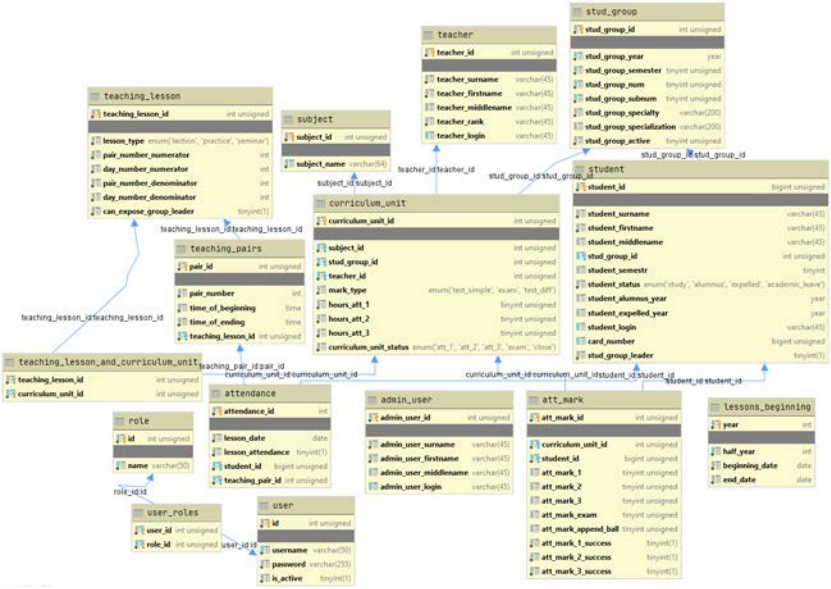


Рис. 5. Схема данных системы

Stud_group – таблица, содержащая группы студентов.

Student – таблица, содержащая в себе данные конкретного студента.

Teacher – таблица, содержащая данные о преподавателях в учебном заведении.

Curriculum_unit – таблица, содержащая единицы учебного плана.

Subject – таблица, содержащая сведения о предметах в учебном заведении.

Teaching_lesson – таблица, содержащая сведения о занятиях в учебном заведении.

Teaching_lesson_and_curriculum_unit – связующая таблица между единицами учебного плана и занятиями в учебном заведении.

Attendance – таблица, содержащая сведения о посещаемости студента.

Att_mark – таблица, содержащая в себе аттестационные результаты студента.

Teaching_pair – таблица, содержащая в себе учебные пары на определенные дни.

Lessons_beginning – вспомогательная таблица для учета по первой неделе числителя/знаменателя.

User – таблица, содержащая информацию о пользователях в системе.

Role – таблица, содержащая информацию о ролях пользователя.

User_roles – связующая таблица между пользователями и их ролями.

Заключение

В ходе выполнения работы были проанализированы существующие решения для учета посещаемости, в результате анализа был получен вывод, что ни одно из них полностью не соответствует требованиям и не обладает необходимым функционалом.

Результатом выполнения работы является набор инструментов и веб-страниц, позволяющий управлять учетом посещаемости и возможность просмотра данных о ней.

Список литературы

1. Moodle [Электронный ресурс]. – Режим доступа : https://edu.vsu.ru/

2. iTeacherBook [Электронный ресурс]. – Режим доступа : https://ipadstory.ru/iteacherbook-zhurnal-prepodavatelya.html

3. Документация по Flask [Электронный ресурс]. – Режим доступа : http://flask.pocoo.org/docs/1.0/

4. Документация по Jinja [Электронный ресурс]. – Режим доступа : http://jinja.pocoo.org/docs/2.10/

5. Документация по SQLAlchemy [Электронный ресурс]. – Режим доступа : https://www.sqlalchemy.org/

Применение глубоких нейронных сетей в задаче диагностики коронавирусной инфекции COVID-19

К. М. Матвеев

Студент магистратуры

А. А. Сирота

профессор

Введение

Актуальной проблемой на сегодняшний день остается текущая пандемия коронавирусной инфекции (COVID–19), вызванная коронавирусом SARS–CoV–2. Статистика числа заболевающих и летальных исходов неуклонно продолжает расти. По данным на январь 2021 года [1] в мире насчитывается порядка 95 миллионов случаев заболевания и чуть более 2-х миллионов летальных исходов, а Россия занимает 4-е место в мире по числу зараженных. За 2020 год одной только Всемирной Организацией Здравоохранения было опубликовано более 3000 исследований на тему COVID–19 [2]. Нет сомнений в чрезвычайной актуальности данной проблемы, требующей решения в минимальные сроки, а именно: поиска лучших методов лечения и диагностики данного заболевания, в том числе диагностики с использованием методов машинного обучения.

Так, в работе [3] описано успешное применение искусственных нейронных сетей для обнаружения рака легких, а работах [4, 5] – методы улучшения исходных изображений и сегментации легких на рентгенограммах грудной клетки с использованием глубокого обучения. В работе [6] была разработана трехмерная модель глубокого машинного обучения, названная COVID-SegNet, которая продемонстрировали возможность использования сверточных нейронных сетей для сегментации областей, пораженных вирусной инфекции на КТ изображениях грудной клетки. А в исследовании [7] было представлено сравнение производительности работы по распознаванию областей, пораженных вирусной инфекцией COVID-19 на КТ изображениях легких, двух моделей глубокого обучения UNet и SegNET.

Таким образом, целью данной работы является разработка алгоритма семантической сегментации КТ изображений органов

грудной клетки с использованием глубокого машинного обучения, а именно – использование классификатора для выявления степени поражения легких и нейронной сети для сегментации для выделения областей матового стекла.

1. Архитектура реализуемой системы

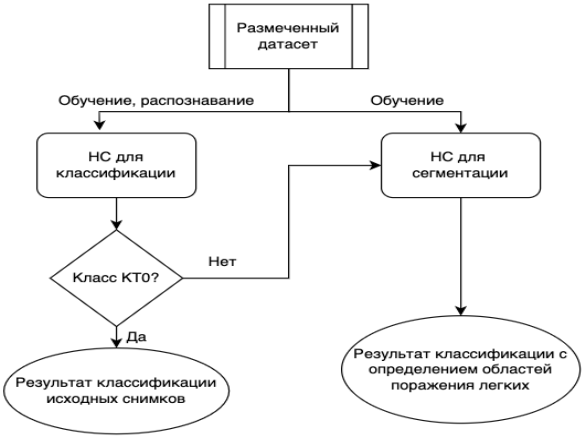


Рис. 1. Архитектура реализуемой сети

Из диаграммы (рис. 1) видно, что размеченный набор данных используется как для обучения классификатора, так и НС для сегментации. После обучения сетей, изображения поступают на вход классификатора в случае, если снимок был отнесен не к классу КТ0, который соответствует отсутствию поражения легких, то он подается на вход нейронной сети для сегментации, которая выделяет области матового стекла в легких.

В ходе исследования был рассмотрен ряд наборов данных, подходящих для обучения. Среди которых «COVID‑19 CT segmentation dataset» [8], набор данных [9], подготовленный на основе данных Китайского консорциума исследований КТ–изображений грудной клетки (China Consortium of Chest CT Image Investigation of CC-CCII) и «MosMedData COVID19_1110» [10], подготовленный центром радиологии Москвы.

В табл. 1 представлено краткое сравнение основных характеристик перечисленных наборов данных.

Таблица 1

Сравнение наборов данных

Наименование набора данных	**Количество снимков (общее)**	**Количество срезов**	**Количество классов**	**Формат изображений**
COVID-19 CT segmentation dataset	9	829	1	NIfTI
China Consortium of Chest CT Image Investigation	–	752	3	JPEG
MosMedData COVID19_1110	1110	35521	5	NIfTI

В итоге обучение как классификатора, так и нейронной сети для сегментации проводилось на наборе данных «MosMedData COVID19_1110». Так как он является наиболее цельным и массивным, а также размечен рентгенологами не только для обозначения областей матового стекла в легких, но и разбит на 5 классов в зависимости от степени поражения легких.

2. Решение задачи сегментации

В качестве сети для сегментации легких была выбрана модель U–Net, так как она демонстрирует хорошие результаты в задачах биомедицинской сегментации, а также не требует большого количества обучающих данных. В ходе эксперимента были реализованы две архитектуры, изображенные на рис. 2. Первый вариант – это классическая сеть, второй – сеть с добавленным dropout слоев для каждого блока со сверткой и использованием слоя UpSampling вместо слоя обратной свертки.

Сети обучалась на 2500 изображениях, тестировалась на 625 снимках. В качестве функции потерь применялась коэффицент Дайса со знаком минус. Обучение проводилось в течение 50 эпох. В качестве метрик качества использовались коэффицент Дайса и binary accuracy. На рис. 3 слева-направо расположены изображения – исходный снимок, маска, предсказание сети и наложение маски на снимок.

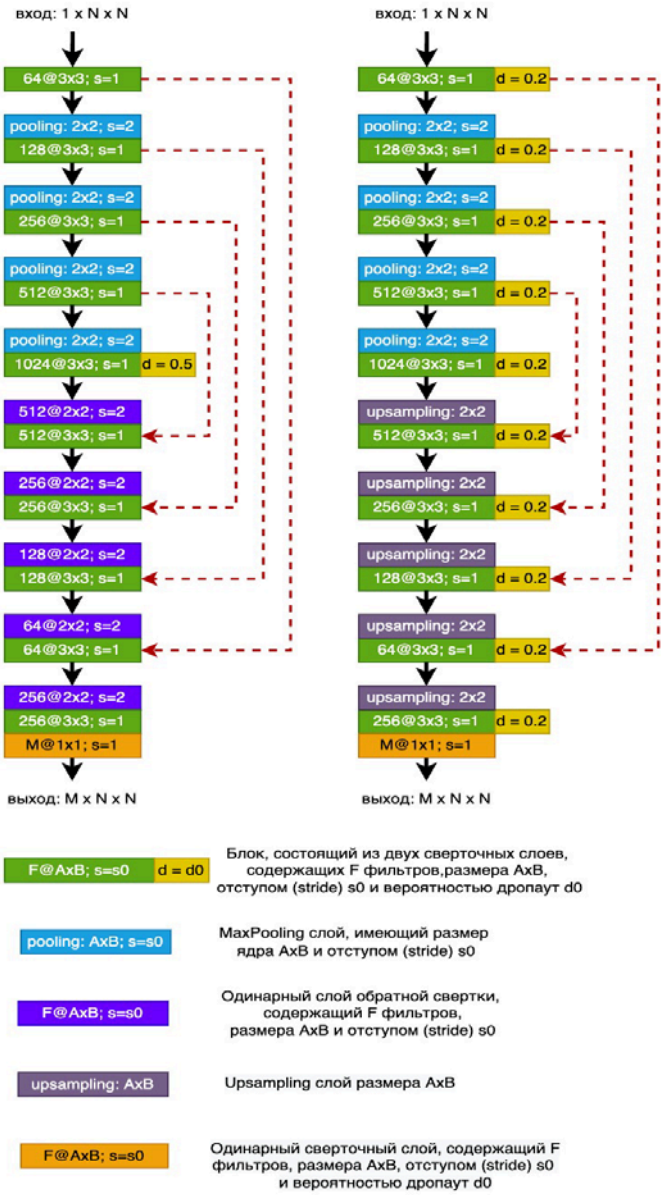


Рис. 2. Архитектуры нейронных сетей для сегментации

Рис. 3. Результаты предсказания сети для сегментации

В табл. 2 представлены результаты тестирования обученных моделей. В результате обучения удалось добиться величины коэффицента Дайса равной 0.85 для второй модели.

Таблица 2

Результаты тестирования сетей для сегментации

Сеть	Метрика		
	Функция потерь	Коэффициент Дайса	Binary accuracy
A (оригинальная U–Net)	-0.76	0.76	0.98
B (U–Net с dropout и UpSampling)	-0.85	0.85	0.99

3. Решение задачи классификации

За основу архитектуры реализованной сети для классификации была взята модель сети ResNet50. В ходе эксперимента были реализованы две архитектуры, изображенные на рис. 4. В первой модели использовалась обученная сеть на наборе данных ImageNet, далее ее выход соединялся с полносвязным слоем из 5 нейронов, вторая модель – это ансамбль из 10 обученных сетей ResNet50 с общим выходом из 5 нейронов.

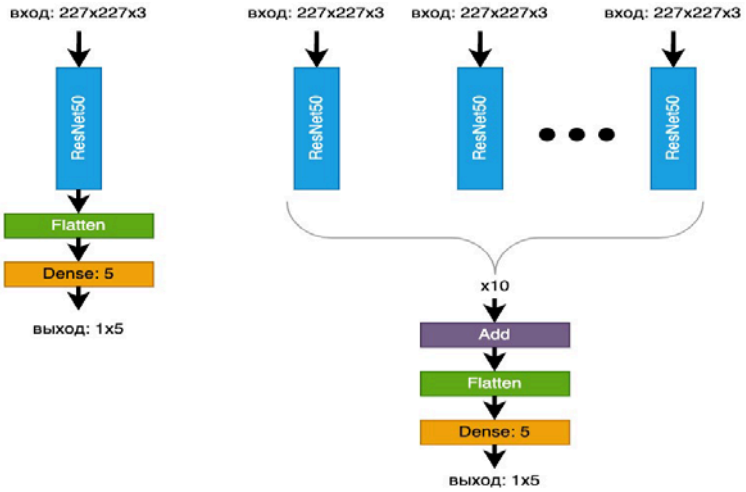


Рис. 4. Архитектуры нейронных сетей для классификации

При обучении в качестве функции потерь применялась функция sparse categorical crossentropy. Обучение проводилось в течение 50 эпох. Использовались такие метрики качества как точность, полнота и F1 метрика.

Таблица 3

Результаты тестирования сетей для классификации

Сеть	Метрика			
	Accuracy	Точность	Полнота	F1
ResNet50	0.90	0.74	0.98	0.84
Ансамбль ResNet50	0.92	0.87	0.99	0.92

В табл. 3 представлены результаты тестирования работы обоих нейронных сетей. Из нее видно, что модель, использующая ансамбль из 10 сетей ResNet50 показала лучшее качество распознавания.

Заключение

В ходе работы был разработан алгоритм семантической сегментации КТ изображений органов грудной клетки с применением двух моделей глубоких нейронных сетей. Данный подход может быть использован для облегчения работы врачей-рентгенологов, а также повышению точности диагностики заболевания коронавирусной инфекции COVID-19.

Список литературы

1. COVID-19 Dashboard by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (JHU) [Электронный ресурс]. – Режим доступа : https://coronavirus.jhu.edu/map.html

2. COVID-19 Studies from the World Health Organization Database [Электронный ресурс]. – Режим доступа : https://clinicaltrials.gov/ct2/who_table

3. Suzuki, K. Computer-aided detection of lung cancer in computed tomography scans: Review and future prospects / K. Suzuki // Image-based computerassisted radiation therapy. – Singapore, 2017. – P. 9-40.

4. Suzuki, K. Image-processing technique for suppressing ribs in chest radiographs by means of massive training artificial neural network (MTANN) / K. Suzuki // IEEE Transactions on medical imaging. – 2006. – T. 25. – Vol. 4. – P. 406-416.

5. Hsien-Huang P Wu. Patient information extraction in digitized X-ray imagery / H.H.P. Wu // Image and Vision Computing. – 2004. – T. 22. – Vol. 3. – P. 215-226.

6. Q. Yan COVID-19 Chest CT Image Segmentation - A Deep Convolutional Neural Network Solution. / Yan Q // ArXiv. – 2020. preprint arXiv:2004.10987

7. Saood, A. COVID-19 Lung CT Image Segmentation Using Deep Learning Methods: UNET Vs. SegNET. / Saood A., Hatem I. // Research Square preprint. – 2020. DOI: 10.21203/rs.3.rs-56882/v3

8. COVID-19 CT segmentation dataset [Электронный ресурс] : набор данных. – Режим доступа : http://medicalsegmentation.com/covid19/

9. CT images from CC-CCII [Электронный ресурс] : набор данных. – Режим доступа : http://ncov-ai.big.ac.cn/download

10. MosMedData: результаты исследования компьютерной томографии органов грудной клетки с признаками COVID-19, 2020 г., версия 1.0 [Электронный ресурс] : набор данных. – Режим доступа : https://mosmed.ai/datasets /covid191110/

Приближенные алгоритмы построения фазового пространства фон Неймана

С. С. Мелихов

Студент магистратуры

Е. А. Киселев

Доцент

Введение

В настоящее время проводятся интенсивные исследования в области квантовой статистики и технологий квантовых вычислений. Одним из очень важных понятий в этих вопросах выступает фазовое пространство. Первые попытки разработать соответствующий математический аппарат были предприняты еще в начале XX века И. фон Нейманом [1]. Достоинствами подхода, предложенного фон Нейманом, являются простота и наглядная физическая интерпретация. Идеи, лежащие в его основе, оказали огромное влияние на последующее развитие квантовой статистики. Несмотря на это, на данный момент осталось несколько нерешенных математических проблем, которые препятствуют широкому внедрению квантовой энтропии фон Неймана в расчетные алгоритмы для анализа хаотической динамики. Одна из ключевых – это построение хорошо локализованного как в координатном, так и в импульсном пространстве ортонормированного базиса. Несколько способов решения данной проблемы представлено в работах [2, 3], однако структура соответствующих базисов является очень сложной, что делает затруднительным их практическое применение.

В данной работе предложено несколько новых алгоритмов построения фазового пространства фон Неймана. Они являются приближенными, но более простыми в реализации.

1. Фазовое пространство фон Неймана

Рассмотрим одномерное движение вдоль оси x. Тогда в классической физике частице с координатой x и проекцией импульса p_x ставится в соответствие точка с координатами (x, p_x) в двумерном пространстве, которое называется фазовым. Если попытаться

воспользоваться этим понятием для описания поведения квантовых систем, то мы столкнемся со следующей проблемой: поскольку величины x и p_x одновременно не могут иметь определенное значение, точке в фазовом пространстве не будет соответствовать ни одно состояние частицы.

И. фон Нейман предложил способ обойти указанные выше трудности с помощью модифицированного фазового пространства [1]. Его основной идеей было использовать вместо x и p_x другую пару величин X, P_x, которые совместно измеримы, причем являются близкими аналогами x и p_x. Дадим более строгую формулировку этих положений.

1. Коммутатор квантовомеханических операторов $\hat{X}$, $\hat{P}_x$, соответствующих величинам X, P_x, должен быть равен нулю:

$$[\hat{X}, \hat{P}_x] = \hat{X}\hat{P}_x - \hat{P}_x\hat{X} = 0. \qquad (1)$$

Для этого необходимо и достаточно, чтобы операторы $\hat{X}$, $\hat{P}_x$ имели общий ортонормированный базис $\phi_k(x)$, $k \in \mathbb{Z}$, образованный их собственными функциями.

2. В собственных состояниях $\phi_k(x)$ средние значения координаты x и импульса p_x должны совпадать с собственными значениями X_k, P_k величин X, P_x в этих состояниях:

$$\begin{aligned} X_k &= \langle \phi_k | \hat{X} | \phi_k \rangle = \langle \phi_k | x | \phi_k \rangle, \\ P_k &= \langle \phi_k | \hat{P}_x | \phi_k \rangle = \langle \phi_k | \hat{p}_x | \phi_k \rangle. \end{aligned} \qquad (2)$$

3. Во всех состояниях $\phi_k(x)$ среднеквадратичные отклонения координаты Δx и импульса Δp_x должны удовлетворять условию

$$\Delta x \Delta p_x \le C\frac{\hbar}{2}. \qquad (3)$$

Здесь $1 \le C < \infty$ – некоторая константа. Это требование означает, что функции $\phi_k(x)$ должны быть хорошо локализованы как в координатном, так и в импульсном пространстве.

Как видим, для построения фазового пространства фон Неймана необходим ортонормированный базис $\phi_k(x)$ с равномерно ограниченной константой неопределенности $\Delta x \Delta p_x$. Сам И. фон Нейман предла-

гал поступить следующим образом [1]: получить $\phi_k(x)$, применив процедуру ортогонализации Грама–Шмидта к волновым пакетам вида

$$\psi_{km}(x) = \exp\left(-\frac{(x-k)^2}{2}\right) e^{i2\pi mx}, k, m \in \mathbb{Z}, \tag{4}$$

которые представляют собой когерентные состояния гармонического осциллятора. К сожалению, как было показано в работах [4, 5], устойчивая ортогонализация этой системы функций оказывается невозможной. В настоящее время ортонормированные базисы с равномерно ограниченной константой неопределенности строятся по иной технологии, используя методы теории вейвлетов [6], а также разложение в ряд Неймана [2, 3]. Однако недостатком такого подхода является сложная структура базисов, делающая их малопригодными для практического применения. Дело в том, что в реальных прикладных задачах фазовое пространство, как правило, имеет большую размерность, поэтому вычислительная сложность алгоритмов не должна быть слишком велика. Мы предлагаем несколько приближенных способов построения базиса $\phi_k(x)$, не требующих привлечения больших вычислительных мощностей. Для простоты будет рассмотрен одномерный случай, но все полученные результаты легко обобщаются.

2. Использование неполной подсистемы когерентных состояний

Рассмотрим вдвое прореженную подсистему когерентных состояний

$$g_{km}(x) = \exp\left(-\frac{(x-k)^2}{2}\right) e^{i4\pi mx}, k, m \in \mathbb{Z}. \tag{5}$$

В отличие от набора функций (4), для данного семейства волновых пакетов (5) устойчивая процедура ортогонализации возможна [5]. Ее можно осуществить, например, пользуясь следующим фактом [5].

Утверждение 1. Пусть

$$h_{km}(x) = h(x-k)e^{i4\pi mx}, k, m \in \mathbb{Z}, \tag{6}$$

$$h(x) = \sum_{k,m=-\infty}^{\infty} c_{km}^{\perp} g_{km}(x), \tag{7}$$

причем коэффициенты $c_{km}^{\perp}$ удовлетворяют соотношению

$$\left| \sum_{k,m=-\infty}^{\infty} c_{km}^{\perp} e^{ikx+imy} \right|^2 = \frac{1}{F(x,y)}, \tag{8}$$

$$F(x,y)=\sqrt{\pi}\theta_3\left(\frac{x}{2},e^{-1/4}\right)\theta_3\left(\frac{y}{2},e^{-4\pi^2}\right), \tag{9}$$

где $\theta_3(x,q)$ – третья тета-функция Якоби

$$\theta_3(x,q)=\sum_{k=-\infty}^{\infty} q^{k^2}\cos(2kx),\ |q|<1. \tag{10}$$

Тогда функции $h_{km}(x)$, $k,m\in\mathbb{Z}$ образуют ортонормированную систему.

Чтобы найти коэффициенты ортогонализации $c_{km}^{\perp}$ с помощью данных формул, нужно из правой части равенства (8) извлечь квадратный корень, а полученную функцию разложить в ряд Фурье. Затем значения $c_{km}^{\perp}$ следует занести в таблицу для ускорения расчетов. График функции $h(x)$, полученный таким способом, показан на рис. 1.

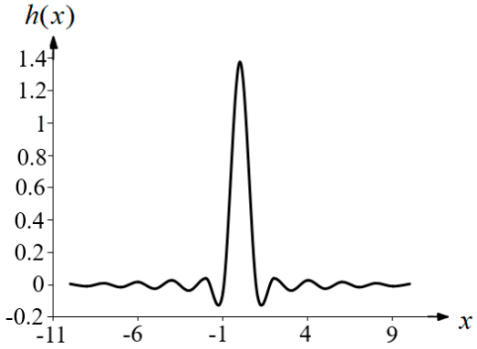


Рис. 1. График функции $h(x)$ (см. формулу (7))

К сожалению, набор функций $h_{km}(x)$, $k,m\in\mathbb{Z}$ не является полным в $L_2(\mathbb{R})$, однако это обстоятельство, на наш взгляд, не является критичным. Дело в том, что на практике границы суммирования всегда приходится ограничивать, выполняя конечномерную редукцию. Поэтому неполнота системы функций $h_{km}(x)$ не является ключевым фактором, гораздо важнее то, что она имеет довольно простую структуру. Тем не менее, ниже будет рассмотрен один из способов расширения набора функций $h_{km}(x)$.

3. Использование двухкомпонентной системы функций

Рассмотрим семейство функций вида

$$u_{km}(x) = u(x-k)e^{i4\pi mx}, k, m \in \mathbb{Z}, \tag{11}$$

$$u(x) = \exp\left(-\frac{x^2}{4}\right)\theta_3\left(\frac{3\pi x}{2}, e^{-\pi^2}\right)e^{i2\pi x}. \tag{12}$$

График функции $u(x)$ показан на рис. 2.

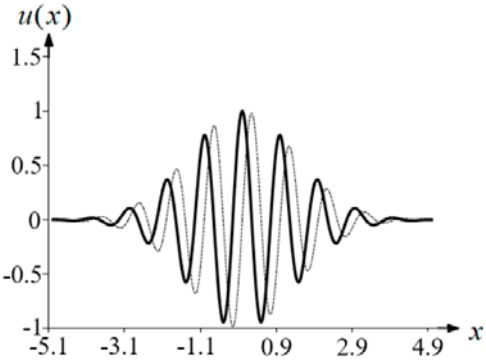


Рис. 2. График действительной части (сплошная линия) функции $u(x)$ (см. формулу (12)) и мнимой (пунктир)

Прямой подстановкой можно убедиться, что $u_{km}(x)$ ортогональны всем функциям $g_{km}(x)$, а значит и всем $h_{km}(x)$. Соответствующие расчеты являются слишком громоздкими, поэтому мы их опустим. За счет этого появляется возможность расширить семейство функций $h_{km}(x)$ до новой ортонормированной системы. Для этого необходимо отдельно провести ортогонализацию $u_{km}(x)$. Действуя аналогично статье [5], приходим к следующему результату.

Утверждение 2. Пусть

$$\upsilon_{km}(x) = \upsilon(x-k)e^{i4\pi mx}, k, m \in \mathbb{Z}, \tag{13}$$

$$\upsilon(x) = \sum_{k,m=-\infty}^{\infty} d_{km}^{\perp} u_{km}(x), \tag{14}$$

причем коэффициенты $d_{km}^{\perp}$ удовлетворяют соотношению

$$\left|\sum_{k,m=-\infty}^{\infty} d_{km}^{\perp} e^{ikx+imy}\right|^2 = \frac{1}{\tilde{F}(x,y)}, \tag{15}$$

$$\tilde{F}(x,y)=\frac{1}{2}\sum_{p,r=-\infty}^{\infty}e^{ipx}\exp\left(-\frac{(y+2\pi r)^2}{32\pi^2}\right)\exp\left(-\frac{p^2}{8}\right)\times$$
$$\times\theta_3\left(\frac{3}{8}(y+2\pi(r+p)),e^{-\pi^2}\right)\theta_3\left(\frac{3}{8}(y+2\pi(r-p)),e^{-\pi^2}\right). \qquad (16)$$

Тогда функции $\upsilon_{km}(x)$, $k,m\in\mathbb{Z}$ образуют ортонормированную систему.

График функции $\upsilon(x)$ показан на рис. 3.

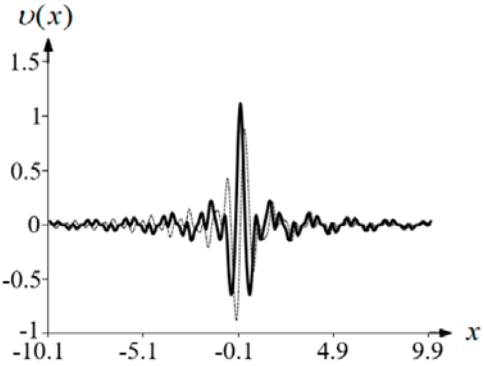


Рис. 3. График действительной части (сплошная линия) функции $\upsilon(x)$ (см. формулу (14)) и мнимой (пунктир)

Набор функций $h_{km}(x)$, $k,m\in\mathbb{Z}$ вместе с $\upsilon_{km}(x)$, $k,m\in\mathbb{Z}$ образует новую, расширенную ортонормированную систему. Вопрос о том, является ли это семейство функций полным в $L_2(\mathbb{R})$, к сожалению, пока остается открытым.

Заключение

В данной работе предложено два новых приближенных способа построения фазового пространства фон Неймана. В их основе лежит поиск хорошо локализованного ортонормированного базиса. Первый способ состоит в использовании прореженной вдвое неполной подсистемы когерентных состояний с последующей ее ортогонализацией. Достоинством этой системы функций является простота ее структуры, а недостатком – неполнота. Второй способ позволяет частично решить проблему неполноты с помощью расширения исследуемого набора функций. К сожалению, ответить на вопрос о том, является ли новая, расширенная система функций полной, пока не удалось.

В будущем планируется приближенно построить операторы $\hat{X}$, $\hat{P}_x$ с помощью двух предложенных систем функций, а также выполнить вычислительные эксперименты, связанные с соответствующей квантовой энтропией.

Список литературы

1. Нейман, И. Математические основы квантовой механики / И. Нейман ; пер. с нем М. К. Поливанова, Б. М. Степанова ; под ред. Н. Н. Боголюбова. – Новокузнецк : Новокузнецкий физ.-мат институт, 2000. – 367 с.
2. Bourgain, J. A Remark on the Uncertainity Principle for Hilbertian Basis / J. Bourgain // Journal of Functional Analysis. – 1988. – № 79. – PP. 136–143.
3. Han, X. Entropy for Quantum Pure States and Quantum H Theorem / X. Han, B. Wu // Physical Review E. – 2015. – № 91. – P. 062106.
4. Переломов, А. М. Обобщенные когерентные состояния и их применения / А. М. Переломов. – М. : Наука, 1987. – 268 с.
5. Киселев, Е. А. Вычисление констант Рисса и ортогонализация для неполных систем когерентных состояний с помощью тета-функций / Е. А. Киселев, Л. А. Минин, И. Я. Новиков // Математический сборник. – 2016. – Т. 207, № 8. – С. 101–116.
6. Новиков, И. Я. Теория всплесков / И. Я. Новиков, В. Ю. Протасов, М. А. Скопина. – М. : Физматлит, 2005. – 616 с.

Методы аппроксимации, основанные на системе целочисленных сдвигов функции Гаусса

К. В. Наконечный

Студент магистратуры

П. В. Шачина

Студент бакалавр

Е. А. Киселев

Доцент

Введение

Большинство современных алгоритмов цифровой обработки сигналов основано на использовании процедуры разложения в ряд по какому-либо набору функций. Сюда можно отнести ставшие на данный момент классическими методы Фурье-анализа, а также различные модификации вейвлет-преобразования. Удачный выбор семейства функций, по которому производится разложение, является залогом эффективной работы алгоритма.

Как правило, дискретизацию сигнала проводят с равномерным шагом. По этой причине весьма удобным математическим аппаратом выступают системы целочисленных сдвигов. Среди них наиболее популярными в настоящее время являются семейства сдвигов, порожденные функцией Гаусса, и фундаментальные сплайны [1, 2].

При использовании сдвигов функции Гаусса возникает ряд проблем. Во-первых, они не являются ортогональными. Во-вторых, если полуширина функции Гаусса более чем в 2 раза превосходит шаг дискретизации, вычислительные алгоритмы работают неустойчиво [3]. В данной работе предлагается один из возможных способов решения первой проблемы, который при этом позволяет снизить погрешность аппроксимации по сравнению с классическими вариантами интерполяции для систем целочисленных сдвигов.

1. Классический вариант интерполяции

Пусть $f(x)$ – некоторая исследуемая функция. Рассмотрим набор целочисленных узлов $x_m = m$, $m \in \mathbb{Z}$. Требуется построить функцию $\tilde{f}(x)$ вида

$$\tilde{f}(x) = \sum_{k=-\infty}^{\infty} c_k \exp\left(-\frac{(x-k)^2}{2\sigma^2}\right), \sigma > 0 \quad (1)$$

с неопределенными коэффициентами c_k, значения которой в узлах совпадают с соответствующими значениями $f(x)$, т. е.

$$\tilde{f}(m) = f(m), m \in \mathbb{Z}. \quad (2)$$

Построение интерполирующей функции $\tilde{f}(x)$ удобно проводить с помощью так называемой узловой функции [1]. Иногда используется также термин «кардинальная функция», а в теории сплайнов – «фундаментальный сплайн» [1, 2].

Определение. Функция

$$\tilde{\phi}(x) = \sum_{k=-\infty}^{\infty} d_k(\sigma) \exp\left(-\frac{(x-k)^2}{2\sigma^2}\right) \quad (3)$$

называется узловой, если в целочисленных точках она принимает следующие значения

$$\tilde{\phi}(m) = \delta_{0,m}, m \in \mathbb{Z}. \quad (4)$$

Чтобы построить $\tilde{\phi}(x)$, необходимо определить коэффициенты $d_k(\sigma)$ в формуле (3). Это было сделано в монографии [1]:

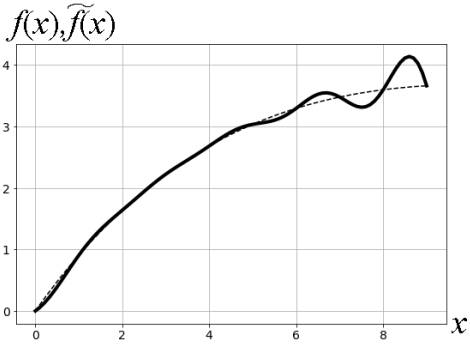
$$d_k(\sigma) = \frac{1}{C(\sigma)} \exp\left(\frac{k^2}{2\sigma^2}\right) \sum_{r=|k|}^{\infty} (-1)^r \exp\left(-\frac{(r+1/2)^2}{2\sigma^2}\right), \quad (5)$$

$$C(\sigma) = \sum_{k=-\infty}^{\infty} (4r+1) \exp\left(-\frac{(2r+1/2)^2}{2\sigma^2}\right). \quad (6)$$

Когда $\tilde{\phi}(x)$ известна, $\tilde{f}(x)$ легко найти по формуле

$$\tilde{f}(x) = \sum_{m=-\infty}^{\infty} f(m)\tilde{\phi}(x-m). \quad (7)$$

На рис. 1 изображен график тестовой функции $f(x) = x\exp(-x/10)$ и построенной для нее по формулам (3), (5-7) интерполирующей функции $\tilde{f}(x)$ при $\sigma = 1$. Из рис. 1 следует, что между узлами погрешность интерполяции может быть достаточно велика. В следующем пункте будет предложен способ уменьшения погрешности без значительного усложнения алгоритма.



пунктир – тестовый сигнал $f(x)$, *сплошная линия – соответствующая ему интерполирующая функция* $\tilde{f}(x)$

Рис. 1. Тестовый сигнал и интерполирующая функция

2. Ортогональная проекция

Хорошо известно, что система целочисленных сдвигов функции Гаусса не является полной в $L_2(\mathbb{R})$ [4]. Это обстоятельство выступает основной причиной того, что между узлами интерполяции может возникать значительная погрешность. Уменьшить погрешность по норме $L_2(\mathbb{R})$ можно, находя ортогональную проекцию исследуемой функции $f(x)$ на подпространство $L_2(\mathbb{R})$, представляющее собой замыкание линейной оболочки системы целочисленных сдвигов функции Гаусса. При этом повышается точность в среднем, но совпадения в некотором заданном наборе узлов (как было при интерполяции) уже наблюдаться, вообще говоря, не будет. Мы разработали алгоритм, который совмещает в себе преимущества интерполяции и ортогонального проектирования.

Рассмотрим функцию

$$h(x) = \sum_{k=-\infty}^{\infty} c_k^{\perp}(\sigma) \exp\left(-\frac{(x-k)^2}{2\sigma^2}\right), \sigma > 0. \tag{8}$$

Потребуем, чтобы ее целочисленные сдвиги $h(x-k), k \in \mathbb{Z}$ образовывали ортонормированную систему, т. е.

$$\int_{-\infty}^{\infty} h(x-k)\overline{h(x-m)}dx = \delta_{k,m}, k, m \in \mathbb{Z}. \tag{9}$$

Известен следующий факт [3].

Утверждение. Для того чтобы система функций $h(x-k), k \in \mathbb{Z}$ была ортонормированной, необходимо и достаточно, чтобы коэффициенты $c_k^{\perp}(\sigma)$ удовлетворяли соотношению

$$\left| \sum_{k=-\infty}^{\infty} c_k^{\perp}(\sigma) e^{ikt} \right|^2 = \frac{1}{\sigma\sqrt{\pi}\theta_3\left(\frac{t}{2}, \exp\left(-\frac{1}{4\sigma^2}\right)\right)}, \tag{10}$$

где $\theta_3(x,q)$ – третья тета-функция Якоби

$$\theta_3(x,q) = \sum_{k=-\infty}^{\infty} q^{k^2} \cos(2kx), |q| < 1. \tag{11}$$

Как правило, в формуле (10) квадратный корень извлекают симметричным образом. В рамках данной работы, вопреки общепринятой технологии, мы осуществили процедуру извлечения корня по лемме Рисса, т. е. разложив правую часть на комплексно сопряженные множители. Для этого полезной оказывается следующая формула [5]:

$$\theta_3(z,q) = \frac{2e^{i\pi\tau/4}}{\theta_2(0,q)} \theta_3\left(z + \frac{\pi\tau}{2}, q^2\right) \theta_3\left(z - \frac{\pi\tau}{2}, q^2\right), q = e^{i\pi\tau}, \tag{12}$$

где $\theta_2(z,q)$ – вторая тета-функция Якоби

$$\theta_2(z,q) = \sum_{k=-\infty}^{\infty} q^{(k+1/2)^2} e^{i(2k+1)z}, q = e^{i\pi\tau}. \tag{13}$$

Далее необходимо разложить полученную функцию в ряд Фурье на отрезке $[-\pi, \pi]$. Это можно сделать численно с помощью дискретного преобразования Фурье. На рис. 2 представлен график функции $h(x)$, полученный таким способом. Интересной особенностью $h(x)$ является то, что при сдвиге на $1/4$ она с хорошей точностью становится практически узловой, т. е. удовлетворяет соотношению

$$h\left(m + \frac{1}{4}\right) \approx \delta_{0,m}, m \in \mathbb{Z}. \tag{14}$$

Благодаря этому обстоятельству, мы получаем математический аппарат, который позволяет одновременно с интерполяцией находить ортогональную проекцию. Это дает возможность снизить погрешность интерполяции.

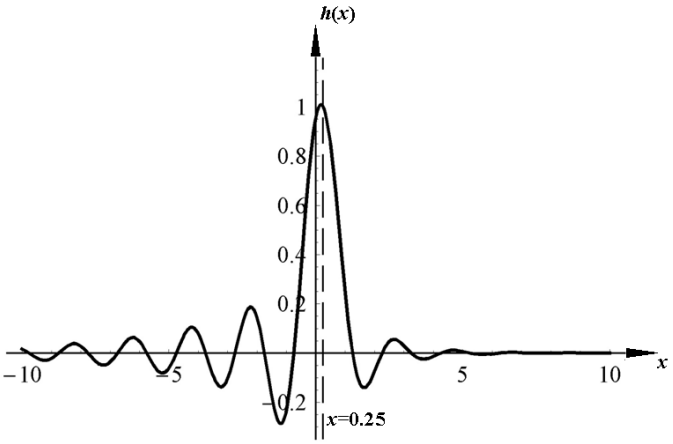


Рис. 2. Функция $h(x)$ (см. формулу (8)), порождающая ортонормированную систему целочисленных сдвигов

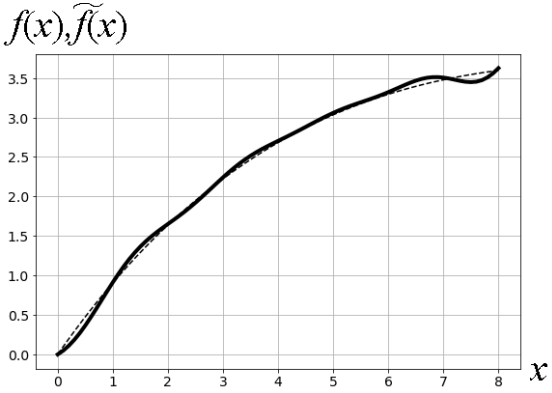
На рис. 3 изображен график тестовой функции $f(x) = x\exp(-x/10)$ и построенной для нее интерполирующей функции $\tilde{f}(x)$ на основе сдвигов $h(x)$ при $\sigma = 1$. Из рис. 3 следует, что точность интерполяции действительно увеличилась по сравнению с классическим вариантом (см. рис. 1).

Заключение

В данной работе рассмотрено два метода аппроксимации с помощью системы целочисленных сдвигов функции Гаусса: интерполяция и ортогонализация. Новизной является то, что ортогонализацию предложено выполнить нестандартным образом: квадратный корень в формуле (10) извлекается несимметрично. Установлено, что полученная ортогональная система сдвигов $h(x-k)$, $k \in \mathbb{Z}$, при этом порождает семейство функций, которые с хорошей точностью можно считать узловыми. Благодаря этому, разработан приближенный алгоритм, который позволяет одновременно с интерполяцией проводить ортогональное проектирование.

Предложенный алгоритм протестирован с помощью вычислительных экспериментов. Расчеты показали, что он работает эффективнее классического варианта интерполяции, поскольку для одних и тех же исходных функций дает в среднем меньшую погрешность между узлами интерполяции. Вычислительная сложность алгоритма при этом не стала

выше, а общая структура по-прежнему является вполне удобной для компьютерной реализации.



пунктир – тестовый сигнал $f(x)$*, сплошная линия – соответствующая ему интерполирующая функция* $\tilde{f}(x)$

Рис. 3. Тестовый сигнал и интерполирующая функция на основе сдвигов

Список литературы

1. Maz'ya, V. Approximate approximations / V. Maz'ya, G. Schmidt. – 350 p. – AMS, 2007. – (Mathematical Surveys and Monographs ; v. 141).

2. Чуи, Ч. Введение в вейвлеты : учебное пособие для студентов вузов / Ч. Чуи ; пер. с англ. Я. М. Жилейкина. – М. : Мир, 2001. – 412 с.

3. О константах Рисса для некоторых систем целочисленных сдвигов / Е. А. Киселев [и др.] // Математические заметки. – 2014. – Т. 96, вып. 2. – С. 239–250.

4. Киселев, Е. А. О построении биортогональных систем подпространств, порожденных целочисленными сдвигами одной функции / Е. А. Киселев, Л. А. Минин, И. Я. Новиков // Математические заметки. – 2014. – Т. 96, вып. 3. – С. 468–470.

5. Уиттекер, Э. Т. Курс современного Анализа : пер. с англ. В 2 т. Ч. 2. Трансцендентные функции / Э. Т. Уиттекер, Дж. Н. Ватсон ; под ред. Ф. В. Широкова. – изд. 2-е. – М. : Физматлит, 1963. – 515 с.

Контекстные помощники и их реализация

Д. И. Науменко

Студент магистратуры

А. Ю. Иванков

Доцент

Введение

В настоящее время контекстные помощники являются одним из наиболее применяемых направлений практически в любой сфере жизни: голосовое управление бортовым компьютером автомобиля, робот-помощник в навигации по сайту, электронный секретарь в банковском приложении и т.д. Большой спрос на подобную функциональность растет в соответствии с ростом уровня жизни современного человека, а также с повышением производительности и возрастающими возможностями систем обработки, хранения и передачи информации, наряду с появлением новых технологий, методов обработки и анализа информации.

Одной из основных проблем при разработке контекстного помощника, да и языковой модели в целом, является разнообразие натурального языка, омонимы и постоянное языковое развитие. В добавлении ко всему вышесказанному, развитие и динамика натурального языка не всегда равномерны, что ведет к применению бесконечного количества различных версий одного и того же языка, например: в зависимости от географического положения, сферы деятельности использующих язык, сферы объекта описания и т.д. Наличие такого рода неравномерности не позволяет использовать одну языковую модель на все случаи жизни, а значит заставляет создавать новые подходы в обработке натуральных текстов в огромном количестве прикладных задач. Данная работа посвящена разработке и программной реализации подхода для обучения языковых моделей в рамках создания контекстного помощника в сфере информационных технологий и телефонии.

1. Токенизация

Для работы контекстного помощника данные, являющиеся высказываниями пользователей на натуральном языке, изначально

необходимо перевести из вида строки в вид последовательного набора объектов (токенов). Для этого воспользуемся WhitespaceTokenizer. Данный токенизатор использует символ пробела в качестве разделителя. Также, любой символ, не подходящий под регулярное выражение **[a-zA-Z0-9_#@&]** будет заменен на пробел, если выполняется хотя бы одно условие:

- этот символ следует за пробелом;
- этот символ предшествует пробелу;
- этот символ является первым в строке;
- этот символ является последним в строке.

2. Извлечение признаков

Извлечение признаков из последовательности токенов может занимать любое количество шагов. В данной работе этот этап ограничен 4 шагами:

- RegexFeaturizer
- LexicalSyntacticFeaturizer
- CountVectorsFeaturizer (конфигурация 1)
- CountVectorsFeaturizer (конфигурация 2)

1. **RegexFeaturizer** извлекает из последовательности токенов признаки, которые будут использованы для классификации сущностей и намерений. Данный инструмент основывается на объявленных в специальном формате регулярных выражениях. Для каждого регулярного выражения будет создан признак, хранящий количество вхождений данного регулярного выражения в сообщение пользователя.

2. **LexicalSyntacticFeaturizer** извлекает из последовательности токенов признаки, которые будут использованы только для определения сущностей. Скользящим окном проходя по каждому токену из сообщения пользователя, создает набор лексических и синтаксических признаков согласно конфигурации. Пример конфигурации с пояснением каждого признака приведставлен в таблице 1.

Каждый токен, как часть признаков, которыми он описывается, может также хранить признаки уже прошедших токенов, свои, а также токенов, следующих за текущим. Наборы признаков для текущего, предыдущих и следующих токенов могут отличаться. Эти настройки зависят от конфигурации скользящего окна.

3. **CountVectorsFeaturizer** извлекает из последовательности токенов признаки, которые будут использоваться только для определения намерений. Создает представление пользовательского сообщения в виде мешка слов. Все токены, состоящие только из цифр будут засчитаны как один и тот же токен в мешке слов.

Таблица 1

Описание применяемой в подходе конфигурации LexicalSyntaticFeaturizer'a

Имя признака	Описание
BOS	Бинарный признак. Хранит информацию о том, является ли токен началом строки.
EOS	Бинарный признак. Хранит информацию о том, является ли токен концом строки.
low	Бинарный признак. Показывает написан ли токен полностью в нижнем регистре.
upper	Бинарный признак. Показывает написан ли токен полностью в верхнем регистре.
title	Бинарный признак. Показывает является ли первый символ токена верхнего регистра, а все остальные – нижнего регистра.
digit	Бинарный признак. Хранит информацию о том, состоит ли токен только из цифр.
prefixN	Строковый признак. Хранит первые N символов из токена.
suffixN	Строковый признак. Хранит последние N символов из токена.
pos	Перечисляемый признак. Хранит название части речи, которой является токен.

Этот инструмент может быть сконфигурирован для создания «эн-грамм» символов или токенов. Максимальные и минимальные размеры «эн-грамм» также конфигурируемы. При использовании в режиме токенов, по умолчанию каждый токен будет лемматизироваться и в расчет будет идти именно лемма токена.

Учитывая большое количество настроек, в данной работе этот инструмент использовался дважды: в первый раз признаки обогащаются в режиме токенов, а во второй раз – в режиме символов.

3. Классификация намерений и сущностей

На данном этапе в этой работе используется DIETClassifier, где DIET расшифровывается как Dual Intent Entity Transformer (Парный Трансформатор для Намерений и Сущностей). Как следует из названия, архитектура данной модели является многозадачной и используется одновременно для классификации намерений пользователя и извлечения сущностей из сообщения пользователя.

На рисунке ниже можно видеть архитектуру данной модели. Красным выделены блоки, создающие векторное представление

объекта. Все полносвязные слои в данной архитектуре на самом деле не полносвязные, а урезаются до ~20% всех соединений и имеют одни и те же веса.

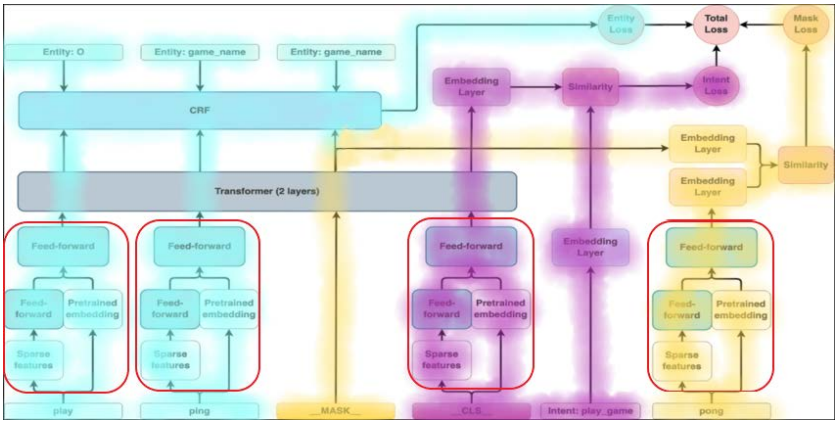


Рис. 1. Архитектура модели DIETClassifier

Процесс тренировки данной модели сводится к минимизации аддитивной функции потерь. Данная функция получается путем сложения трёх других функций потерь.

Первая функция потерь выделена желтым цветом на рисунке. Она завязана на понятии маскирования. Во время обучения один из токенов заменяется маской. Замененный токен, проходя через цепочку полносвязных слоев, получает векторное представление размерности 256. Маска, проходящая через блок трансформеров, получает векторное представление той же размерности. Далее ищется расстояние между данными векторами, это расстояние и является одной из функций потерь

Вторая функция потерь выделена фиолетовым цветом на рисунке. Она, по принципу первой функции, показывает расстояние между объектом предложения (сообщения пользователя) и объектом намерения.

Третья функция потерь выделена голубым цветом на рисунке. Данная функция считает разницу в предсказанных и истинных значениях сущностей для каждого токена. После трансформатора идет слой с CRF (условные случайные поля). Это необходимо для того, чтобы между предсказанными сущностями также была какая-то обучаемая связь, которая влияла бы на предсказание класса сущности.

В центре всей архитектуры находится двуслойный трансформатор. Именно в нем происходит большая часть математики по классификации

намерений пользователя, выделения сущностей, а также предсказания маскированного токена. Трансформаторы сами по себе используют механизм «внимания» (attention). Каждый токен имеет векторное представление. Пусть это будет вектор V длиной n, что показывает количество признаков, которым описывается вектор. Количество таких векторов равно количеству токенов в сообщении, пусть это будет N.

$$V_i = [x_0 ... x_n], i \in [1, n]. \quad (1)$$

Механизм «внимания» позволяет подкорректировать значения вектора, внеся в него «контекст», то есть представление каждого токена влияет как на само себя, так и на представления остальных токенов в сообщении. Для изменения значений используется коэффициент пересчета W, который вычисляется путем векторного умножения представлений токенов друг на друга с последующей нормализацией.

$$S_i = [S_{i1} ... S_{in}], i \in [1, N], \quad (2)$$

$$S_{ij} = V_i \bullet V_j = V_i^T V_j = \sum_{k=0}^{n} V_{ik} V_{jk}, \quad (3)$$

$$W_i = [W_{i1} ... W_{in}], i \in [1, N], \quad (4)$$

$$W_{ij} = S_{ij} / \max(S_i); \sum_{j=0}^{n} W_{ij} = 1. \quad (5)$$

Пересчитанный с учетом контекста вектор Y является суммой помноженных векторов V на весовые коэффициенты W.

$$Y_i = \sum_{j=0}^{N} W_{ij} * V_j, i = [1, N]. \quad (6)$$

Таким образом в каждом из векторов Y хранится не только информация о признаках, которые были выделены из токена, но и информация о контексте сообщения. Заметим, что размерности векторов V и Y совпадают, что означает возможность неоднократного повторения данной операции.

Итого, на выходе после применения данного классификатора предсказанное намерение пользователя и выделенные из сообщения сущности могут быть использованы далее.

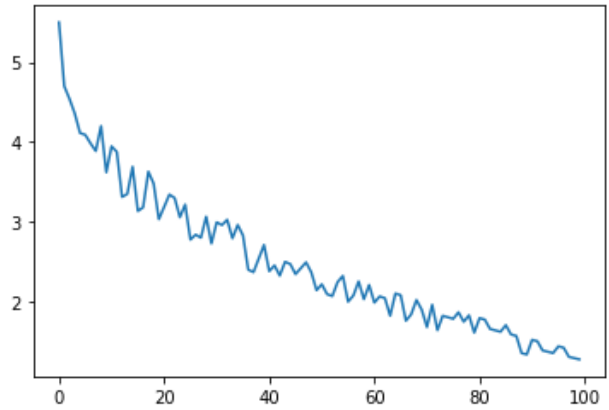


Рис. 2. График изменения функции потерь

4. Выбор следующего действия

Имея информацию о намерении пользователя и выделенных из сообщения сущностях, контекстным помощником выполняется одно из предписанных действий. Действием может выступать любая операция, заранее определённая разработчиком, а сущности могут являться входными параметрами для этой операции, например:

- запрос в базу данных;
- выполнение вычислений;
- запрос на сторонний API;
- сценарный ответ пользователю;
- уточняющий ответ пользователю с запросом значений недостающих сущностей.

Данное действие конфигурируется с помощью двух политик: политики правил и политики историй. По политике правил, если были извлечены определенные сущности и выявлено определенное намерение, то контекстный помощник обязательно выполнит определенное ответное действие. Согласно политике историй, если действие пользователя не подошло ни под одно правило, то следующее действие выбирается на основании предопределенных возможных течений диалога, а также во внимание принимаются все предыдущие вопросно-ответные блоки.

Заключение

Данная статья посвящена разработке и программной реализации подхода для обучения языковых моделей в рамках создания

контекстного помощника. В работе были описаны примеры выделения признаков из сообщений, написанных на натуральном языке и построение модели для определения намерений пользователя и выделения сущностей. В итоге данной разработки получился полностью рабочий алгоритм, реализованный в виде модуля приложения. Использование данного решения позволяет отказаться от предобученных языковых моделей в задачах со специфическим, узконаправленным контекстом. В дальнейшем предполагается использование данного модуля в полноценном контекстном помощнике в задаче конфигурации объектов сферы информационных технологий и телефонии.

Список литературы

1. Лопер, Э. Natural language processing with Python / Э. Лопер, С. Берд, Ю. Кляйн. – O'REILLY, 2009. – 502 с.
2. Гольдберг, Й. Нейросетевые методы в обработке естественного языка / Й. Гольдберг; пер. с анг. А. А. Слинкина. – М. : ДМК Пресс, 2019. – 282 с.
3. Документация фрэймворка RASA [Электронный ресурс] : база данных. – Режим доступа: https://rasa.com/docs/

Оценивание знаний студентов в рамках образовательного процесса с помощью нечетких временных рядов

В. Г. Сафонов

Студент магистратуры

М. Г. Матвеев

Профессор

Н. А. Алейникова

Доцент

Введение

Во многих высших учебных заведениях для оценки знаний студентов применяется балльно-рейтинговая система, основанная на порядковых шкалах и последующем переводе баллов из одной шкалы в другую. Примером является 50-ти балльная порядковая шкала с последующим переводом в 4-х балльную порядковую шкалу. Как правило, для определения среднего балла студента используется средняя арифметическая, что объясняется относительной простотой получения данной характеристики. Однако широкое использование среднего балла как универсального параметра не оправдано, потому что имеет значительные ограничения [1]:

1. на среднее арифметическое чрезмерно влияют значения в выборке, которые слишком малы или слишком велики, поэтому среднее арифметическое не будет объективной характеристикой оцениваемого субъекта;
2. в порядковых шкалах при допустимых преобразованиях (к которым относятся все строго возрастающие преобразования) средние арифметические дают противоречивые результаты. Например, если рассмотреть две порядковые шкалы, между которыми установлено соответствие (таблица 1), то средний балл, полученный на этих шкалах с помощью средней арифметической, будет противоречивым.

Кроме того, авторы исходят из того, что отметка не всегда является объективным способом оценивания знаний и способностей студента. Оценивая студента с помощью отметок, преподаватели по сути

пытаются охарактеризовать его способности к обучению, трудолюбие, проявленный интерес к предмету.

Таблица 1

Соответствие между двумя порядковыми шкалами

50 балльная шкала	1-24	25-34	35-44	45-50
4 балльная шкала	2	3	4	5

По 50-ти балльной шкале лучше будет студент 2, а по 4-х балльной – студент 1 (см. таблицу 2).

Таблица 2

Расчет средних арифметических в разных порядковых шкалах

Студент	Шкала	Баллы				Средние арифметические
Студент 1	50-я шкала	25	25	45	50	36,25
	4-я шкала	3	3	5	5	4,00
Студент 2	50-я шкала	34	40	42	40	39,00
	4-я шкала	3	4	4	4	3,75

При этом относя его к одному из классов (которые в дальнейшем будем называть состояниями студента) «Неудовлетворительно», «Удовлетворительно», «Хорошо», «Отлично». Отметка, полученная студентом, в каждый момент времени лишь в той или иной мере соответствует этим понятиям. Отметки студента зависят от очень многих факторов, не обязательно связанных с его знаниями, в том числе и таких как: не выспался, не доучил, переволновался и т.д. Преподаватели тоже не всегда могут объективно оценивать знания, могут колебаться в выборе той или иной отметки, могут в силу разных причин по разному относиться к самому студенту (студент много пропускает, но на аттестации показывает хорошие результаты и т.д.), может не быть четко сформулированных критериев, по которым ставятся отметки. Это позволяет говорить о том, что на отметки студента влияют различные внешние и внутренние факторы, искажающие представление об истинном состоянии знаний студента. Способный обучающийся, в силу обстоятельств, мог пропустить занятия и получить плохую отметку. Но это не помешает ему оставаться потенциально способным и вернуться в свой стационарный статус.

Целью данной работы является установление соответствия между состояниями студента и временными рядами отметок, им полученных, с последующим прогнозированием этих состояний.

2. Разработка алгоритма для оценки знаний студента

Состояния студентов обозначим следующим образом:

1. $\tilde{S}_1 =$ «Неудовлетворительно»,
2. $\tilde{S}_2 =$ «Удовлетворительно»,
3. $\tilde{S}_3 =$ «Хорошо»,
4. $\tilde{S}_4 =$ «Отлично».

Оцениваемое понятие «знания студента» можно представить, как лингвистическую переменную

$$\left(X, \tilde{S}, U, G, M\right), \tag{1}$$

где X – название переменной – «знания студента», $\tilde{S} = \left(\tilde{S}_1, \tilde{S}_2, \tilde{S}_3, \tilde{S}_4\right)$ – терм-множество нечетких переменных (состояний студентов), U – универсальное множество, в качестве которого в дальнейшем будем рассматривать порядковую 50-ти балльную шкалу, G – синтаксическое правило, M – семантическое правило, определяющие функции принадлежности для каждого терма [2].

Следует сказать, что в каждый момент времени состояния студента непосредственно не наблюдаемы. Имеются лишь их оценки в виде отметок (баллов). Это могут быть отметки за аттестацию, контрольные работы в течение семестра и т.д. Обозначим отметки по одной дисциплине за определенный промежуток времени (например, семестр) как

$$X = \left(x_1; x_2; ...; x_n\right). \tag{2}$$

Совокупность отметок (2) представляет собой временной ряд, моменты времени $t = 1; 2; ...; n$ фиксированы. Далее полагаем, что x_j – это балл, выставленный по 50-ти балльной системе.

Как уже говорилось, баллы лишь приближенно могут характеризовать то или иное соответствие определенному состоянию студента $\tilde{S}_j$. Для построения функций принадлежности нечетких переменных $\tilde{S}_j$, разобьем дискретных интервал [0; 50] на 4 пересекающихся подынтервала и над каждым введем треугольную функцию принадлежности, как показано на рис. 1. Параметры треугольной функции принадлежности можно определить экспертным путем [3].

В результате имеем временной ряд баллов, которые с определенными значениями функции принадлежности $\mu_{\tilde{s}_j}(x_t)$ характеризуют состояния знаний студентов (рис. 2).

Подобного рода временной ряд можно отнести к нечетким временным рядам, который далее будем обозначать через $\tilde{X}$.

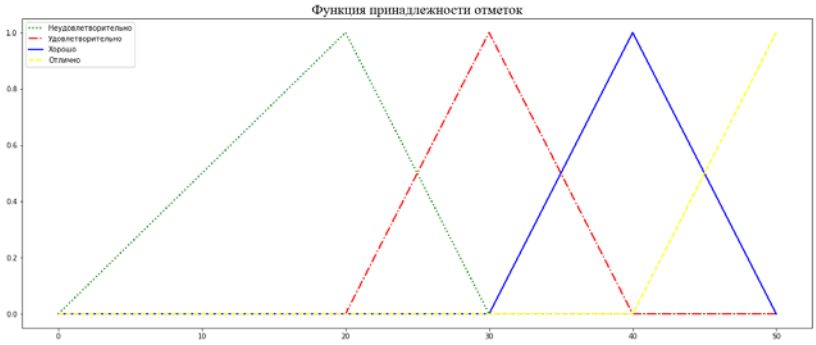


Рис. 1. Функции принадлежности отметок

Нечетким временным рядом (НВР) называют упорядоченную в равноотстоящие моменты времени последовательность наблюдений над некоторой системой с изменяющимися состояниями, если значение состояния в момент времени t может быть выражено с помощью нечеткой переменной $\tilde{X}$ [4].

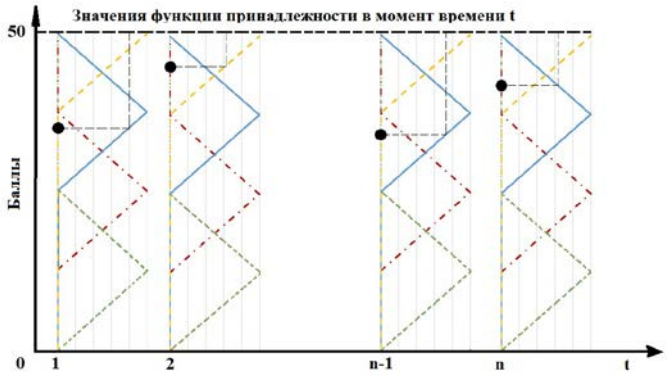


Рис. 2. Значение функции принадлежности в момент времени *t*

Определим состояние в момент времени t, при условии, что в этот момент времени наблюдается значение уровня временного ряда x_t, по формуле [5]:

$$\tilde{S}_{j,t} : \mu_{\tilde{S}_j}(x_t) = \max_j \mu_{\tilde{S}_j}(x_t). \quad (3)$$

Тот факт, что, будучи в момент времени $t-1$ в состоянии $\tilde{S}_{i,t-1}$, в следующий момент времени t студент оказался в состоянии $\tilde{S}_{j,t}$ обозначим с помощью отношения импликации:

$$\tilde{S}_{i,t-1} \to \tilde{S}_{j,t}. \quad (4)$$

Таким образом, по формулам (3), (4) будет осуществлен переход от временного ряда баллов к последовательности импликаций состояний:

$$\tilde{S}_{i,1,1} \to \tilde{S}_{i,2,2} \to ... \to \tilde{S}_{i,n-1,n-1} \to \tilde{S}_{i,n,n}, \ \left(i_p = \overline{1,4}\right). \quad (5)$$

Обозначим через k_{ij} частоту переходов из i-го состояния в j-е $\left(i, j = \overline{1,4}\right)$.

Поскольку состояний конечное число, и переход из одного состояния в другое происходит в дискретные моменты времени, то в предположении, что процесс перехода обладает свойством марковости, адекватным описанием динамики состояний будут дискретные марковские цепи [6]. В этом случае вероятности того, что студент будет в момент времени t находиться в состоянии $\tilde{S}_{j,t}, j = \overline{1,4}$, можно найти по формуле:

$$p^t = M^t p^{(t-1)}, \ t = 1; 2; ...; n, \quad (6)$$

где t – дискретное время; $p^t = \left(p_1^t; p_2^t; p_3^t; p_4^t\right)$ – вектор состояний студента, заданный вероятностями нахождения студента в соответствующем состоянии; M^t – стохастическая матрица 4×4 вероятностей переходов из состояний с элементами p_{ij}, где p_{ij} – вероятности перехода системы из состояния $\tilde{S}_{i,t-1}$, $\left(i = \overline{1,4}\right)$ в состояние $\tilde{S}_{j,t}$, $\left(j = \overline{1,4}\right)$. Начальное распределение p^0 – задано.

Вероятности $p_{ij} = \left(\tilde{S}_{i,t-1} \to \tilde{S}_{j,t}\right)$ могут меняться во времени, но на начальном этапе исследований примем их константами. В этом случае марковская цепь называется однородной.

Для работы с однородной марковской цепью необходимо задать стохастическую матрицу M. Элементы матрицы M предлагается находить по формулам:

$$\sum_{j=1}^{4} p\left(\tilde{S}_{i,t-1} \to \tilde{S}_{j,t}\right) = 1, \quad i, j = \overline{1,4}, \tag{7}$$

$$p_{ij} = p\left(\tilde{S}_{i,t-1} \to \tilde{S}_{j,t}\right) = \frac{k_{ij} \sum_{t=1}^{n} \mu_{\tilde{s}_j}(x_t)}{\sum_{j=1}^{4} k_{ij} \sum_{t=1}^{n} \mu_{\tilde{s}_j}(x_t)}, \quad i, j = \overline{1,4}. \tag{8}$$

С помощью матрицы M можно, во-первых, прогнозировать состояния студента на следующем шаге. Во-вторых, можно определить предельные (финальные) вероятности состояний студента:

$$\overline{p}_j = \lim_{t \to \infty} p_j^t, \; j = \overline{1,4}, \tag{9}$$

не зависящие от вектора начальных условий. Вектор $\overline{p} = \left(\overline{p}_1; \overline{p}_2; \overline{p}_3; \overline{p}_4\right)$ показывает вероятности установившихся (стационарных) состояний студента. Для нахождения вектора $\overline{p}$ используется векторно-матричное уравнение:

$$\overline{p} \times \left(M - E\right) = 0, \tag{10}$$

где E – единичная матрица. Формула (10) представляет собой систему линейных алгебраических уравнений с числом неизвестных равным количеству возможных состояний. Эта система является системой зависимых уравнений, т.к. одно из них является результатом линейных преобразований, выполненных над оставшимися уравнениями. Поэтому любое из них может быть исключено из дальнейшего рассмотрения, а оставшиеся уравнения дополнены нормировочным уравнением:

$$\sum_{j=1}^{4} \overline{p}_j = 1. \tag{11}$$

С помощью предельных вероятностей можно определить итоговую оценку знаний студента:

$$S_0 = 2\overline{p}_1 + 3\overline{p}_2 + 4\overline{p}_3 + 5\overline{p}_4. \quad (12)$$

Таким образом, нами разработан следующий алгоритм для оценки знаний студента:

1. Необходимо определить терм-множества состояний студентов $\tilde{S}$.

2. На основе отметок студента за определенный период времени разбить этот период на пересекающиеся подынтервалы, число которых совпадает с числом состояний студента.

3. По формуле (3) от временного ряда оценок необходимо перейти к временному ряду состояний студента.

4. Составляется последовательность импликаций состояний (формула (5)) и определяется частота переходов k_{ij} из i-го состояния в j-е $\left(i, j = \overline{1,4}\right)$.

5. Элементы стохастической матрицы M определяются из условия (7) и по формуле (8).

Если $\sum_{j=1}^{4} k_{ij} = 0,\ i = \overline{1,4}$, т.е. какие-то состояния не встречались в (5), то эту ситуацию будем считать ситуацией полной неопределенности и $p_{ij} = 0.25,\ i, j = \overline{1,4}$.

6. Для определения вероятностей стационарных состояний студента (9) решается векторно-матричное уравнение (10) с учетом условия (11).

7. Находится итоговая оценка знаний студента по формуле (12).

3. Апробация алгоритма для оценки знаний студента

В качестве примера работы предложенного алгоритма рассмотрим три временных ряда отметок успеваемости студента по 50-ти балльной шкале, за семестр (18 недель) (см. табл. 3).

Таблица 3

Успеваемость студентов

Студент	Баллы									Среднее
	1	1	2	4	5	6	7	8	9	
1	21	26	35	30	40	30	40	36	40	38,778
2	50	44	49	47	48	40	42	45	40	38,778
3	40	41	35	37	39	39	39	35	44	38,778

Окончание таблицы 3

Студент	Баллы									Среднее
	10	11	12	13	14	15	16	17	18	
1	46	45	38	39	48	47	49	38	50	38,778
2	40	36	41	30	37	24	34	21	30	38,778
3	36	38	44	37	41	38	40	44	31	38,778

Для студента 1 имеется тенденция к повышению баллов (см рис. 3)

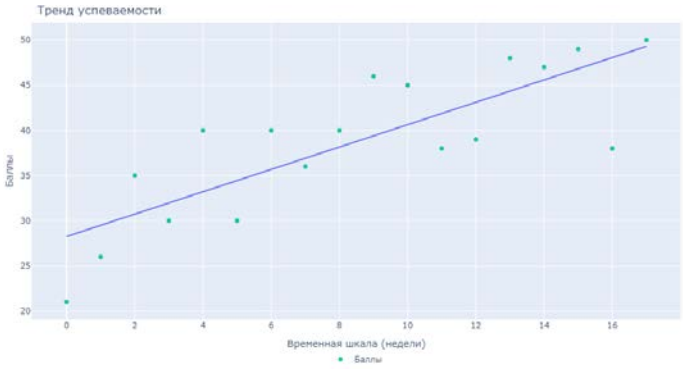


Рис. 3. Баллы и тренд успеваемости для студента 1

Для студента 2 характерно снижение среднего уровня оценок (см. рис. 4).

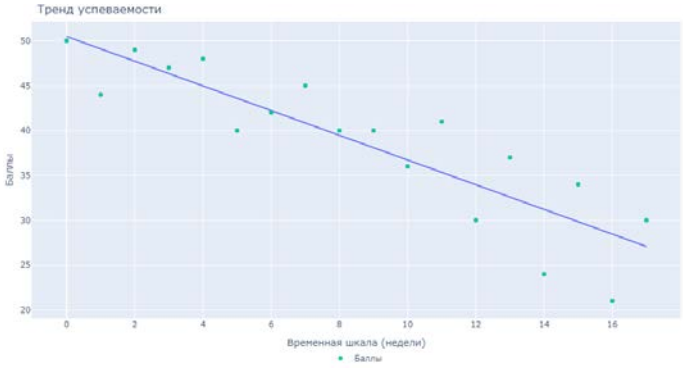


Рис. 4. Баллы и тренд успеваемости для студента 2

Студент 3 во время семестра учился равномерно с небольшими отклонениями (см. рис. 5).

У всех студентов при этом средний бал одинаковый – 38,778.

С помощью предложенного алгоритма найдем итоговую оценку знаний первого студента.

В соответствии с первым и вторым этапом алгоритма разобьём шкалу от 0 до 50 на четыре отрезка по количеству состояний студента, как показано на рис. 1.

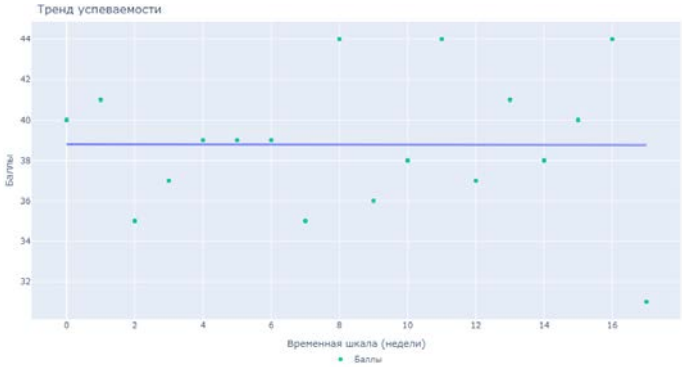


Рис. 5. Баллы и тренд успеваемости для студента 3

Временной ряд баллов у студента 1 примет вид:
X = (21; 26; 35; 30; 40; 30; 40; 36; 40; 46; 45; 38; 39; 48; 47; 49; 38; 50).

По формуле (3) перейдем от временного ряда баллов к временному ряду состояний первого студента, получим:

$$\tilde{S} = \begin{pmatrix} \tilde{S}_{1,1}, \tilde{S}_{2,2}, \tilde{S}_{3,3}, \tilde{S}_{4,2}, \tilde{S}_{5,3}, \tilde{S}_{6,2}, \tilde{S}_{7,3}, \tilde{S}_{8,3}, \tilde{S}_{9,3}, \\ \tilde{S}_{10,4}, \tilde{S}_{11,4}, \tilde{S}_{12,3}, \tilde{S}_{13,3}, \tilde{S}_{14,4}, \tilde{S}_{15,4}, \tilde{S}_{16,4}, \tilde{S}_{17,3}, \tilde{S}_{18,4} \end{pmatrix}.$$

Например, то, что в момент времени $t = 1$ наблюдалось состояние $\tilde{S}_{1,1}$, определили по формуле (3) следующим образом:

$$\tilde{S}_{1,1} : \max_j \mu_{\tilde{S}_j}(21) = \max\left\{\mu_{\tilde{S}_1}(21), \mu_{\tilde{S}_2}(21), \mu_{\tilde{S}_3}(21), \mu_{\tilde{S}_4}(21)\right\} =$$
$$= \max\{0.9, 0, 0, 0\} = \mu_{\tilde{S}_1}(21).$$

Следовательно, $x_1 = 21$ в большей мере соответствует состоянию $\tilde{S}_1$.

Последовательность импликаций состояний знаний первого студента будет следующей:

$$\tilde{S}_1 \to \tilde{S}_2 \to \tilde{S}_3 \to \tilde{S}_2 \to \tilde{S}_3 \to \tilde{S}_2 \to \tilde{S}_3 \to \tilde{S}_3 \to \tilde{S}_3 \to$$
$$\to \tilde{S}_4 \to \tilde{S}_4 \to \tilde{S}_3 \to \tilde{S}_3 \to \tilde{S}_4 \to \tilde{S}_4 \to \tilde{S}_4 \to \tilde{S}_3 \to \tilde{S}_4. \quad (13)$$

Посчитаем частоты переходов k_{ij} и запишем их в таблицу 4.

Таблица 4

Частоты переходов

	$\tilde{S}_1$	$\tilde{S}_2$	$\tilde{S}_3$	$\tilde{S}_4$
$\tilde{S}_1$	0	1	0	0
$\tilde{S}_2$	0	0	3	0
$\tilde{S}_3$	0	2	3	3
$\tilde{S}_4$	0	0	2	3

В соответствии с условием (7) и формулой (8) определим элементы стохастической матрицы:

$$M = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & P_{42} & p_{43} & p_{44} \end{pmatrix}.$$

Например, поскольку из состояния $\tilde{S}_1$ согласно (13) можно перейти только в состояние $\tilde{S}_2$, то из условия (7) следует, что элемент $p_{12} = 1$, а все остальные элементы в этой строке равны 0.

Так как из состояния $\tilde{S}_3$ можно перейти в состояния $\tilde{S}_2$, $\tilde{S}_3$, $\tilde{S}_4$, то

$$p_{32} == \frac{k_{32} \sum_{t=1}^{18} \mu_{\tilde{s}_2}(x_t)}{\sum_{j=1}^{4} k_{3j} \sum_{t=1}^{18} \mu_{\tilde{s}_j}(x_t)} = 0,25.$$

Аналогично находим остальные элементы матрицы M:

$$M = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0.25 & 0.375 & 0.375 \\ 0 & 0 & 0.4 & 0.6 \end{pmatrix}.$$

Для определения вероятностей стационарных состояний студента $\tilde{S}_1$, $\tilde{S}_2$, $\tilde{S}_3$ и $\tilde{S}_4$ (9) решается векторно-матричное уравнение (10) с учетом (11).

Для этого решим систему линейных алгебраических уравнений:

$$\begin{cases} (0-1)\overline{p}_1 + 0\overline{p}_2 + 0\overline{p}_3 + 0\overline{p}_4 = 0, \\ 1\overline{p}_1 + (0-1)\overline{p}_2 + 0.301\overline{p}_3 + 0\overline{p}_4 = 0, \\ 0\overline{p}_1 + 1\overline{p}_2 + (0.451-1)\overline{p}_3 + 0.548\overline{p}_4 = 0, \\ \overline{p}_1 + \overline{p}_2 + \overline{p}_3 + \overline{p}_4 = 1. \end{cases}$$

Решив данную систему, получим:

$$\overline{p}_1 = 0;\ \ \overline{p}_2 = 0.172;\ \ \overline{p}_3 = 0.57;\ \ \overline{p}_4 = 0.258.$$

По формуле (12) определяем итоговую оценку: 4.087

При этом, если считать успеваемость по среднему баллу, то он составит 38.778, что тоже соответствует оценке хорошо.

Аналогично рассчитаем итоговые оценки для студента 2 и студента 3 (табл. 3). Результаты расчётов представлены в табл. 5.

Таблица 5

Итоговая успеваемость студентов

Вероятность состояния	Студент 1	Студент 2	Студент 3
$\tilde{S}_1$	0	0.255	0
$\tilde{S}_2$	0.172	0.303	0
$\tilde{S}_3$	0.57	0.369	0
$\tilde{S}_4$	0.258	0.073	1
Итоговая оценка	4.087	3.26	3.857
Средний балл	38.778	38.778	38.778

Таким образом, разработанная система оценки знаний позволяет увидеть различия в успеваемости студентов, хотя, если считать по формуле средней арифметической, средний балл у них одинаковый.

Заключение

Широкое использование среднего балла как универсального параметра не оправдано, потому что имеет значительные ограничения, а также авторы исходят из того, что отметка не всегда является объективным способом оценивания знаний и способностей студента.

В ходе работы установлено соответствия между состояниями студента и временными рядами отметок, им полученных, с последующим прогнозированием этих состояний.

Разработан и реализован алгоритм для оценки знаний студента с использованием языка программирования Python и библиотек plotly, numpy, pandas, skfuzzy, matplotlib.

Разработанная система позволяет увидеть различия в успеваемости студентов.

Список литературы

1. Кокарева З.А. Оценочная деятельность в начальной школе : учебно-методическое пособие / З.А. Кокорева ; Вологда. изд-во ВИРО. 2007. – 108 с.

2. Заде Л.А. Понятие лингвистической переменной и его применение к принятию приближенных решений / Л. А. Заде -М. : Мир, 1976.-165 с.

3. Матвеев М.Г., Шмелев М.А., Алейникова Н.А. Информационные технологии формирования сервисов на электронной торговой площадке / М.Г Матвеев, М.А. Шмелев, Н. А. Алейникова. / Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2021. – № 1. – С. 63-73.

4. Пегат А. Нечеткое моделирование и управление / А. Пегат. — М.: Лаборатория знаний, 2020. — 801 с.

5. Ярушкина Н.Г. Интеллектуальный анализ временных рядов / Н.Г. Ярушкина, Т.В. Афанасьева, И.Г. Перфильева. -Ульяновск: УлГТУ, 2010. -320 с.

6. Каштанов, В. А. Случайные процессы / В. А. Каштанов, Н. Ю. Энатская. – М.: Издательство Юрайт, 2017. – 156 с.

Обзор методов оценки качества изображения

Д. Е. Семенихина

Студент магистратуры

А. Ю. Иванков

Доцент

Введение

В настоящее время большое количество информации поступает к нам через зрительный канал восприятия, что непосредственно влияет на развитие таких областей науки как цифровая обработка изображений и компьютерное зрение.

Для того, чтобы поддерживать и улучшать качество изображений важно, чтобы системы передачи, управления, сбора и обработки изображений оценивали его качество на каждом этапе.

Во многих системах обработки информации требуется высокий уровень детализации сцен, для чего используют алгоритмы повышения качества изображения, а именно алгоритмы сверхразрешения (СР). Под сверхразрешением понимают математические методы, позволяющие улучшить или восстановить изображение сверх физически достижимого.

Одна из нерешенных проблем в области цифровой обработки изображений – нахождение универсальных количественных критериев оценки качества изображения. До сих пор наиболее точными считаются субъективные критерии оценивания, которые являются недостаточно эффективными, так как требуют привлечения достаточного большого количества людей и затратны по времени.

Данная статья посвящена анализу существующих методов оценки качества изображения и направлена на выявление наиболее оптимальных алгоритмов оценивания.[1]

1. Классификация алгоритмов получения сверхразрешения

Алгоритмы сверхразрешения основаны на объединении повторяющейся информации из нескольких изображений с низким разрешением (НР) для формирования одного изображения с высоким разрешением (ВР).

СР-алгоритмы можно разделить на различные классы, в соответствии со следующими параметрами:

– область преобразования;
– количество входных изображений;
– технический принцип реализации.

Классификация алгоритмов получения сверхразрешения представлена на рис. 1.

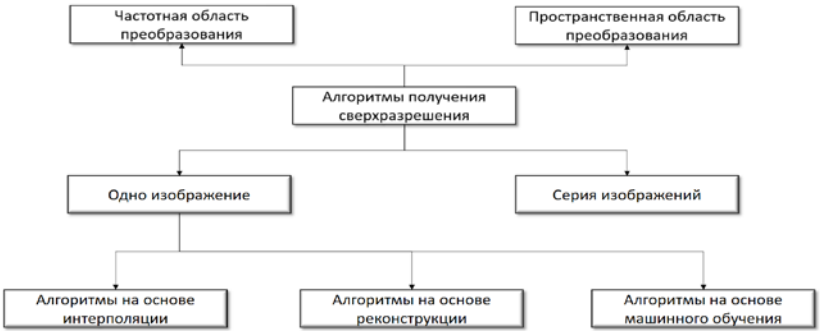


Рис. 1. Классификация алгоритмов получения сверхразршения

На практике получение серии изображений одной сцены довольно затруднительный процесс, поэтому СР-алгоритмы на основе нескольких изображений являются расширением алгоритмов, основанных на одном изображении.

Алгоритмы получения сверхразрешения на основе интерполяции подробно описаны в работах [1-4].

Алгоритмы получения сверхразрешения на основе реконструкции подробно описаны в работах [1, 5, 6].

На сегодняшний день большое внимание направлено на получение сверхразрешения с помощью применения методов глубокого машинного обучения (МО), в частности, свёрточных нейронных сетей и их различных модификаций. Данные методы описаны в статьях [1, 7-12]. Одной из сложностей применения алгоритмов МО является необходимость в достаточной большой обучающей выборки, которая должна содержать примеры анализируемых изображений.

2. Обзор методов оценки качества изображений

Под качеством изображениям будем понимать степень точности, с которой различные системы обработки информации захватывают, хранят, сжимают, передают и отображают сигналы, формирующие изображение.

Критерии оценки качества изображений можно разделить на два класса - субъективные и объективные, что отражено на рис. 2.

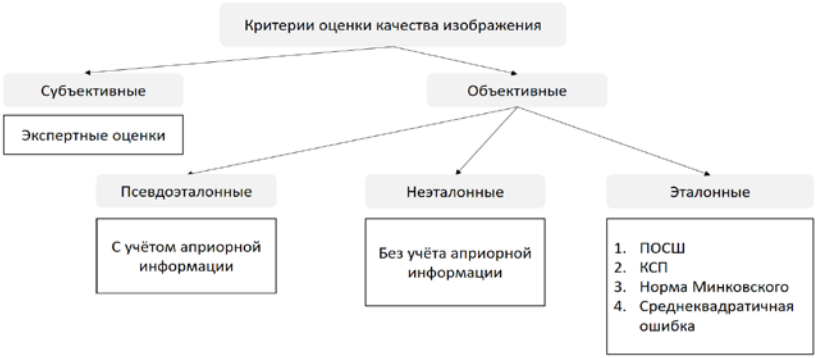


Рис. 2. Классификация методов оценки качества изображений

Субъективные критерии оценки качества изображения

Субъективный подход основан на использовании экспертных оценок и является одним из самых точных и надежных, так как в качестве эксперта выступает человек.

На результат субъективных оценок влияют различные факторы: например, условия эксперимента или характер рассматриваемых изображений. В связи с этим Международным Союзом Электросвязи (International Telecommunication Union) были созданы стандарты, которые описывают порядок проведения экспериментов для оценки качества изображений. К ним относятся:

– ITU-R BT.500-11;
– ITU-T P.910 ;
– ITU-R BT.814-1;
– ITU-R BT.1129-2.

В работах [13, 14] подробно описаны методы субъективной оценки изображений, которые можно классифицировать следующим образом: методы, которые основаны на предъявлении одиночных стимулов и методы, которые основаны на предъявлении пар стимулов.

К методам, в которых предоставляется одиночный стимул относятся:

– Абсолютный категориальный рейтинг;
– Абсолютный категориальный рейтинг со скрытым эталоном;
– Непрерывная оценка качества одиночных стимулов.

К методам, в которых предоставляется пара стимулов относятся:

– Категориальная оценка ухудшения;
– Метод оценки на основе парных стимулов с использованием непрерывной шкалы;
– Парное сравнение;
– Попарная оценка сходства;
– Непрерывная оценка качества на основе одновременных сдвоенных стимулов.

Несмотря на высокую точность субъективных оценок, данный подход обладает рядом недостатков: высокая стоимость работ, отсутствие объективности и универсальности.

Объективные критерии оценки качества изображения

Целью объективного подхода оценки качества изображений является разработка математических моделей, способных точно и автоматически предсказывать качество изображения. Идеальная объективная оценка должна имитировать прогнозы субъективной оценки человека.

Объективные метрики оценки качества изображений разделяют на три группы [15]**:**

– Эталонные (full-reference, FR);
– Псевдоэталонные (reduced-reference, RR);
– Неэталонные (no-reference, NR).

Эталонные метрики предполагают наличие исходного изображения, которое рассматривается как опорное или эталонное изображение при сравнении, так как оно не зашумлено и имеет идеальное качество. Наиболее популярные эталонные метрики:

– Среднеквадратичная ошибка;
– Пиковое отношение сигнал-шум;
– Норма Минковского;
– Мера структурного подобия.

Псевдоэталонные метрики основаны на использовании не самого эталонного изображения, а на ряде признаков, извлечённых из него. Априорная информация показывает силу определенного типа искажения исходного изображения. К искажениям можно отнести: сжатие, размытие, шум.

Алгоритмы оценки качества изображений на основе эталонных и псеводоэталонных метрик достаточно просты в реализации, но требуют наличия исходного изображения или априорной информации о нем. Результаты, которые получают при использовании данных методов, недостаточно коррелируются с субъективными оценками экспертов.

Неэталонные метрики основаны на знаниях о работе зрительной системы человека. Данные методы не используют исходное

изображение или априорную информацию о нём для получения оценки, что делает их достаточно универсальными. К неэталонным метрикам относятся [16-18]:

– Слепая/неэталонная оценка пространственного качества изображений (BRISQUE);
– Оценка качества естественных изображений (NIQE).

Метод BRISQUE прогнозирует оценку качества с помощью модели регрессии опорных векторов (SVR), обученной с использованием базы данных изображений, содержащей соответствующие значения дифференциальной средней оценки мнения (экспертные оценки).

Метод NIQE прогнозирует оценку качества с помощью модели многомерного гауссовского распределения. Оценка соответствует расстоянию между заданным изображением и моделью идеального изображения в пространстве признаков NSS. Признаки NSS основаны на вычислении нормализованных коэффициентов яркости в пространственной области и моделируются как многомерное распределение Гаусса. Искажения моделируются как возмущения гауссовсго распределения.

Отличительной особенностью алгоритмов, которые не учитывают априорную информацию, является необходимость процедуры предварительного обучения на специальных тестовых базах данных.

Обучение необходимо для нахождения функции, которая связывает значения признаков и оценку качества на выходе. Отыскание такой функции – задача регрессии, которая решается при помощи методов МО: машина опорных векторов, рандомизированные деревья, искусственные нейронные сети.

3. Базы тестовых изображений

На данный момент существуют тестовые базы, с помощью которых проводится обучение универсальных алгоритмов нахождения оценки качества изображения. Данные базы содержат изображения с различными типами искажений и соответствующие субъективные оценки качества, выделим наиболее популярные из них:

– LIVE Database;
– TID2013;
– TID2008;
– Toyama;
– IVC;
– CSIQ.

В работе [19] был проведён сравнительный анализ баз тестовых изображений, результаты исследования отображены на табл. 1.

Таблица 1

Сравнительный анализ эталонных баз тестовых изображений

№	Характеристика	База тестовых изображений					
		TID2008	TID2013	LIVE	Toyama	IVC	CSIQ
1	Количество изображений	1700	3000	779	196	235	866
2	Количество разных типов искажений	17	24	5	2	4	6
3	Количество участников экспериментов	846	971	161	16	15	35
4	Способ оценивания визуального качества	Относительная оценка		Абсолютная оценка по 5-бальной шкале			Относительная оценка
5	Количество оцениваний изображений в ходе эксперимента	255816	524340	25000	3136	3525	5000
6	Шкала полученных оценок MOS	0..9		0..100	1..5		0..1
7	Дисперсия оценок MOS	0,63	0,6	250	1,5	-	-
8	Относительная дисперсия оценок MOS	0,031	0,035	0,083	0,189	-	-

Исходя из данных таблицы можно сделать вывод, что TID2013 обладает наилучшими характеристиками: большее количество изображений и искажений.

Заключение

В ходе исследований выло выявлено, что наиболее универсальным объективным методом оценки изображений являются алгоритмы, использующие неэталонные метрики.

Работу данных алгоритмов можно разделить на два основных этапа: получение качественных признаков из изображения и их сравнение с соответствующей субъективной оценкой, которая хранится в специальных базах данных.

После процедуры обучения подобные алгоритмы потенциально способны оценить качество абсолютно любого изображения при условии, что рассматриваемый тип искажений присутствовал в базе, что делает его не полностью универсальным.

Список литературы

1. Huang ,D. A Short Survey of Image Super Resolution Algorithms/ D. Huang, H. Liu // Journal of Computer Science Technology Updates. – China, 2015. – Т. 2. – С. 19-29.
2. Schultz, R. A Bayesian approach to image expansion for improved definition/ R. Schultz, R. Stevenson // IEEE Transactions on Image Processing. – USA, 1994. – Т. 2. – С. 233-242.
3. Maeland, E. On the comparision of interpolation methods/ E. Maeland// IEEE Transactions on Medical Imaging. – USA, 1988. – Т. 1. – С. 213-217.
4. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М. : Техносфера, 2005. – 1072 с.
5. Super-Resolution of Wavlet-Encoded Images [Электронный ресурс]. – Режим доступа: https://arxiv.org/pdf/1705.01258.pdf/
6. Artifact Reduction for Set Theoretic Super Resolution Image Reconstruction with Edge Adaptive Constraints and High-Order Interpolants [Электронный ресурс]. – Режим доступа: https://citeseerx.ist.psu.edu /viewdoc/download?doi=10.1.1.162.1445&rep=rep1&type=pdf/
7. Super-Resolution from a Single Image [Электронный ресурс]. – Режим доступа: https://www.wisdom.weizmann.ac.il/~vision/ single_image_SR/files/single_image_SR.pdf/
8. Саввин, С. В. Алгоритмы построения многокадрового сверхразрешения изображений в условиях аппликативных помех на основе глубоких нейронных сетей / С. В. Саввин, А. А. Сирота // Компьютерная оптика. – 2022. – Т. 46, № 1. – С. 130-138.
9. Super-Resolution through neighbor embedding [Электронный ресурс]. – Режим доступа: https://ieeexplore.ieee.org/document/1315043/
10. Coupled dictionary training for image super-resolution in [Электронный ресурс]. – Режим доступа: https://citeseerx.ist.psu.edu /viewdoc/download?doi=10.1.1.309.1345&rep=rep1&type=pdf/
11. Nonparametric blind super-resolution [Электронный ресурс]. – Режим доступа: https://openaccess.thecvf.com/content_iccv_2013/papers /Michaeli_Nonparametric_Blind_Super-resolution_2013_ICCV_paper.pdf/
12. Fast image image super-resolution based on in-place example regression [Электронный ресурс]. – Режим доступа: https://openaccess.thecvf.com/content_cvpr_2013/papers/Yang_Fast_Image_ Super-Resolution_2013_CVPR_paper.pdf/

13. Оценки качества для анализа цифровых изображений [Электронный ресурс]. – Режим доступа: http://uiip.bas-net.by/structure/l_ori/starovoitov/Starovoitov_Publication_section/15_Starovoitov.pdf/

14. Субъективная оценка качества изображений: методологический обзор/ М.А. Грачева [и др.] // Сенсорные системы. – 2019.– Т. 33, № 4. – С. 287-304.

15. Subjective and Objective Quality Assessment of Image: A Survey [Электронный ресурс]. – Режим доступа: https://arxiv.org/abs/1406.7799/

16. Image Super-Resolution Quality Assessment: Structural Fidelity Versus Statistical Naturalness [Электронный ресурс]. – Режим доступа: https://arxiv.org/abs/2105.07139/

17. Learning a no-reference quality metric for single-image super-resolution [Электронный ресурс]. – Режим доступа: https://www.sciencedirect.com/science/article/abs/pii/S107731421630203X/

18. Learning-Based Quality Assessment for Image Super-Resolution [Электронный ресурс]. – Режим доступа: https://arxiv.org/abs/2012.08732/

19. Интегрированная метрика визуального качества изображений [Электронный ресурс]. – Режим доступа: https://dropdoc.ru/doc/347678/integrirovannaya-metrika-vizual._nogo-kachestva-izobrazhenij/

Проектирование и разработка технологического стека для обеспечения релизного процесса в рамках облачной разработки для крупного корпоративного решения

М. В. Старкин

Студент магистратуры

С. В. Власов

Доцент

Введение

В процессе разработки на одном большом проекте могут участвовать множество команд, например, такие как, разработчики, инженеры по тестированию, DevOps и релиз-инженеры.

В процессе поддержки и разработки программного продукта требуется фиксировать определенные версии продукта, необходимые для различных команд. Это могут быть новые релизные версии, версии, несущие в себе срочные исправления, версии, имеющие в себе новый определенный набор изменений. Если продукт является микросервисным, и микросервисов в его составе достаточно много, процесс ручного отведения релиза и сопутствующих операций, например, контроль версий, становится очень затратным по таким критериям, как время и ресурсы. Если релизную версию проекта требуется отводить раз в определенный достаточно большой промежуток времени, и это можно реализовать с помощью ручного труда релиз-инженеров, то версии с исправлениями, наоборот, должны выпускаться постоянно, и в последствии они должны устанавливаться на продуктовое оборудование, оборудование инженеров по тестированию, разработчиков.

Если в компании нет своих инструментов, оптимизирующих данные процессы, то компании требуется достаточно большой штат релиз-инженеров, что подразумевает большую статью расходов. Также отсутствие инструментов по автоматизации данных процессов оставляет возможность человеческой ошибки, что в данном случае может привести к определенным потерям из-за необходимости воспроизвести какие-то части процессов повторно.

Также, если в рамках проекта не налажен процесс передачи версий разрабатываемого продукта между командами, в особенности между инженерами по тестированию и инженерами по разработке, это может приводить к ошибкам, например, таким, как наличие программных ошибок в продукте, которые были уже исправлены, но данные исправления не попали в версию, которую установили на тестовый стенд.

Чтобы наладить процесс передачи новых версий продукта от разработчиков к инженерам по тестированию для последующего тестирования и сделать его более прозрачным, была поставлена задача спроектировать и разработать систему, которая позволяла бы отслеживать поток разработанных, протестированных, а также выданных заказчику версий.

1. Текущее состояние системы

Для дальнейшего описания требуется определить некоторые основные понятия и технологии, которые будут использоваться в дальнейшем.

Пайплайн – последовательность выполнения стадий, каждая из которых включает несколько задач [1]. Docker – программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации. Релиз – версия продукта, которая является итогом некоторого отрезка разработки, содержащая в себе обновление кодовой базы продукта.

Перед тем как начать проектирование новой системы, необходимо провести анализ текущего состояния системы и определить основные проблемы.

Разработка и конфигурация продукта происходит в рамках трех основных этапов, уровней (рис. 1).

Первый уровень – уровень репозитория, то есть непосредственно кодовая база микросервисов. В результате их сборки через удаленный инстанс Jenkins происходит генерация Docker-образов, использующихся в дальнейшем. Далее идет второй уровень - проектный. Это также репозиторий в используемой системой управления репозиториями кода, который содержит в себе указания на все микросервисы, из которых состоит продукт. В текущем варианте используется прямое указание на версии используемых микросервисов – на конкретные версии Docker-образов. Это один из недостатков текущей системы, так как при каждом изменении в сервисе необходимо обновить Docker-образ в данном репозитории. Но с другой стороны это предоставляет определенный плюс в том, что сборка с определенного коммита всегда даст

одинаковый результат, тем самым, обеспечивая воспроизводимость определенного состояния [1].

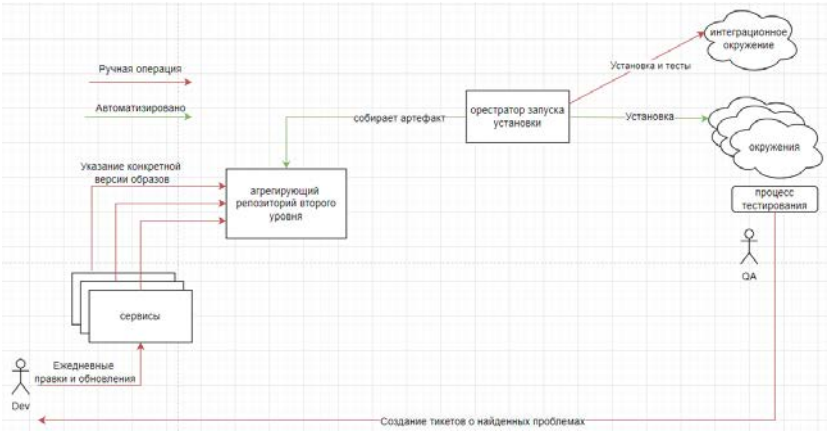


Рис. 1. Схематичное упрощенное представление первых двух уровней

Этот метод предоставляет возможность определения, какие образы были использованы путем простого просмотра файлов внутри репозитория. Результатом сборки этого агрегирующего сервиса является новая версия манифеста приложения-продукта, который и будет устанавливаться через удаленный инстанс Jenkins в некоторое PaaS окружение. Третий уровень – сборка итогового артефакта, который будет выдан заказчику. Включает в себя информацию о версии продукта, то есть манифест, который был получен на втором уровне, а также все версии инфраструктурных сервисов. Инфраструктурная часть продукта не является темой данной работы.

Рассмотрим основные проблемы, которые есть в системе. Первая из них - отсутствие прозрачности передачи версии от разработчиков к инженерам по тестированию. Проблема заключается в том, что в данный момент процесс взаимодействия является таковым, что инженеры по тестированию устанавливали на свои окружения не конкретную версию продукта, которую им указали, а состояние агрегирующего репозитория в какой-то момент, и не всегда она совпадала по версиям микросервисов с версией продукта, которую разработчики считали стабильной. Также, как следствие этой проблемы, отсутствовал стабильный процесс передачи фиксированной протестированной версии дальше на третий уровень.

Второй основной проблемой является отсутствие автоматизации по отведению релиза для данного продукта. Данная проблема накладывает дополнительную нагрузку на DevOps-инженера проекта по отведению новых релизных веток для сервисов, обновление конфигурационного репозитория и другие операции, которые необходимо провести на данном технологическом стеке.

2. Проектирование новой системы

При анализе были выявлены основные две проблемы – отсутствие прозрачности выдачи версий продукта разработчиками инженерам по тестированию, и также отсутствие автоматизации по релизным процессам, что приводило к дополнительной нагрузке DevOps-инженера.

Была спроектирована и разработана следующая система. Ее схематичное упрощенное представление с компонентами, процессами и действующими пользователями, которые будут взаимодействовать с данной системой, представлена на рис. 2.

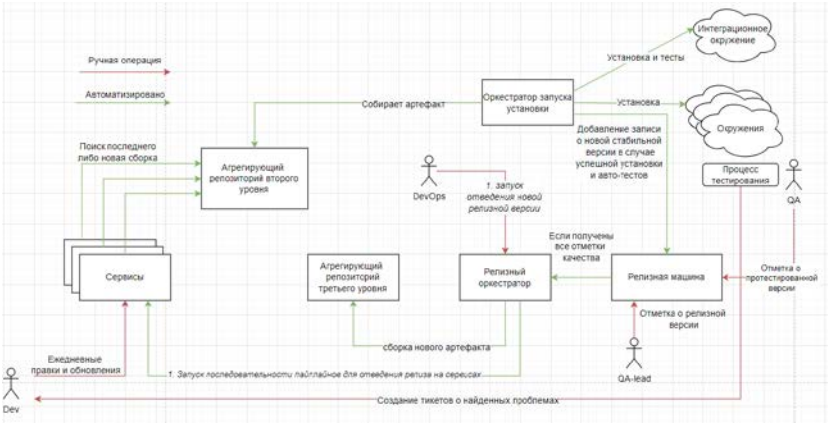


Рис. 2. Схематичное упрощенное представление новой системы

Для стабилизации выдачи новых версий на проекте была введена ежедневная ночная установка всего продуктового решения в тестовое окружение. На первом этапе это ввелось при помощи базовых возможностей распределенной системы хранения кода Gitlab, которая предоставляет возможность запуска пайплайнов в определенный отложенный момент времени при помощи расписания [1]. Пайплайн запускался на определенном сервисе, который был разработан ранее. Он является одной из конфигурационных точек установки продукта, а

также выполняет роль оркестратора установки. Также при каждой ночной установке после успешного разворота запускались автотесты на данном окружении для проверки тесткейсов, которые были автоматизированы. Если установка проходила все этапы успешно, версию продукта, которая была установлена, можно было считать стабильной и выдавать инженерам по тестированию для полного цикла тестирования, также включающее в себя мануальные шаги. Это было принято считать первой качественной проверкой. Необходимо было где-то сохранять такие стабильные версии. Для того, чтобы хранить информацию о том, что версия продукта прошла какие-либо стадии, было принято решение разработать компоненту, которая могла бы хранить эту информацию. Для первой версии была использована упрощенная схема. Была создана дополнительная компонента, которая содержала в себе файл, который содержал в себе записи о версиях продукта, а также их качественных отметках. Часть полей в этих записях проставлялось самой системой в автоматическом режиме, часть же полей должно было проставляться определенными пользователями. Если версия получала отметки во всех полях, которые отвечали за качественные проверки, то запускалась автоматически сборка артефакта третьего уровня с включенным в него версией продукта.

Для решения проблемы автоматизации релизных процессов были использованы следующие инструменты компании, разработанные ранее:

1. Оркестратор – сервис, задачей которого является для списка сервисов построение дерева зависимостей и в порядке от листов к корню этого дерева запускать пайплайны для сервисов.
2. Unified-service-pipeline – компонента, содержащая реализацию большинства процессов жизненного цикла для микросервиса.

3. Реализация

Сервис, обеспечивающий хранение версий, представлен репозиторием на Gitlab, содержащим конфигурационные файлы, а также некоторую логику для поддержания корректного состояния репозитория.

Для отображения таблицы с версиями используется встроенная функциональность Gitlab - Wiki, представляющая возможность хранить информацию/документацию, привязанную к репозиторию в формате отчетов. Пользователи взаимодействуют с сервисом посредством запуска пайплайнов для этого репозитория. Все возможные действия представлены набором стадий, описанных в gitlab-ci.yaml файле. Каждая их этих стадий вызывала выполнение некоторой команды release-cli.

Applications

Manifest	Stable deploy	Auto test	QA-tested	Ready to dvm	Manifests expire date
		+			2022-04-23
	+				2022-04-02

Рис. 3. Пример таблицы с некоторыми полями

Варианты поставки разработанного решения

Разработанное решение может поставляться двумя вариантами:

1. Установка релизной cli при помощи пакетного менеджера pip3
2. Docker-образ с предустановленной релизной cli, а также всеми необходимым техническими средствами для обеспечения работы релизной машины.

Заключение

Проведенный анализ, проектирование и разработка позволяют повысить эффективность системы, ввести определенные пункты качества, зафиксировать роли пользователей, повысить управляемость процессов за счет ускорения и автоматизации процессов, а также введения непрерывного релизного процесса на проекте.

Дальнейшее развитие можно продолжать в следующих направлениях: увеличение количества автоматизируемых процессов и дальнейшее развитие компоненты, хранящей информацию о версиях продукта, в сторону повышения надежности, например, за счет использования базы данных.

Список литературы

1. Документация Gitlab, Gitlab CI/CD Pipeline Configuration [Электронный ресурс]: сайт. – Режим доступа : https://docs.gitlab.com/ee/ci/yaml/README.html

Качественная оценка настроения музыкальных произведений

Е. А. Стеблева

Студент магистратуры

А. Ю. Иванков

Доцент

Введение

Музыка — это мощный язык для выражения наших чувств. Эмоции и настроения легко отражаются в музыке. Когда мы занимаемся спортом, мы склонны слушать энергичную музыку, точно так же, когда мы взволнованы или устали, приятная расслабляющая песня может помочь нам успокоиться.

Классификация музыки является довольно сложно задачей, поскольку эмоциональная реакция разных слушателей на одну и ту же композицию может быть разной. Большая часть алгоритмов современной классификации музыки основана на определении общего жанра исполнителя, а не на чувстве, порождаемой композицией. Автоматическое определение музыки было бы чрезвычайно полезно для сортировки больших объемов цифровой музыки (Itunes, Spotify), а также для определения похожих песен для онлайн-радиосервисов (Pandora), основываясь на сходстве настроений музыки, а не на похожих исполнителях.

Данная статья посвящена разработке алгоритма количественной оценки настроения музыкальных произведений. Основными задачами исследования являются: исследование известных методов анализа музыкальных композиций, их тестирование и сравнение результатов, а также разработка своего алгоритма количественной оценки музыки.

1. Традиционная модель настроения

Одной из самых известных моделей классификации настроения является модель психолога Роберта Тайера [1]. В большинстве существующих методов классификации музыкального настроения используется именно она. Модель делит песни на 8 категорий от счастливых до грустных и от спокойных до энергичных.

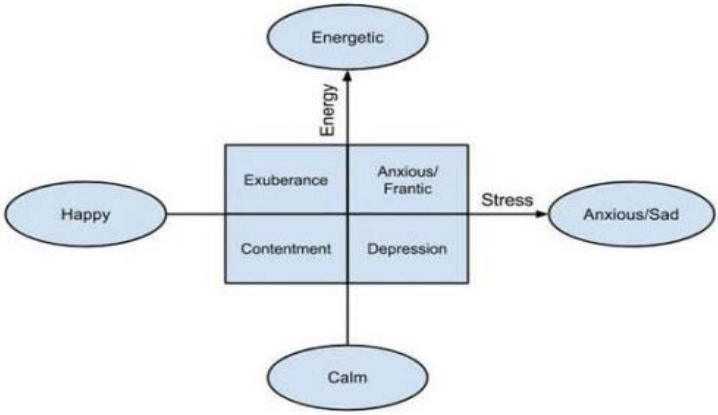


Рис. 1. Традиционная модель настроения

На рис. 2 представлены все 8 категорий модели с указанием музыкальных компонентов, которые описывают каждую категорию. Так, например, энергичные и счастливые песни более интенсивны и выше по тону, чем грустные и спокойные.

Mood	Intensity	Timbre	Pitch	Rhythm
Happy	Medium	Medium	Very High	Very High
Exuberant	High	Medium	High	High
Energetic	Very High	Medium	Medium	High
Frantic	High	Very High	Low	Very High
Anxious/Sad	Medium	Very Low	Very Low	Low
Depression	Low	Low	Low	Low
Calm	Very Low	Very Low	Medium	Very Low
Contentment	Low	Low	High	Low

Рис. 2. Характерные особенности настроений музыкальных композиций

2. Формирование обучающей выборки

На основе традиционной модели настроения Роберта Тайера в качестве основных категорий было принято решение взять следующие: «энергичность», «спокойствие», «веселье», «грусть».

Для подготовки датасета использовался стриминговый сервис Spotify. Данный сервис удобен тем, что музыка в нем уже классифицирована не только по настроению, но и по некоторым другими характеристикам:

1. Акустика: показатель от 0.0 до 1.0 того, является ли трек акустическим.

2. Танцевальность: описывает, насколько от 0.0 до 1.0 трек подходит для танцев, основываясь на сочетании различных музыкальных элементов, включая темп, стабильности ритма и общую регулярность.

3. Энергичность: представляет собой меру от 0.0 до 1.0 того, насколько трек интенсивен и активен. Как правило, энергичные треки кажутся быстрыми и громкими. Характеристики восприятия, влияющие на это, включают в себя воспринимаемую громкость, тембр, динамический диапазон, скорость начала трека.

4. Инструментальность: предсказывает, содержит ли трек вокал. Различные звуки в этом контексте рассматриваются как инструментальные. Чем ближе значение инструментальности к 1.0, тем больше вероятность того, что трек не содержит вокального содержания.

5. Живость: более высокие значения представляют собой повышенную вероятность того, что трек был исполнен вживую.

6. Громкость: общая громкость дорожки в децибелах (дБ). Значения громкости усредняются по всей дорожке и нужны для сравнения относительной громкости дорожек.

7. Наличие речи: определяет наличие произносимых слов в дорожке. Чем больше запись похожа на речь (например, ток-шоу, аудиокнига, поэзия), тем значение атрибута ближе к 1.0. Значения выше 0.66 описывают треки, которые, вероятно, полностью состоят из произнесенных слов. Значения от 0.33 до 0.66 описывают треки, которые могут содержать как музыку, так и речь, например треки в стиле рэп. Значения ниже 0.33, скорее всего, представляют треки, в которых нет слов.

8. Валентность: мера от 0.0 до 1.0, описывающая музыкальную позитивность, передаваемую треком. Треки с высокой валентностью звучат более позитивно, в то время как треки с низкой валентностью звучат более негативно.

9. Темп: общий расчетный темп дорожки в ударах в минуту (BPM). В музыкальной терминологии темп — это степень быстроты музыкального произведения.

mood	popularity	length	danceability	acousticness	energy	instrumentalness	liveness	valence	loudness	speechiness	tempo	key	time_signature
Calm	40.47	193201.170	0.408640	0.882410	0.155783	0.890175	0.117788	0.150011	-20.934005	0.040607	108.892180	5.47	3.760
Energetic	42.42	213207.345	0.538110	0.030978	0.870085	0.124303	0.234581	0.403070	-4.962845	0.076633	130.375665	5.28	3.950
Happy	47.48	222282.345	0.619370	0.109826	0.780095	0.128872	0.220174	0.503411	-6.564875	0.063018	124.692345	5.03	3.970
Sad	37.78	247505.235	0.495431	0.577900	0.390422	0.201679	0.140375	0.286996	-10.542255	0.041858	115.985005	5.56	3.835

Рис. 3. Классификация музыкальных композиций на основе данных Spotify

3. Модель классификации музыкальных композиций

В качестве модели классификации музыки было принято решения выбрать обычную сверточную нейронную сеть. Входной слой модели принимает 10 признаков, затем идет слой с 8 узлами и на выходе 4 узла. Функция активации - Relu, функция потерь является логистической функцией, а алгоритм градиентного спуска Адама является оптимизатором. Важно отметить, что модель обучалась на 640 выборках (80% основных данных).

```
640/640 [==============================] - 0s 30us/sample - loss: 0.5575 - acc: 0.7437
Epoch 290/300
640/640 [==============================] - 0s 45us/sample - loss: 0.5572 - acc: 0.7437
Epoch 291/300
640/640 [==============================] - 0s 39us/sample - loss: 0.5568 - acc: 0.7469
Epoch 292/300
640/640 [==============================] - 0s 47us/sample - loss: 0.5563 - acc: 0.7453
Epoch 293/300
640/640 [==============================] - 0s 44us/sample - loss: 0.5559 - acc: 0.7453
Epoch 294/300
640/640 [==============================] - 0s 45us/sample - loss: 0.5556 - acc: 0.7453
Epoch 295/300
640/640 [==============================] - 0s 34us/sample - loss: 0.5551 - acc: 0.7422
Epoch 296/300
640/640 [==============================] - 0s 33us/sample - loss: 0.5546 - acc: 0.7422
Epoch 297/300
640/640 [==============================] - 0s 30us/sample - loss: 0.5542 - acc: 0.7437
Epoch 298/300
640/640 [==============================] - 0s 31us/sample - loss: 0.5537 - acc: 0.7437
Epoch 299/300
640/640 [==============================] - 0s 36us/sample - loss: 0.5532 - acc: 0.7406
Epoch 300/300
640/640 [==============================] - 0s 33us/sample - loss: 0.5529 - acc: 0.7406
```

Рис. 4. Обучение модели классификации музыки

4. Результаты классификации

Для более наглядной демонстрации результатов классификации на рис. 5 представлена матрица ошибок [2]. Можно заметить, что с точностью в 74% модель хорошо классифицирует спокойные и грустные песни, однако с классификацией энергичных и веселых песен есть некоторые проблемы.

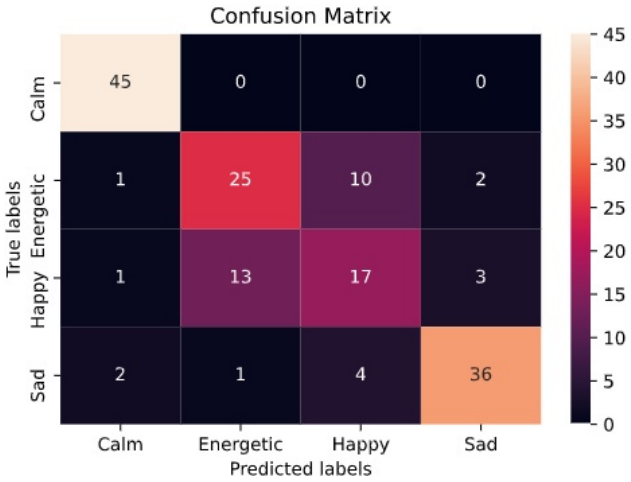


Рис. 5. Матрица ошибок

На рис. 6 представлен результат работы классификатора на конкретном примере.

Рис. 6. Результат работы классификатора

Заключение

В результате данного исследования были проанализированы различные модели классификации музыки. Была построена собственная модель сверточной нейронной сети для классификации музыкальных произведений. Точность модели достигла 74%. С помощью Spotify URI [3] можно классифицировать любую композицию, которая доступна на сервисе Spotify.

Выделив музыкальные компоненты, которые поддаются количественному измерению (ритм, гармония, тембр и др.) позволит нам сопоставить произведения с определенными категориями на основе ожидаемых данных для каждого типа настроения.

Список литературы

1. Robert E. Thayer The Origin of Everyday Moods: Managing Energy, Tension, and Stress / Robert E. Thayer // Oxford University Press, 1997. – 290p.

2. Confusion matrix [Электронный ресурс] – Режим доступа: https://en.wikipedia.org/wiki/Confusion_matrix

3. Spotify URI [Электронный ресурс] – Режим доступа: https://community.spotify.com/t5/FAQs/What-s-a-Spotify-URI/ta-p/919201

Реализация и исследование алгоритмов трассировки соединений на основе методов клеточной декомпозиции

А. В. Степанова

Студент магистратуры

М. А. Дрюченко

Доцент

Введение

На сегодняшний день одной из важных задач в областях: трассировки электронных цепей, навигации роботов или игроков на картах, размещения взаимосвязанных объектов на плоскости является планирование пути. В ней заключается 3 главных аспекта. Первое – построенный или спланированный путь должен соединять заданные начальную и конечную точки. Второе – он не должен наилучшим образом обходить имеющиеся препятствия. И третье – он должен являться оптимальным в смысле наличия изломов, его длины по сравнению с остальными возможными путями.

Существует большое число классификаций методов планирования пути. Например, рассматривая контекст интеллектуальных технологий, их делят на эвристические и традиционные методы. Если же рассматривать среду, в которой требуется проложить путь, то бывают методы планирования пути в статистической окружающей среде и динамической (объекты могут перемещаться на карте или лабиринте со временем). Так же существует классификация по полноте известной информации об окружающей среде: методы с полной и неполной информацией. В первом случае речь идёт о глобальном планировании, во втором – о локальном.

В данной работе рассмотрим группу методов, в основе которых лежит клеточная декомпозиция карты.

1. Методы на основе клеточной декомпозиции

В основе данного класса лежит задача дискретизации окружающей среды. Примерами таких методов являются следующие алгоритмы:

- Алгоритм Дейкстры;
- A*;
- D*;

– Theta*;
– Метод распространения волнового фронта.

Существует две группы идей реализации декомпозиции пространства: приближённая и точная. Первая идея реализуется с помощью сетки, покрывающей карту. Эту идею предложил Эльф [1], она заключается в разделении пространства на клетки одинакового размера, каждая из которой содержит в себе информации о проходимости (0 – свободна, 1 – есть препятствие). Сетки просты в использовании, их легко создавать и поддерживать. Но существуют и недостатки. Например, увеличение трудоёмкости при уменьшении шагов сетки, он становится ощутим при работе в среде большого объёма. Для решения этой проблемы предложены подходы, основанные на нерегулярных сетках, например, дерево квадрантов в двумерном случае и дерево кубов в трёхмерном [2]. Вторая идея – точной клеточной декомпозиции заключается в делении окружающей среды на клетки, пользуясь гранями препятствий. Поэтому здесь важную роль играет решение задачи разложения многоугольника на выпуклые области. Известными подходами точной клеточной декомпозиции являются: вертикальное и трапециевидное разложения [3].

В данном классе методов, как и в методах на основе графов, необходим дополнительный этап построения пути, заключающийся в поиске пути на графе. Используются разные стратегии поиска относительно оптимального решения. Они делятся на эвристические (информированные) и на классические (неинформированные). К классическим относятся: поиски в ширину и глубину и их модификации, поиск по критерию стоимости, двунаправленные поиск.

Хоть и эвристические стратегии не имеют строгого обоснования, они дают оптимальное решение в большинстве практических задач. Их особенностью является формирование эвристических правил, которые могут хорошо работать для одной задачи и давать плохой результат для другой. Поэтому их формулируют в соответствии с конкретной задачей.

Наиболее широко распространённым и известным является алгоритм A*, который был описан в 1968 году Питером Хартом, Нильсом Нильсоном и Бертрамом Рафаэлем. Он является расширение алгоритма Дейкстры, который был предложен в 1959 году. Преимущество A* заключается в более высокой производительности за счёт использования эвристик. В нём реализуется целевая функция, являющаяся суммой функции стоимости достижения рассматриваемой точки из начальной и эвристической функции оценки расстояния от рассматриваемой точки до конечной точки.

Существует весомый недостаток алгоритмов на основе клеточной декомпозиции – ограниченность вариантов направления перемещения объекта, вызванная структурой сетки. Для решения данной проблемы были предложены алгоритмы D* [4] и Theta* [5]. D* или по-другому динамический A* был опубликован в 1993 году [6] и представляет собой доработку алгоритма A* с помощью перепланировки. Он использует меньшее число клеток за счёт того, что в нём не планируется сразу весь путь до конечной точки. В работе [7] D* улучшен путём сокращения количества расширенных узлов, сокращения времени работы. Примерами алгоритмов, которые используют динамическую перепланировку, являются Lifelong Planning A* (LPA*) [8] и D* Lite [9].

Одним из применяемых алгоритмов на основе клеточных декомпозиции является метод распространения волнового фронта. Его особенность заключается в отсутствии поиска [10].

Принцип работы данного алгоритма следует из его названия. Реализуется распространение волны во всех направлениях от начальной точки, словно как от камня, брошенного в воду. В его модификации Fast Marching Method (FMM) сначала рассчитывается время $T(X)$ как функция точки X, за которое волновой фронт достигает конечную точку, распространяясь от исходной. Функция $T(X)$ является решение уравнения в частных производных

$$|\nabla T(X)|\,F(X)=1, \tag{1}$$

называемого уравнением эйконала [11], с начальным условием $T(X_0)=0$, где X_0 – начальная точка распространения волны, $F(X)$ – скорость распространения волнового фронта в точке X. После нахождения решения уравнения, траектория из точки $\mathbf{X}$ может быть найдена как кривая наискорейшего спуска по поверхности

$$W=T(X) \tag{2}$$

одним из методов спуска.

Рассмотрим подробнее алгоритмы A* и волновой алгоритм Ли, и сравним результат и время работы в одинаковых условиях.

2. Волновой метод (метод Ли)

Данный алгоритм принадлежит классу методов на основе клеточной декомпозиции, таким образом это алгоритм поиска кратчайшего пути на планарном графе (дискретном рабочем поле). Идея работы алгоритма основывается на методе поиска в ширину, то есть волна распространяется постепенно от первой ячейки, посещая все

соседние на следующем уровне, если они уже не отмечены как посещённые. Он предназначен для нахождения кратчайшего пути от стартовой ячейки к конечной ячейке, если это возможно. В противном случае он информирует о неудаче, например если путь к целевой точке заблокирован препятствиями.

Данный алгоритм лучше всего использовать, когда объекты находятся рядом на карте, поскольку волна распространяется во всех направлениях и требует больших ресурсов памяти для своей работы.

Работа алгоритма состоит из двух частей. В первой – от источника к приемнику распространяется волна. Во второй выполняется обратный ход (восстановление кратчайшего пути), в процессе которого из ячеек волны формируется путь. Волна, идущая от источника к приемнику, на каждом шаге первой части алгоритма пополняется свободными ячейками дискретного рабочего поля, которые, во-первых, еще не принадлежат волне, и, во-вторых, не являются одним из четырёх соседей ячеек, которые уже попали в волну на предыдущем шаге. При обратном ходе в итоговый путь включается по одной ячейке каждого шага распространения волны. При выборе какую из ячеек добавить в конечное решение, приоритет отдаётся той ячейке, для которой не требуется менять направления маршрута. Так же в простейшем случае критерием оптимальности маршрута является его минимальная длина. То есть при построении маршрута целевая функция представляет собой инкремент значений весов ячеек соседних с ячейками предыдущего фронта волны.

В реальных задачах, например при трассировке печатных плат, проложение пути (трассы) выполняется многократно, что влечет за собой существенные временные затраты [12].

Рассмотри последовательность шагов волнового алгоритма:

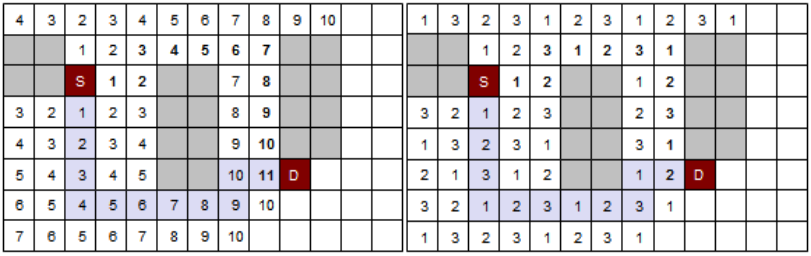
1. На первом этапе помечаем стартовую ячейку как посещённую и присваиваем ей нулевое расстояние $d = 0$;

2. Далее для каждой ячейки, отмеченной числом d, присваиваем всем соседним проходимым, еще не посещённым ячейкам число $d+1$, до тех пор, пока не достигнута целевая точка и есть еще не посещённые свободные ячейки;

3. В случае если достигнута финишная ячейка, то переходим в неё и начинаем обратное движение ;

4. Обратный проход осуществляется следующим образом: выбирается среди соседних ячеек, такая, в которой значение д принимает меньшее значение среди остальных и на 1 меньше, чем в текущей. Итерации продолжаются, пока не будет достигнута стартовая точка.

Графическая интерпретация работы волнового алгоритма приведена на рисунке. Стартовая ячейка обозначена «S», конечная – «D», темные ячейки соответствуют препятствиям.



а) б)

Рис. 1. Графическая интерпретация работы волнового алгоритма

Достоинствами волнового алгоритма являются:

– высокая эффективность нахождения трасс за счет исследования всех свободных ячеек рабочего поля;

– оптимальность найденных трасс по длине;

– возможность ранней остановки алгоритма в случае столкновения волны с ячейкой, соответствующей вертикальному или горизонтальному сегменту из той же цепи, что и трассируемый проводник (трасса в этом случае восстанавливается от точки столкновения).

Что касается недостатков, то среди них следует отметить существенные временные затраты при поиске трасс на больших схемах, поскольку алгоритм тратит время на исследование всех направлений, даже бесперспективных. В ПК волновой алгоритм применяется для построения трасс при интерактивном редактировании чертежей для ограниченного числа элементов.

3. Алгоритм A* («A-star»)

Так как при поиске пути сиз точки в точку при значительном удалении данных точек волновой алгоритм оказывается избыточным. На практике для уменьшения числа вычислений применяются различные эвристические функции, позволяющие задать приоритет расширения границ – в направлении к цели больше, чем в других направлениях.

Одним из самых распространенных алгоритмов эвристического поиска, почти всегда гарантирующим отыскание оптимального решения, является алгоритм A*. Он является классическим примером алгоритма поиска пути минимальной стоимости между двумя

вершинами на взвешенном графе, реализующий стратегию перемещения «в порядке ухудшения», т.е. первую очередь он посещает наиболее перспективные вершины, которые потенциально принадлежат кратчайшему пути к цели. По аналогии с рассмотренным выше волновым алгоритмом узел графа представляет объект, находящийся в определенной ячейке рабочего поля, и передвижение в соседние клетки соответствует перемещению в соседние состояния или смежные узлы [13].

Использование эвристик позволяет алгоритму A* не посещать узлы графа, которые, скорее всего, не ведут к оптимальному решению, что позволяет отыскивать решения быстрее и с меньшими затратами памяти. Рассчитываемая в процессе выполнения алгоритма эвристическая функция имеет вид:

$$F(v) = f_S(v) + f_D(v) + \sum_{i=1}^{r} h_i(v), \qquad (3)$$

где $f_S(v)$ – минимальная стоимость маршрута из начальной вершины в текущую вершину v, $f_D(v)$ – приближение стоимости маршрута от v до целевой вершины, h_i – дополнительные критерии оптимальности маршрута, которые могут оценивать удаленность v от центральной линии между элементами, удаленность v от других соединений, приоритет текущего направления и т.п. Использование h_i опционально. Для вычисления стоимостей маршрутов f_S, f_D используются стандартные функции расстояний между двумя точками.

Алгоритм A* хранит информацию о двух структурах данных. Созданные, но еще не посещенные узлы заносятся в список открытых узлов, а все посещенные узлы – в список закрытых. Реализация таких структур и способ их применения сильно влияет на производительность. На каждом шаге работы алгоритма для соседних с текущим узлов (соответствующих текущему направлению движения) вычисляется F, и полученные значения добавляются в список открытых узлов. Наиболее перспективный для перемещения узел в списке открытых должен иметь минимальное значение F.

Алгоритм останавливается, когда конечная вершина попадает в список полностью исследованных или закрытых вершин. Найденный путь воспроизводится с помощью указателей на предыдущую вершину.

Графическая интерпретация работы алгоритма A*. Стартовая ячейка обозначена «S», конечная – «D», темные ячейки соответствуют препятствиям.

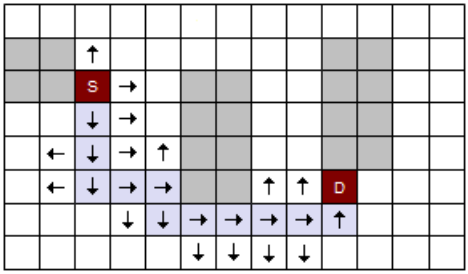


Рис. 2. Графическая интерпретация работы волнового алгоритма

На практике A* является очень гибким. Так как, его легко модифицировать для поиска маршрута к множеству целей. Результаты работы алгоритма A* сильно зависят от качества выбранного эвристического приближения *F*.

4. Реализация и тестирование

В качестве алгоритмов для реализации были выбраны выше рассмотренные Волновой алгоритм и A*.

Была поставлена задача построения маршрута между объектами, которые могут принадлежать нескольким цепям, порт соединения соответствует одному номеру цепи. Маршрут от исходного объекта прокладывается либо до конечного назначения, либо до пересечения с уже имеющимся соединением одноимённой цепи.

Изначально задаётся количество объектов на сетке, число их портов и возможное количество цепей в системе.

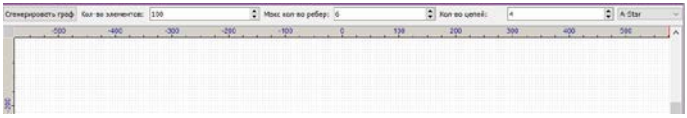


Рис. 3. Начало работы

После того как начальные условия заданы происходит генерация объектов случайным образом. После этого выполняется алгоритм трассировки по цепям. На вход метода подаётся два объекта класса QPoint. В качестве эвристик для алгоритма A* были использованы следующие:

- Стоимость маршрута;
- Штраф за смену направления;
- Штраф за пересечения других проводниковых соединений;
- Штраф за отклонения от центра между точками соединения.

В зависимости от условий оптимальности решаемой задачи можно менять список эвристик. После того как конечная точка или ближайшее проводниковое соединение было достигнуто, выполняется обратный проход для формирование итогового маршрута.

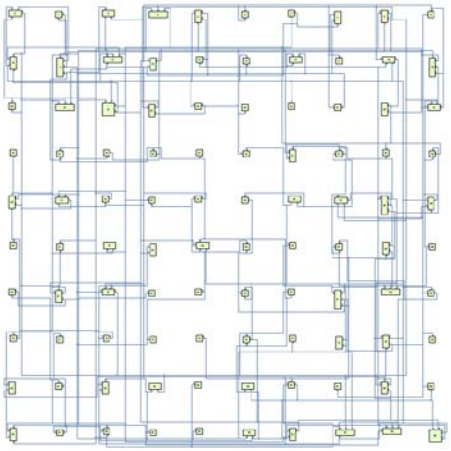


Рис. 4. Пример работы A*

На вход Волнового алгоритма подаются две точки одноимённой цепи. Волна распространяется в 4-х возможных направлениях от источника. При посещении ячейки помечаются последовательностью цифр 1-2-3-1-2-3. По достижению волной целевой точки происходит восстановление маршрута по последовательности меток 3-2-1-3-2-1.

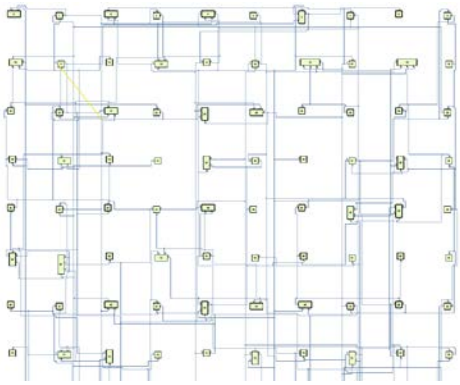


Рис. 5. Пример работы Волнового алгоритма

Приведём таблицу сравнения результатов работы двух алгоритмов:

Таблица

Сравнение результатов работы:

Кол-во элементов	Макс кол-во рёбер	Кол-во цепей в системе	Время работы (A*)	Время работы (Ли)
100	5	10	0,1	2,12
500	5	55	0,29	54,61
1000	10	135	15,21	250,61

Исходя из полученных результатов видно, что алгоритм A* работает гораздо быстрее, плюс ему требуется меньше памяти при построении маршрута. Поэтому при рассмотрении систем с больших числом объектов и их возможных соединений целесообразнее использовать алгоритм A*.

Реализованные варианты алгоритма трассировки, на основе волнового и алгоритма A*, позволяют строить локально оптимальные маршруты для большого числа связанных между собой объектов. Реализация может применяться в приложениях, требующих быстрого построения/перестроения маршрутов, например, в приложениях, предусматривающих визуализацию и интерактивную коррекцию больших чертежей и диаграмм.

Заключение

Данная статья посвящена разработке и реализации программного модуля для трассировки соединений с помощью алгоритмов, основанных на клеточной декомпозиции. Описаны алгоритмы A-void и волновой алгоритм, а также создание самого разрабатываемого программного модуля. В итоге данной разработки получился рабочий программный модуль, реализованный в виде оконного приложения. Использование данного решения позволяет прокладывать соединения в графах с большим количеством элементов, имеющих различную степень связности.

Список литературы

1. LaValle, S. M. Planning algorithms / S. M. LaValle; Camb. N.Y. : Camb. Univ. Press, 2006. – 826 p.

2. Yang, K. 3D smooth path planning for a UAV in cluttered natural environments / K. Yang, S. Sukkarieh // IEEE/RSJ intern. conf. on intelligent robots and systems. – Nice, France, Sept. 22-26, 2008. – P. 794–800.

3. Elfes, A. Using occupancy grids for mobile robot perception and navigation / A. Elfes // Computer. – 1989. – P. 46–57.

4. 3-D path planning in a dynamic environment using an octree and an artificial potential field / Y. Kitamura [and all] // IEEE-RSJ intern. conf. on intelligent robots and systems. – Pittsburgh, USA, Aug. 05-09, 1995. – P. 474–481.

5. A real-time obstacle detection and reactive path planning system for autonomous small-scale helicopters. / J. Redding [and all] // AIAA Guidance, navigation and control conf. and exhibit. – Hilton Head, USA, Aug. 20–23, 2007. – P. 989–1010.

6. Chazelle, B. Triangulating a nonconvex polytope / B. Chazelle, L. Palios // Discrete and Computational Geometry. – 1990. – P. 505–526.

7. Russell, S. J. Artificial intelligence: A modern approach. / S. J. Russell, P. Norvig // 3rd ed. Upper Saddle River: Prentice Hall. – 2010. – P. 1132.

8. Ferguson, D. Using interpolation to improve path planning: The field D* algorithm / D. Ferguson, A. Stentz // Journal of Fields Robotics. – 2006. – P. 79–101.

9. Theta*: Any-angle path planning on grids / Daniel K. [and all] // Journal of Artificial Intelligence Research. – 2010. – P. 533–579.

10. Koenig, S. D* lite / S. Koenig, M. Likhachev // 18th national conf. on artificial intelligence. – Edmonton, Alberta, Canada, July 28–August 1, 2002. – P. 476–483.

11. Волновой алгоритм поиска пути [Электронный ресурс]. – Режим доступа: http://www.100byte.ru/100btwrks/wv/wv.html

12. Introduction to A* Pathfinding [Электронный ресурс]. – Режим доступа: https://www.raywenderlich.com/3016-introduction-to-a-pathfinding

Реализация Wi-Fi mesh сети на основе системы на кристалле MCom-02 (ЭЛВИС)

С. А. Трухачев,

Студент магистратуры

Д. Н. Борисов

Доцент

Введение

Научно производственный центр электронные вычислительно-информационные системы (НПЦ «Элвис») является одним из немногих на Российском рынке, осуществляющий разработку и изготовление микропроцессорной техники, ряду изделий которого присвоен статус отечественных микросхем первого и второго уровня, что в условиях импортозамещения позволяет использовать их в оборонной промышленности РФ.

В работе рассматривается реализации Wi-Fi mesh сети система-на-кристалле (или сокращенно СнК) на базе собственной платформы проектирования НПЦ «Элвис» – «Мультикор» [1, 2]. Процессоры серии «Мультикор» – это однокристальные программируемые многопроцессорные СнК, созданные на базе библиотеки IP – ядер платформы Мультикор.

В связи с этим процессоры данной серии сочетают в себе функционал двух классов приборов: микроконтроллеров и цифровых процессоров обработки сигналов что позволяет применять устройства с данным процессором в широком спектре задач таких как: контроль данных, передача информации, а также ее обработка [3].

В данной работе рассматривается принципы работы DSP кластера семейства «Мультикор» – процессором 1892ВМ14Я [2] («Мультиком-02», MCom-02), а также особенности развертывания частной MESH-сети с использованием отладочной платы Салют-ЭЛ24Д1 на базе системы на кристалле MCom-02.

1. Система на кристалле и ее особенности

Микропроцессор «Мультиком-02» (MCom-02, 1892ВМ14Я) является 6-ядерным сигнальным микропроцессором с пониженным энергопотреблением, созданным для связных, мультимедийных,

навигационных, а также встраиваемых мобильных приложений. Данный функционал осуществляется за счет двухъядерного центрального процессора Cortex-A9 с частотой до 816 МГц, цифрового сигнального двухъядерного процессора ELcore-30M (с частотой до 672 МГц) и графического процессора Mali-300 (250 млн. пикселей/с).

Кластер DSP представляет собой двухпроцессорную MIMD (multiple instruction, multiple data) систему. Каждое DSP ядро имеет в наличии собственную программную память и может работать независимо от остальных ядер. На верхнем уровне DSP-кластера имеется набор общих для всего кластера регистров управления и состояния [4].

Для синхронизации работы DSP ядер в кластере имеется в наличии два механизма: механизм прерываний и механизм обменов через XBUF в синхронном режиме. Каждое DSP ядро может формировать прерывание для любого другого ядра в кластере, используя соответствующие регистры. Ядро, получившее прерывание, переходит в состояние «выполнения» (RUN), причем, если оно было остановлено, то исполнение подпрограммы начинается с адреса, который хранится в специальном регистре этого ядра [4].

Помимо SoC, на плате модуля находится 2 ГБ ОЗУ DDR3, 4 ГБ флэш-памяти NAND, модуль eMMC объемом 32 ГБ. Список интерфейсов включает USB 2.0, Ethernet 10/100/1000 Мбит/с, два порта SpaceWire, видеовходы TTL (12 бит), MIPI CSI2 (2 порта, 4 линии), видеовыходы TTL RGB (24 бит), MIPI DSI (4 линии), а также микрофонный и линейный аудиовходы, линейный выход и выходы для подключения наушников. Кроме того, есть интерфейсы SDMMC, I2S, I2C (3 порта), UART (4 порта), SPI (2 порта), четыре канала ШИМ, MFBSP (LPORT, SPI, I2S, GPIO), GPIO (до 116 портов). Модуль размерами 60 х 60 х 5,5 мм (представлен на рис. 1) рассчитан на напряжение питания 3,3 В. Типовая потребляемая мощность не превышает 5 Вт [5].

В табл. 1 приведена сводная информация аппаратных характеристик популярных решений по отношению к 1892ВМ14Я.

2. Wi-Fi mesh сети

Mesh-сети могут интегрировать в себе различные сетевые и радиотехнологии. Самый распространённый на сегодняшний день стандарт беспроводного соединения устройств – Wi-Fi. Поэтому и сами mesh-сети строятся в основном на этой технологии. А используются такие сети преимущественно для организации локальных (LAN) и городских (MAN) сетей.

Таблица 1

Популярные решения одноплатных микрокомпьютеров

	MCom-02	**Raspberry PI3**	**Orange PI PLUS 2E**	**Banana Pi M3**	**Cubieboard5**
Частота ЦПУ	672 МГц – 816 МГц	1 ГГц	1.2 ГГц	1.6 ГГц	2.0 ГГц
Количество ядер	6	4	4	8	8
Память	2 GB DDR3	1 GB DDR2	2 GB DDR3	2 GB DDR3	2 GB DDR3
Ethernet	10/100/1000 M RJ-45	10/100M RJ-45	10/100/1000M RJ-45	10/100/1000M RJ-45	10/100/1000M RJ-45
Wi-Fi	802.11 b/g/n	802.11 n	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n
Питание	5V@2A via MiroUSB	5V@2A via MiroUSB	5V@2A via DC	5V@2A via MiroUSB	5V@2A via DC
Вес	46 Грамм	48 Грамм	45 Грамм	48 Грамм	425 Грамм

Особая актуальность mesh-сетей определяется развитием микроэлектроники, появлением множества различных устройств, способных работать автономно долгое время, имеющих особенность многократной смены режима (online нахождения в сети и offline-выхода из сети) и нуждающихся в обмене информацией со своим окружением, а возможно и с управляющим или информационным центром.

Одно из преимуществ mesh-сетей – независимость. Можно создать свою мобильную сеть передачи данных, которую никто не контролирует, и всё время оставаться на связи. Чем больше абонентов – тем плотнее и надёжнее сеть. Таким образом, можно всегда оставаться на связи в местах, где отсутствует сетевая инфраструктура. Это может оказаться весьма полезно в районах повышенного риска (где вынуждены работать специальные бригады), в местах дикой или неосвоенной природы (где проводят исследования учёные, археологи, геологи, а так же при использовании в военной промышленности) и удалённых населённых пунктах, где абсолютно каждое абонентское устройство (например, смартфон местного жителя или станция, установленная в транспортном средстве участкового) может принимать участие в процессе передачи важной информации до адресата.

Сетевой протокол 802.11s, который используется при постройке MESH-сетей имеет преимущества перед протоколом 802.11 такие, как автоматическая маршрутизация, обеспечивающий такие возможности, как топология сети "каждый с каждым"; устойчивость при отказе

отдельных компонентов; масштабируемость сети – увеличение зоны информационного покрытия в режиме самоорганизации; динамическая маршрутизация трафика, контроль состояния сети.

В полноценных mesh-сетях нельзя перехватить трафик и запретить распространение информации. Это, в свою очередь, может противоречить государственным законам конкретной страны или региона. В то же время, государственные структуры и военные ведомства по этим же самым причинам заинтересованы в освоении и организации подобных сетей.

В настоящее время существует 5 самых популярных протоколов для организации Wi-Fi Mesh сетей у каждого есть свои достоинства и недостатки с которыми можно ознакомится в таблице (табл. 2) [6].

Таблица 2

Популярные протоколы для организации Wi-Fi Mesh сетей

	CJDNS	B.A.T.M.A.N.	DTN	Netsukuku	OSPF
Авто-назначение адреса	Да	Нет	Нет	Да	Нет
Авто-конф. Маршрутизация	Да	Да	Да	Да	Частично
Распределенная маршрутизация	Да	Да	Да	Да	Частично
Объеденение сетей	Да	Нет	Нет	Нет	Нет
IPv4/v6	IPv6	IPv4/v6	IPv4/v6	IPv4	IPv4
Шифрование трафикка внутри сети	Да	Нет	Нет	Нет	Нет
Авто-настройка	Да	Да	Да	Нет	Да
Разработка	Активная	Закончена	Активная	Нет	Закончена
Поддержка UNIX\LINUX\OpenWRT	Да	Да	Да	Да	Да
Поддержка Windows	В разработке	Нет	Нет	Нет	Нет
Поддержка Mac OS X	Да	Да	Да	Да	Да
Потребление ресурсов	Низкое	Низкое	Низкое	Высокое	Низкое
Оверлейны режим работы	Да	Нет	Нет	Нет	Нет
Интеграция в ядро Linux	Нет	Да	Нет	Нет	Да

В сравнительной табл. 2 больше всего преимуществ имеет сетевой протокол CJDNS, но нашем случае предпочтение было отдано сетевому протоколу B.A.T.M.A.N., так как разработка данного протокола уже закончена, это значит, что даже если и будут выходить какие-то обновления, которые могут повлиять на работоспособность между устройствами с разными версиями, то это будет происходить гораздо реже у протокола B.A.T.M.A.N. чем у CJDNS. Так же протокол CJDNS может работать только с IPv6. Сетевой протокол B.A.T.M.A.N. присутствует в стандартных депозитариях linux систем в том числе и ALT Linux, что упрощает его установку на отладочный модуль.

3. Реализация внешнего Wi-Fi адаптера на основе драйвера RTL8188EUS

Встроенный Wi-Fi модуль платы Салют-ЭЛ24Д1 не имеет возможность одновременно работать как Wi-Fi клиент и Wi-Fi точка доступа. Поэтому было принято решение реализовать внешний сетевой Wi-Fi адаптер на основе драйвера RTL8188EUS. Внешний сетевой адаптер будем использовать как Wi-Fi точка доступа, а встроенный модуль как Wi-Fi Ad-hoc, что позволит существенно увеличить область покрытия Wi-Fi сети.

Realtek RTL8188EUS – это высокоинтегрированная однокристальная беспроводная локальная сеть 802.11n (WLAN). Интерфейс устройства USB (USB 1.0/1.1/2.0 совместимый) контроллер. RTL8188EUS представляет собой комплексное решение для высокопроизводительного интегрированного устройства беспроводной локальной сети [7].

Базовая полоса WLAN RTL8188EUS реализует мультиплексирование с ортогональным частотным разделением (OFDM) с 1 каналом передачи и 1 каналом приема и совместим со спецификацией IEEE 802. 1 1n. Полоса пропускания устройства 20 МГц и 40 МГц.

Контроллер WLAN RTL8188EUS поддерживает быструю автоматическую регулировку усиления приемника (AGC) с синхронными и асинхронными контурами управления между антеннами, функциями разнесения антенн. На конртолере реализована функция управления мощностью передачи для получения лучшей производительности в аналоговых частях приемопередатчика [7].

Печатная плата была спроектирована в утиллите Sprint-Layout 6 (рис. 1) и реализована путем травления в хлором железе.

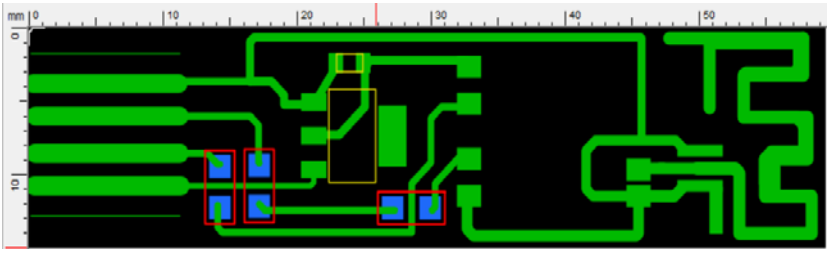


Рис. 1. Схема реализации WiFi адаптера

Для правильной работы на адаптере реализован стабилизатор напряжения ams1117-33 на 3.3V для питания модуля, 3 перемычки 12060

Ом, для надежности так же был добавлен 1 конденсатор 0.1 Мкф. В дальнейшем для индикации и понижения входного напряжения в схему был добавлен диод. Так же на плате сразу реализована антенна (рис. 2).

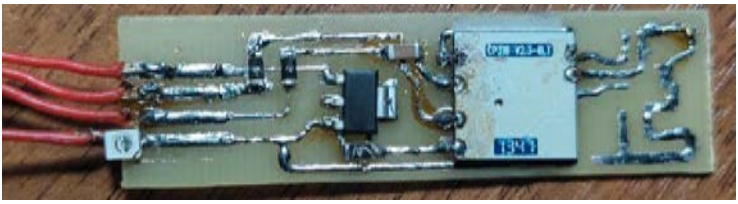


Рис. 2. Реализация WiFi адаптера на чипе RTL8188CUS

Полученый адаптер имеет возможность подключения через USB, а так же через GIPO пины платы Салют-ЭЛ24Д1 [8]. Для windows подойдет драйвер TP-LINK RTL8188TL-WN725N_V2_121123.zip, в нашем же случае устройство необходимо прошить с помощью библиотеке на python hostapd-rtl8188 для реализации точки доступа на этом устройстве.

4. Развертывания частной MESH-сети с использованием MCom-02

Для настройки сетевых интерфейсов необходимо воспользоваться режимом AD HOC. В режиме Ad Hoc клиенты устанавливают связь непосредственно друг с другом. Устанавливается одноранговое взаимодействие по типу «точка-точка», и компьютеры взаимодействуют напрямую без применения точек доступа. При этом создается только одна зона обслуживания, не имеющая интерфейса для подключения к проводной локальной сети.

Для полноценной работы Wi-Fi MESH сетей необходимо минимум 2 устройства. На обоих должна быть произведена первоначальная настройка, но на основном устройстве, в которое, например, может быть подключён Ethernet кабель для доступа в интернет, должен быть дополнительно установлен и настроен DHCP сервер для того, чтоб пользователи могли без дополнительных настроек, автоматический получать ip-адрес и полноценно пользоваться построенной MESH сетью.

Для установки DHCP сервера на главное устройство мы выберем для реализации точки доступа пакет hostapd (host access point daemon) – сервис (демон), реализующий программно точку доступа на беспроводном интерфейсе и предоставляющий средства аутентификации клиентов этой точки доступа. Он поддерживает

привязку ip-адреса к компьютеру или автоматическую настройку ip-адресов из заданного диапазона.

Настройка сетевого протокола B.A.T.M.A.N. производится при помощи утилиты batctl. Данная утилита присутствует в стандартном депозитарии и можно установить с помощью – sudo apt-get install batctl. После проверки установки необходимо загрузить модуль ядра выполнив команду modprobe batman-adv.

С помощью утилиты nmtui создаем точку доступа выбрав режим Ad-Hoc и настроить сетевой интерфейс. Канал можно выбрать любой главное, чтобы у всех участников MESH сети он был одинаковый. MTU рекомендуется выставить именно – 1528 для избегания фрагментации пакетов.

Последним действием необходимо указать настроенный беспроводной адаптер в утилите batctl следующим образом: batctl if add wlan0. Для автозапуска соединения необходимо сделать так, чтобы при установке соединения через Network Manager автоматический выполнялась команда добавления интерфейса в утилиту batctl после любого сетевого события (рис. 3).

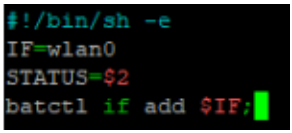


Рис. 3. Автоматическое добавление сетевого интерфейса при любом событии в Mesh-сеть

Заключение

Mesh-сеть, построенная по принципу ячеек, в которой рабочие станции соединяются с друг другом и способны принимать на себя роль коммутаторов для остальных участников является достаточно сложной в настройке, однако при такой топологии, у построенной сети реализуется высокая отказоустойчивость, так как при обрыве одного из соединений, остальные продолжат функционировать, перестроив маршрут минуя потерянный узел.

Проведя ряд практических экспериментов, удалось доказать, что отладочная плата Салют-ЭЛ24Д1, дополненная драйвером RTL8188EUS, на базе системы на кристалле MCom-0 при использование сетевого протокола B.A.T.M.A.N, отвечает всем требованиям Mesh-сетей, от сюда можно сделать вывод что подобную технологию можно использовать в данном отладочном модуле в зависимости от спецификации задач устройства.

Список литературы

1. Микросхема интегральная 1892ВМ14Я [Электронный ресурс]: Руководство пользователя – Режим доступа: https://multicore.ru/mc/data_sheets/Manual_1892VM14YA.pdf

2. Система на кристалле 1892ВМ14Я [Электронный ресурс]: Технические характеристики. – Режим доступа: http://multicore.ru/index.php?id=1335

3. Baer, J. Microprocessor Architecture: From Simple Pipelines to Chip Multiprocessors / J. Baer. – Cambridge : Cambridge University Press, 2009. – 382 p.

4. Солодухин, В. М. Обработка аудиосигнала на DSP-кластере СнК САЛЮТ-ЭЛ24Д / В. М. Солодухин, Д. Н. Борисов // Труды молодых ученых факультета компьютерных наук ВГУ; под ред. Д. Н. Борисова; Воронежский государственный университет. – Воронеж : ООО «ВЭЛБОРН», 2021. – Вып. 1. – С. 411-420.

5. Модуль отладочный Салют-ЭЛ24Д1 [Электронный ресурс]. – Режим доступа : https://multicore.ru/index.php?id=1353

6. Само организующие Mesh-сети для частного использования [Электронный ресурс]: Руководство пользователя – Режим доступа: http://msm.omsu.ru/jrns/jrn40/GussMesh2016.pdf

7. RTL8188EUS. General Description [Электронный ресурс] : Режим доступа : https://www.realtek.com/en/products/communications-network-ics/item/rtl8188eus

8. 1892ВМ14Я: Linux. – Режим доступа : ftp://ftp.elvees.com/1892VM14YA/linux

Реализация алгоритмов в Oracle Siebel CRM с использованием Siebel Workflow

Д. М. Хрунина

Студент магистратуры

Д. И. Соломатин

Старший преподаватель

Введение

Любым компаниям, работающим с большими объемами разрозненных данных, а в особенности компаниям, работающим с людьми (напр., банки), требуется инструмент, который позволит собрать все данные в одном месте, систематизировать их и предоставлять сотрудникам доступ к этим данным с учетом их процессных ролей, т.е. регулировать уровни доступа. Примером такого инструмента является программный комплекс Oracle Siebel CRM, позволяющий не только систематизировать данные, но и автоматизировать процессы обмена данными. Однако, хотя данный комплекс поставляется с готовыми решениями «из коробки», не требующими дополнительной настройки, (напр., Siebel Call Center, Siebel Financial Services и другие), зачастую компаниям-потребителям требуется более тонкая настройка системы под нужды предприятия и конкретной отрасли. Под настройкой в данном случае подразумевается создание объектов, которыми оперирует система, отражающих информационные процессы на предприятии, а также автоматизация этих процессов.

На данный момент поддержка таких систем поставщиком прекращена, однако, мгновенно импортозаместить такие системы невозможно, в том числе потому, что это потребует огромного объема работ по переносу данных и настроек в новую систему – полной настройки нового комплекса с нуля. Поэтому существующие продукты продолжают эксплуатировать и расширять. Более того, теперь и техподдержка полностью переходит на отечественных интеграторов, в связи с чем ответственность за разработку качественного ПО и потребность в квалифицированных кадрах возрастает.

1. Постановка задачи

Требуется автоматизировать процесс создания записи на основе данных, получаемых из внешнего источника. Внешний источник передает данные в виде иерархии – особого типа данных Siebel. Эту иерархию, а также идентификатор родительской записи, требуется получать в методе создания записи.

2. Поток операций Siebel

Oracle Siebel CRM для разработки методов предоставляет 2 основных инструмента: язык eScript, использующийся для написания методов, как привязанных к конкретным объектам, так и отдельных, формируемых в пакеты, именуемые бизнес-сервисами, и Workflow, или поток операций Siebel. Лучшей практикой считается избегать написания скриптов, в том числе потому, что работать с Workflow проще и интереснее, поэтому для реализации требуемого метода необходимо разработать поток операций.

Siebel Workflow делятся на два типа: контекстные – хранящие контекст записи, с которой они работают, и неконтекстные. Неконтекстные потоки используют, когда необходимо выполнить действия без привязки к активной записи или когда вызов потока происходит не из системы Siebel и получить контекст нет возможности. Такие потоки не предоставляют стандартных возможностей для работы с записями (поиск, создание, обновление и т.д.). Контекстные же Workflow позволяют, например, обработать существующую запись без необходимости искать ее по идентификатору – контекст активной записи передается автоматически. Таким образом, для создания новой записи, дочерней к уже существующей, потребуется создать контекстный поток операций.

Для разработки Workflow используется графическая среда разработки, предоставляемая Siebel Tools – инструментом разработки Oracle Siebel CRM. Интерфейс среды разработки потоков операций представлен на рис. 1.

В левом окне расположены все доступные элементы потока операций, так называемые шаги, а также коннекторы. По центру – окно конструктора Workflow, где, перетаскивая туда нужные шаги, выстраивают поток операций подобно блок-схеме. В нижней части – окно атрибутов, или полей, всего потока или выбранного шага [1].

Каждый поток операций должен начинаться и заканчиваться соответствующей меткой. Шаги должны быть последовательно соединены стрелками-коннекторами. Простейший Workflow будет состоять только из меток начала и конца потока, соединенных стрелкой (рис. 1), такой поток не выполняет никаких функций.

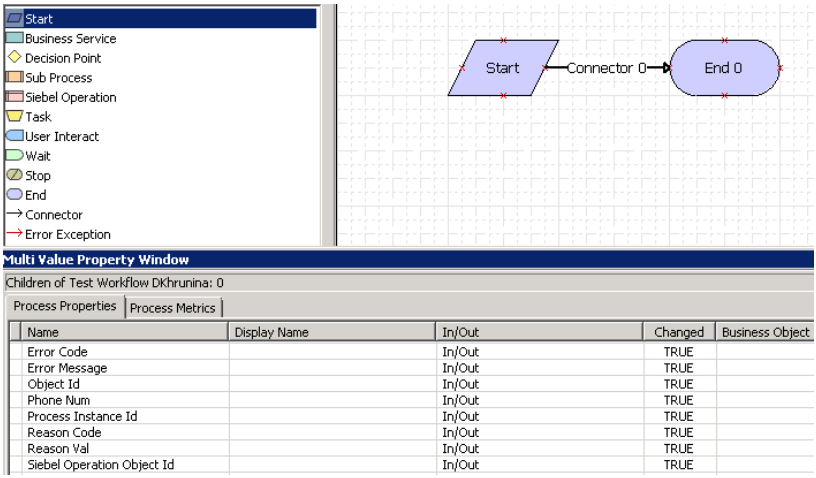


Рис. 1. Интерфейс редактора потока операций

Для построения нужного метода потребуется три типа исполняемых шагов (рис. 2), а также коннекторы и метки начала и конца потока:

1. Business Service – шаг, позволяющий вызвать метод выбранного бизнес-сервиса;
2. Siebel Operation – шаг для работы с записями в системе: поиска, создания, удаления и т.д.
3. Decision Point – шаг ветвления потока по некоторому условию[1].

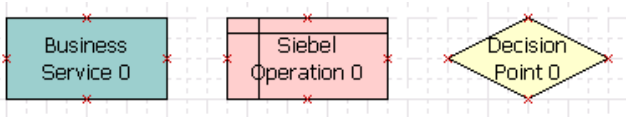


Рис. 2. Исполняемые шаги разрабатываемого потока операций

Для передачи данных между шагами потока необходимо сохранить их в общие атрибуты потока, а затем передать в новый шаг [1]. Также в общие поля записываются входные и выходные параметры всего Workflow. Кроме того, каждый шаг имеет свои атрибуты – входные и выходные параметры, которые принимает каждый метод, условия поиска. Некоторые шаги требуют определения свойств шага – так, например, для шага Business Service необходимо указать имя бизнес-сервиса и имя вызываемого метода.

3. Разработка структуры требуемого метода

Требуется разработать метод, который на основании полученных из внешней системы данных создаст новую запись, содержащую некоторые поля: номер телефона, номер договора, причину создания записи и источник – в данном случае тут указывается внешняя система.

На вход в метод передается иерархия Siebel – дерево данных, а также идентификатор контекстной записи. Контекст будет получен автоматически.

Требуется получить из иерархии номер телефона и ключ для поиска причины, а также получить из словаря значение причины. Далее необходимо проверить, существует ли в системе активная запись переданного номера телефона для выбранного клиента, создать ее, если требуется, а затем создать конечную запись. Идентификатор клиента можно получить из контекстной записи.

4. Получение значений из иерархии и словаря

Для формирования потока операций необходимо создать новую запись Workflow и указать объект, в контексте которого будет работать поток: FINCORP Account [1].

В потоке потребуется получить 3 значения, для них нужно создать атрибуты потока. Для этого в нижнем окне редактора Workflow требуется создать 3 новые записи типа String (строка) и с именами Phone Num, Reason Code и Reason Val. Также необходимо указать для этих атрибутов тип входа/выхода: так как это локальные данные, подходящий тип – None.

Далее необходимо выстроить структуру потока: 4 шага Business Service и Decision Point. Поток будет сначала получать данные из иерархии, затем – из словаря, затем проверять, удалось ли получить значение из словаря, и в случае ошибки устанавливать сообщение об ошибке (рис. 3).

Для каждого шага Business Service требуется указать имя бизнес-сервиса и вызываемого метода в окне свойств (рис. 4) [1]. Для получения значений из иерархии используют метод GetProperty из пакета PRM ANI Utility Service. Для получения значения из словаря – метод LookupValueByInt из пакета TSC Common Functions. Для установки сообщения об ошибке подходит метод Echo пакета Workflow Utilities.

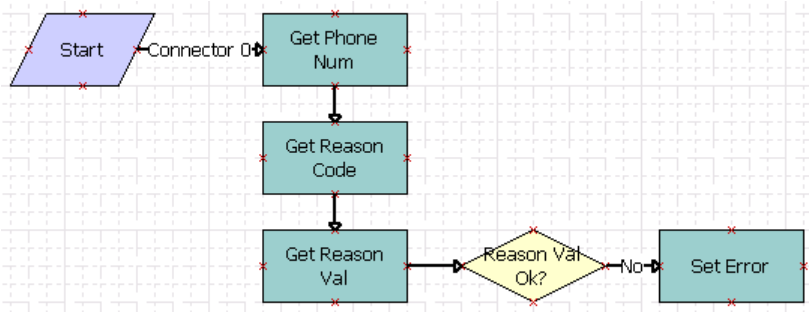


Рис. 3. Получение значений из иерархии и словаря – структура

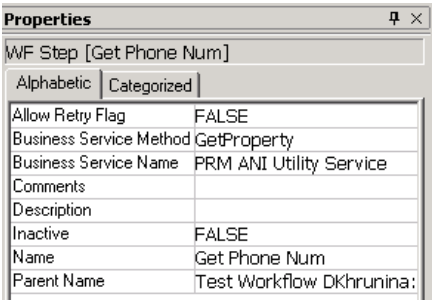


Рис. 4. Окно свойств шага

Шаг Decision Point не требует настройки таких параметров, однако требует создания правил перехода в ту или иную ветвь. Условия перехода указываются в свойствах соответствующего коннектора (рис. 5) [1].

Используемые методы имеют обязательные параметры. Метод получения значений из иерархии требует на вход саму иерархию, путь к родительскому элементу и имя элемента, значение которого необходимо получить. Метод, обращающийся к словарю, требует в качестве входных параметров имя словаря и ключ. Echo не имеет обязательных входных параметров, так как используется для установки значений локальных атрибутов потока операций.

Для передачи нужных параметров в методы нужно создать соответствующие атрибуты каждого шага: указать имена входных параметров, тип передаваемого значения и значение или имя локального поля потока.

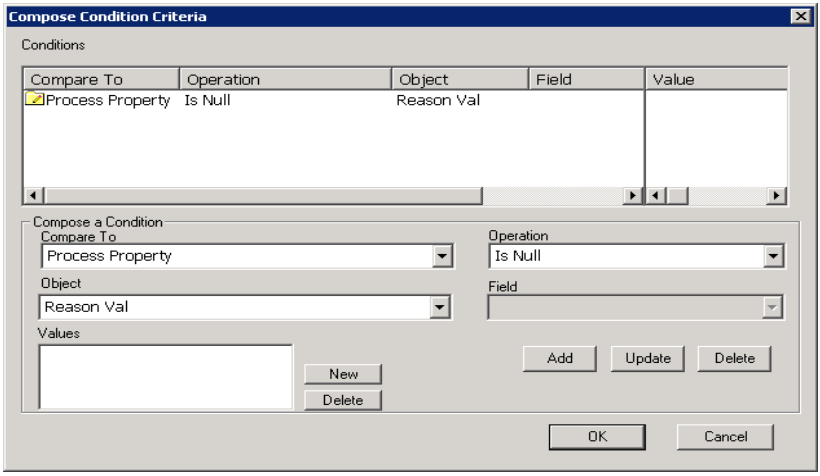


Рис. 5. Окно создания условий перехода по ветке

Для передачи строковых значений используют тип Literal, для полей – Process Property (рис. 6) [1]. Аналогичным образом необходимо создать выходные параметры – записать полученные значения в локальные поля. В шаге установки сообщения об ошибке требуется указать код ошибки и передаваемое сообщение.

Sequence	Input Argument	Type	Value	Property Name
2	Hierarchy Path:	Literal	SendApplicationRestructuringReq	
1	SiebelMessage	Process Property		Siebel Message
3	Property Name	Literal	PhoneNumber	

Рис. 6. Атрибуты шага Get Phone Num

Далее следует создать условие перехода по ветке – переход по ошибочной ветке осуществляется, если не удалось получить значение из словаря, то есть вернулось Null (рис. 5).

5. Создание записи

В этой части требуется проверить, существует ли активная запись переданного номера телефона для текущего клиента, создать ее, если требуется, а затем создать конечную запись. Для работы с записями потребуется шаг Siebel Operation, а также оператор ветвления (рис. 7).

Шаги Siebel Operation требуют указать объект, с которым каждый шаг работает, а также выполняемую операцию – поиск или создание записи [1]. Для поиска требуется задать спецификацию – условие, по которому будут отобраны записи.

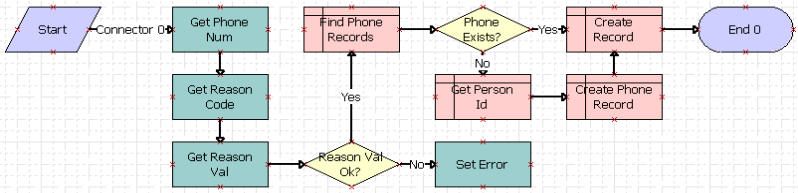


Рис. 7. Итоговый поток операций

Так как используется контекстный поток операций, эта задача упрощается – при поиске записей телефона не требуется указывать идентификатор пользователя, т.к. поиск осуществляется в рамках контекстной записи и дочерних к ней записей. При создании записи следует указать поля, которые необходимо заполнить, и передать соответствующие значения. Для этого в атрибутах шага создают соответствующие записи.

Ветвление в данном случае следует осуществлять по критерию найденной записи – если хотя бы одна запись нашлась, поток переходит по ветке «Yes» к созданию конечной записи. При этом для второго коннектора необходимо в свойствах указать тип «Default» – в эту ветку будет осуществлен переход, если ни одно из условий не выполнено.

При создании конечной записи нужно передавать идентификаторы записей договора и телефона, а также причину создания записи и источник. При этом поле источник можно заполнять, получая нужное значение из словаря. Для этого требуется указать тип передаваемого в атрибут значения Expression – выражение – и вместо строкового значения указать код получения значения из справочника.

Заключение

Статья посвящена разработке и реализации метода в комплексе Oracle Siebel CRM для создания записей в системе на основе данных, полученных из внешней системы. Описаны технологии разработки потоков операций Siebel. Использование данного решения позволит автоматизировать процесс работы Siebel с внешними системами и снизить нагрузку на персонал. В дальнейшем предполагается разработать функциональность, которая позволит принимать и обрабатывать данные в формате XML и подготавливать их для работы в Oracle Siebel CRM, что позволит расширить применяемость технологии.

Список литературы

1. Siebel Business Process Framework: Workflow Guide [Электронный ресурс]: официальная документация. – Режим доступа: https://docs.oracle.com/cd/E14004_01/books/PDF/BPFWorkflow.pdf

Содержание

Научное издание

ТРУДЫ МОЛОДЫХ УЧЁНЫХ
ФАКУЛЬТЕТА КОМПЬЮТЕРНЫХ НАУК ВГУ

В ы п у с к 2

Под редакцией *Д. Н. Борисова*

Издано в авторской редакции

Подписано к использованию 2.06.22
Тираж 500 экз. Заказ 222

ООО «Вэлборн»
394068, г. Воронеж, Московский пр-т, 98
Тел. +7 (930) 4035-418
http://wellborn@scirep.ru, e-mail: wellborn@scirep.ru