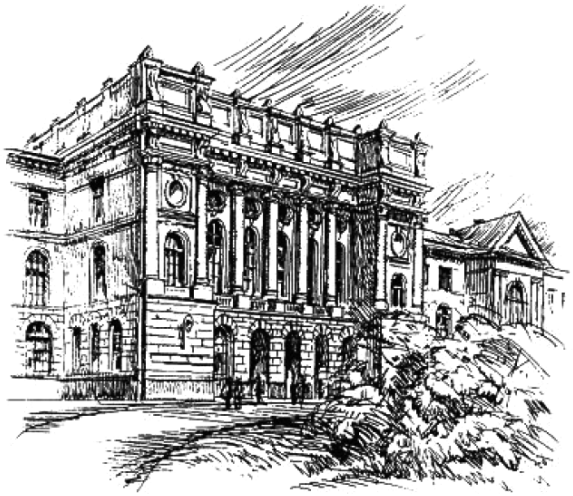
Министерство науки и высшего образования Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО
DELL TECHNOLOGIES
EPAM SYSTEMS

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ В ТЕОРИИ И ПРАКТИКЕ ПРОГРАММИРОВАНИЯ

Сборник материалов конференции

22 апреля 2021 года



ПОЛИТЕХ-ПРЕСС
Санкт-Петербургский
политехнический университет
Петра Великого

Санкт-Петербург
2021

УДК 004
ББК 32.973
С56

Современные технологии в теории и практике программирования : сборник материалов конференции, 22 апреля 2021 г. – СПб. : ПОЛИТЕХ-ПРЕСС, 2021. – 236 с.

В сборнике публикуются материалы докладов студентов, представленные на научно-практическую конференцию, проводимую Санкт-Петербургским политехническим университетом Петра Великого и организованную Институтом компьютерных наук и технологий при поддержке Санкт-Петербургского Центра Разработок Dell Technologies и компании EPAM. Доклады отражают уровень подготовленности студентов – участников конференции, в области применения современных средств и технологий разработки программного обеспечения.

Представляет интерес для специалистов в области информационных технологий, методов разработки программных проектов различного назначения, систем и средств автоматизации инженерного проектирования, а также для учащихся и работников системы высшего образования.

Редакционная коллегия:
Директор ВШПИ ИКНТ *П. Д. Дробинцев*, профессор *И. Г. Черноруцкий*

Печатается по решению
Совета по издательской деятельности Ученого совета
Санкт-Петербургского политехнического университета Петра Великого.

© Dell Technologies, 2021
© EPAM Systems, 2021
© Санкт-Петербургский политехнический университет Петра Великого, 2021

ISBN 978-5-7422-7264-9

Приветствие от Санкт-Петербургского Центра Разработок Dell Technologies

Уважаемые участники конференции
«Современные технологии в теории и практике программирования»!

Непрерывное развитие технологий в последние годы показало, что именно доступ к качественным данным позволит компании выйти на новый уровень, улучшив обслуживание заказчиков. А использование всех преимуществ владения данными поможет выйти вперед в конкурентной гонке. Организации собирают все больше и больше данных, однако данные накапливаются гораздо быстрее, чем можно было бы качественно их проанализировать. Возникает потребность в новых исследованиях и технологиях.

По этой причине Dell Technologies с 2013-го года поддерживает конференцию «Современные технологии в теории и практике программирования», которая позволяет молодыми учеными продемонстрировать результаты их исследований в области информационных технологий.

Сотрудничеству между Санкт-Петербургским центром разработок Dell Technologies и Политехническим университетом уже более двенадцати лет. За эти годы нами было реализовано более пятнадцати совместных образовательных проектов. С 2016-го года в Политехническом университете реализуются студенческие проекты с открытым исходным кодом, в рамках которых участники вносят свой вклад в продукты, которыми пользуются ведущие ИТ-компании всего мира. Сотрудники компании проводят спецкурсы для студентов-политехников по тематикам современного программирования, управления программным продуктом, хранению данных, тестированию программного обеспечения и DevOps подходам. Более того, выпускники Политеха проходят стажировки в компании Dell Technologies, по окончании которых, лучшие студенты получают возможность стать инженерами нашего центра.

Целью как данной конференции в целом, так и в частности секции Dell Technologies является, в первую очередь, поддержка молодых ученых, их научных исследований в области информатики и информационных технологий; информирование молодых ученых о передовых технологических трендах в индустрии, о современных программных продуктах и решениях, представленных крупнейшими мировыми ИТ-корпорациями. Для компаний, занимающихся разработкой и сопровождением ПО, равно как и другой деятельностью в области ИТ, сотрудничество с университетами жизненно важно для дальнейшего устойчивого развития.

Желаю вам успешной работы на конференции, профессионального роста и творческих побед!

С уважением,

Павел Борисович Егоров

Генеральный директор

Санкт-Петербургского Центра Разработок Dell Technologies

Выпускник Санкт-Петербургского политехнического университета

Приветствие от компании EPAM

Уважаемые гости конференции!

Мы рады приветствовать студентов, аспирантов и молодых учёных в качестве участников ежегодной научно-практической конференции «Современные технологии в теории и практике программирования», партнёром которой традиционно выступает компания EPAM.

В прошлом году система образования вынужденно столкнулась с беспрецедентным испытанием — пандемия коронавируса COVID-19 оказалась тем фактором, которого никто не мог ожидать и предвидеть. Тем не менее, массовый переход образования в дистанционный режим и ситуация социальной изоляции стали не только опытом преодоления, но и открытием множества нестандартных подходов. Информационные технологии окончательно проникли практически в каждый аспект деятельности компаний и организаций, в повседневную жизнь каждого из нас. Вопрос скорости разработки и внедрения новых решений стали не только факторами успеха, но и выживания. Гибкость, адаптивность, поиск нетривиальных решений — вот обязательные факторы для тех, кто стремился к открытию новых возможностей, несмотря на трудности. Именно цифровая трансформация предъявляет особые требования к квалификации будущих ИТ-профессионалов.

Компания EPAM активно поддерживает проекты и инициативы, целью которых является практический вклад в совершенствование процессов подготовки молодых специалистов. В рамках конференции мы будем рады поделиться своим опытом реализации инновационных проектов для лидеров российского и мирового бизнеса, рассказать о перспективах развития новых инструментов в области информационных технологий. Надеемся, что это поможет вам утвердиться в правильности выбранного пути и поможет приблизиться к тому этапу, на котором ключевые изменения IT-ландшафта будут зависеть от принятых вами решений.

С уважением,

директор офиса в Санкт-Петербурге

Марианна Округина

Информационное сообщение о проведении научно-практической конференции студентов, аспирантов и молодых ученых «Современные технологии в теории и практике программирования»

В целях содействия подготовке к будущей работе в профессиональных программистских коллективах, обеспечивающих высокое качество программного продукта, в целях поддержки изучения современных информационных технологий и инструментальных средств в соответствии с мировыми стандартами и действующими международными сертификационными требованиями, а также для выявления талантливых молодых специалистов в области разработки и использования программных систем, инженерных проектов и моделей, лидеры в разработке программного продукта компании Dell Technologies и EPAM совместно с Санкт-Петербургским Политехническим Университетом объявляют научно-практическую конференцию **«Современные технологии в теории и практике программирования».**

1. В рамках мероприятия планируется провести:
   - Пленарное заседание и открытые лекции ведущих специалистов компаний Dell Technologies и EPAM
   - Конкурс-конференцию теоретических и практических работ - претендентов на именные дипломы компаний Dell Technologies и EPAM.
2. Сроки проведения мероприятия: **22 апреля 2021 года**.
3. Формат проведения: онлайн, на платформе Microsoft Teams.

Предлагаемая тематика конкурса:

- Программная инженерия: приложения, продукты и системы
- Программная инженерия: инструментальные средства и технологии проектирования и разработки
- Программная инженерия: методы и алгоритмы теории программирования
- Технологии обработки и анализа больших массивов данных на основе работ EPAM
- Подходы к разработке ПО на основе технологий Dell Technologies.

Данный конкурс рассчитан на студентов 1-6 курсов, аспирантов и молодых ученых, обучающихся или работающих по направлениям информатики, вычислительной техники, и по направлениям использования информационных технологий, владеющих основами современных промышленных технологий и инструментарием разработки программных продуктов и проектов. Предполагается, что участник должен проявить свои знания и умения не столько в области программирования различных математических головоломок, сколько в области разработки и использования программных продуктов и систем в условиях, максимально приближенных к реальным процессам проектирования и разработки современных систем различной степени сложности.

Темы работ могут быть взяты в соответствии с тематикой секции конференции.

На конкурс принимаются работы, оформленные в соответствии с заданными требованиями и представленные в организационный комитет конференции не позднее **16 марта 2021 года**. Требования к представлению проектов представлены в документах [I-II].

4. Количество работ конкурсантов не ограничено.
5. Конкурс принятых к участию работ проводится в один этап и протекает в виде представления презентаций и секционных докладов. Приглашения авторам отобранных работ будут рассылаться в период с 9 по 15 апреля 2021г. Решение о награждении участников конкурса принимается конкурсной комиссией. Требования к проведению

докладов представлены в документе «Порядок проведения презентаций и выступлений на конкурсе-конференции» [III].

6. Объявления о условиях проведения конкурса и вся дополнительная информация будут представлены на сайте ИКНТ СПбПУ в разделе Конференции. Приглашения на участие в конкурсе-конференции будут разосланы в ведущие университеты Северо-Запада.

ЖЕЛАЕМ УСПЕХОВ И ИНТЕРЕСНОЙ РАБОТЫ!

Научные руководители конференции:

директор ВШПИ ИКНТ Дробинцев Павел Дмитриевич,

профессор Черноруцкий Игорь Георгиевич

секретарь конференции Эламик Татьяна Николаевна, тел. 552-76-66 часы приема: вторник и четверг с 15 до 17 часов.

Адрес: Политехническая ул., 29, 3 уч.корпус, а.300

E-mail: **icc_2021@mail.ru**

I. Порядок представления проектов на конкурс-конференцию.

Для представления разработанных проектов в Конкурсный центр устанавливаются следующие правила:

- Желающие принять участие в конкурсе – конференции на этапе предварительного сбора материалов на почтовый адрес icc_2021@mail.ru присылают материалы секционного доклада и анкету участника.
- На заочном этапе – первом туре - конкурсная комиссия на основе тезисов отбирает работы для непосредственного участия в конкурсе, после чего высылает приглашение по электронной почте (при наличии) или по телефону.
- Очный этап проводится в **виде секционных докладов для теоретических или прикладных поисковых работ.**
- Конкурсная комиссия оценивает представленные доклады по представлению руководителей секций и отбирают наиболее значимые из них для награждения. Участники очного этапа награждаются специальными дипломами Dell Technologies или EPAM.

II. Требования к содержанию конкурсных работ.

Предполагается, что конференция будет проходить в рамках пяти секций:

С1.Программная инженерия: приложения, продукты и системы

Задача конкурса в разделе С1: продемонстрировать в тезисах и презентации знания и умения в создании программного продукта на основе доклада, содержащего обоснование предлагаемого решения и анализа преимуществ по сравнению с существующими.

(Например, программные проекты, представляющие завершенный продукт или незавершенный, но исполняемый в соответствии с требованиями к промышленному продукту)

Рекомендуется особо отмечать передовые технологии, поддержанные в программных продуктах и оценку эффективности их применения.

С2.Программная инженерия: инструментальные средства и технологии проектирования и разработки

Задача конкурса в разделе С2: продемонстрировать знания и умения в создании и/или применении перспективных методов и технологий разработки программного обеспечения на основе доклада, содержащего обоснование предлагаемого решения и анализа преимуществ по сравнению с существующими. Наряду с технологиями разработки программных продуктов на секции рассматриваются программные средства систем управления жизненным циклом информации.
(Например, автоматизация разработки спецификаций, доказательства корректности спецификаций, их использование при генерации кода, автоматизация тестирования, средства обеспечения качества программного продукта, средства управления разработкой программного продукта, средства идентификации и информационной безопасности, управляющие и встроенные применения, беспроводная телекоммуникация, моделирование контроллеров и других аппаратно-программных решений, средства управления качеством реализации программного продукта, системы управления электронным документооборотом, безопасностью хранения данных, виртуализация вычислительных ресурсов.)

Рекомендуется особо отмечать передовые технологии, поддержанные в программных продуктах и оценку эффективности их применения.

С3. Программная инженерия: методы и алгоритмы теории программирования

Задача конкурса в разделе С3: продемонстрировать знания и умения в разработке и применении формальных методов при создании и/или улучшении либо оптимизации характеристик программного обеспечения на основе доклада, содержащего обоснование предлагаемого решения и анализа преимуществ по сравнению с существующими.

(Например, алгоритмы и методы для проверки корректности программного продукта, исполнимые спецификации, формальные методы для применения в программировании и т.п.)

Оценка применимости предлагаемых подходов на практике и оценка эффективности применения.

С4. Технологии обработки и анализа больших массивов данных на основе работ EPAM

Задача конкурса в разделе С4: продемонстрировать знания и умения в разработке ПО с применением новых технологий EPAM в области обработки и анализа больших массивов данных, на основе доклада, содержащего обоснование предлагаемого решения и анализа преимуществ.

С5. Подходы к разработке программного обеспечения на основе технологий Dell Technologies

Задача конкурса в разделе С5: продемонстрировать знания и умения в разработке ПО с применением новых технологий Dell Technologies. на основе доклада, содержащего обоснование предлагаемого решения и анализа преимуществ.

III. Порядок проведения презентаций и выступлений.

В процессе сбора материалов на конкурс-конференцию конкурсная комиссия отбирает лучшие работы для представления презентаций и выступлений, о чем участник конкурса ставится в известность.

Порядок проведения выступления на конкурсе-конференции:

1. В ходе конференции используется проекционное оборудование.
2. Для доклада при помощи Microsoft PowerPoint готовится презентация.
3. Презентация должна содержать не менее 10 слайдов. Обязательными являются следующие слайды:
   - Титульный, на котором должна быть представлена следующая информация:
     - Название проекта.
     - Фамилия и имя докладчика;
     - Учебное заведение, которое он представляет;
     - Фамилия И.О. научного руководителя.
   - Область применения проекта.
   - Какие задачи решаются проектом.
   - 2 слайда на описание концепции проекта.
   - 2 слайда на особенности реализации проекта.
   - Представление полученных результатов проекта.

- Перечень использованных для реализации продуктов (указать, какие технологии и для чего использовались в проекте).
- Заключение.

4. Длительность выступления не должна превышать 10 минут.

IV. Положение о системе награждения победителей конкурса – конференции.

Победителей конкурса определяет конкурсное жюри, в состав которого включены ведущие преподаватели вузов. Жюри оценивает представленные материалы и отбирает лучшие из них для участия в работе конференции. Работы будут премироваться следующим образом:

1. Тезисы докладов по всем отобранным комиссией проектам будут опубликованы в сборнике тезисов докладов конкурса-конференции.
2. Победители получат именные дипломы.

Тезисы докладов конкурса-конференции

Секция «Программная инженерия: приложения, продукты и системы»

УДК 004.62

Алейников П.И. (1 курс магистратуры), Рамейков А.С. (2 курс магистратуры)
Фёдоров С.А., ст. преподаватель

СИСТЕМА ПРОГНОЗИРОВАНИЯ ВЫРАБОТКИ СОЛНЕЧНОЙ ЭЛЕКТРОСТАНЦИИ

Основой для сегодняшних инноваций во всех сферах является электроэнергия. Любое ее отключение даже на незначительное время доставляет как минимум неудобства отдельно взятым группам людей, так и быть катастрофической из-за вывода из строя важных инфраструктурных объектов. Также стоить отметить, что на сегодняшний момент все большую популярность обретают гибридные энергосистемы, состоящие из станций, работающих на углеводороде, то есть на полезных ископаемых, так и из альтернативных источников, таких как солнечные электростанции. Однако проблемой таких систем является то, что из-за невозможности предугадать выработку электроэнергии альтернативными электростанциями, станции, работающие на полезных ископаемых всегда должны работать параллельно, вырабатывая излишки энергии. В данной работе поставлена задача предсказания выработки электроэнергии солнечной электростанции на некоторое время вперед.

На рисунке 1 представлен подход к реализации системы предсказания выработки

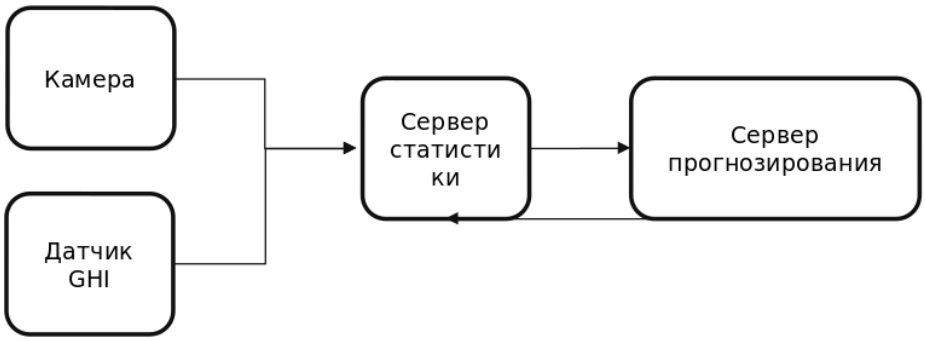


Рисунок 1 – Дизайн системы

Камера: Камера отвечает за отслеживание облаков, она направлена ввысь, а также комплектуется линзами «Рыбий глаз», которые обладают угловым полем начиная от 180 градусов для покрытия всего неба. Это нужно так как полусферические изображения позволяют получить максимальный пространственный охват, ограниченный только кривизной земли.

Датчик GHI: Для того, чтобы набор данных для прогнозирования был полон рядом с камерой должен быть расположен датчик GHI, который отслеживает общее количество

коротковолнового излучения, принимаемого поверхностью, в нашем случае солнечной панелью, расположенной горизонтально к земле.

Сервер статистики: Сервер статистики собирает данные с датчиков и камер и располагает их в паре фотография и значение GHI, при этом он подготавливает и держит в кеше данные за N количество времени для более точного прогноза. Как видно из рисунка именно сервер статистики является связующим звеном между модулем прогнозирования.

Сервер прогнозирования:
Сервер прогнозирования делится на несколько модулей:

- Модуль обработки изображения
- Модуль анализа солнечного излучения
- Модуль предсказания выработки

Модуль обработки изображения включает в себя такие этапы обработки как распознавания типа облаков по методу соотношения между каналом красного и синего цветов исходного изображения с камеры с последующим анализом изображения в библиотеке ClearSky, после этого вычисляется расстояние до облака, с помощью теней которое оно отбрасывает, а также при помощи положения солнца. После этого данные передаются в модуль анализа солнечного излучения. В нем анализ оценивается в зависимости от удаленности датчика от камеры и в зависимости от различных типов облаков. По этим данным строятся временные ряды карт излучения для каждого датчика GHI. После этого модуль предсказания может объединяет данные движения облаков и карт излучения и может вычислить дальнейшее движение облаков посредством экстраполяции текущего изображения на полученные ранее данные, а с помощью карт излучения получить значение GHI в любой интересующей позиции

Вывод: Была разработана архитектура и описаны модули системы для предсказания выработки солнечной электростанции.

ЛИТЕРАТУРА

1. Thomas Schmidt , Martina Calais, Eric Roy, Ashton Burton, Detlev Heinemann, Thilo Kilper, Craig Carter, "Short-term solar forecasting based on sky images to enable higher PV generation in remote electricity networks", Renewable Energy and Environmental Sustainability, 2017.
2. Soumyabrata Dev, Florian M. Savoy, Yee Hui Lee, Stefan Winkler, "Estimation of solar irradiance using ground-based whole sky imagers", IEEE International Geoscience and Remote Sensing Symposium, pp. 7236-7239, 2016
3. Thomas Schmidt, John Kalisch, Elke Lorenz, Detlev Heinemann, "Evaluating the spatio-temporal performance of sky-imager-based solar irradiance analysis and forecasts", Atmospheric chemistry and physics, т.16, №. 5, pp 3399-3412, 2016
4. Samuel R.West, Daniel Rowe, Saad Sayeef, Adam Berry, "Short-term irradiance forecasting using skycams: Motivation and development", Solar Energy, т.110, pp. 188-207, 2014

УДК 004.855.2, 81.322.3

Алексеева М.В. (4 курс бакалавриата)
Тимофеев Д.А., вед. программист
Самочадин А.В., к.т.н., доцент

ПОСТРОЕНИЕ СТАТИСТИЧЕСКОГО ПАРСЕРА CCG ДЛЯ РУССКОГО ЯЗЫКА

При решении многих задач обработки естественных языков (например, для машинного перевода, построения чат-ботов или систем извлечения информации) часто нужно производить синтаксический анализ текста на основе грамматики языка. Среди способов описания грамматик большой интерес представляют комбинаторные категориальные грамматики (CCG) — лингвистически обоснованный формализм, который позволяет описывать широкий класс языковых явлений и в то же время допускает эффективный

синтаксический анализ, а также предоставляет достаточно простой способ систематически строить семантические представления предложений на основе их деревьев вывода.

Для построения грамматик, охватывающих широкий класс реальных текстов, чаще всего используется статистический вывод. Автоматическому выводу грамматик на основе CCG посвящено большое количество работ, однако в них в основном рассматриваются только грамматики английского языка. Для морфологически богатых языков, в том числе для русского языка, автоматический вывод грамматик CCG и применение построенных грамматик для решения прикладных задач обработки теста остается недостаточно исследованным.

Данная работа посвящена разработке и реализации алгоритма автоматического вывода грамматики CCG для русского языка на основе неразмеченного корпуса текстов. В основе работы лежат алгоритмы, разработанные Й. Биском и Дж. Хокенмайер для английского языка [1], которые были адаптированы для русского языка, в первую очередь с учетом его морфологии.

В CCG к каждому слову и словосочетанию привязаны одна или несколько синтаксических категорий. Категории могут быть атомарными или составными. Составные категории строятся с помощью операторов «\» и «/», аргументами которых также являются синтаксические категории. Если конструкция имеет составную категорию A/B, это означает, что непосредственно справа от нее должна находиться конструкция с категорией B, а их объединение будет иметь категорию A. Категория A\B интерпретируется аналогично, но конструкция с категорий B должна находиться непосредственно слева. Группировка конструкций и преобразование категорий производится с помощью фиксированного набора правил. Каждому слову можно приписать базовый набор категорий, опираясь на то, какой части речи соответствует слово. Такой подход позволяет разбирать многие предложения, но для анализа более сложных и длинных приходится использовать категории, более точно учитывающие возможные связи между словами. Необходимо также задать распределение вероятностей соответствия категорий словам: статистический парсер CCG использует это распределение, чтобы выбрать из всех возможных для предложения деревьев вывода наиболее вероятное.

Биск и Хокенмайер предлагают следующий подход к формированию словаря.

1. Выбирается базовый набор атомарных синтаксических категорий и набор ограничений на вид составных категорий, которые могут быть приписаны словам.

2. Для набора предложений, используемого в качестве обучающей выборки, с помощью существующих статистических инструментов производится разметка частей речи.

3. На основе обучающей выборки формируется словарь, в который заносятся те удовлетворяющие выбранным ограничениям категории, для которых предложения оказываются выводимыми.

4. На основе сформированного словаря с помощью алгоритма ожидания-максимизации строится статистическая модель, описывающая распределение вероятностей синтаксических категорий для каждого слова в словаре.

Непосредственное использование доступной реализации этого подхода для русского языка невозможно: в русском языке каждому слову соответствует гораздо большее количество форм, согласование которых необходимо учитывать при формировании дерева вывода, при этом порядок слов является свободным. Для адаптации алгоритма к особенностям русского языка необходимо сформулировать правила вывода новых категорий и предусмотреть способы хранения различных форм слов и их связей.

Исходными данными для построения модели является неразмеченный корпус текста, для каждого предложения в котором сначала производится морфологический анализ. Каждому слову приписывается набор стандартных категорий. Далее производится синтаксический разбор предложения с учетом согласования морфологических признаков слов. В словаре должны остаться те категории, с которыми предложения удалось разобрать, с добавлением

соответствующих ограничений на морфологические признаки. После этого строится вероятностная модель.

Для реализации данного программного комплекса выбран язык Python, для которого существует большое количество библиотек для обработки текстов на естественных языках, а также средств интеграции с библиотеками на других языках программирования и сетевыми сервисами.

Для предварительной обработки и частеречной разметки текстов на русском языке используется библиотека pymorphy2 - библиотека с открытым исходным кодом, несколько менее точная по сравнению с аналогичной библиотекой pymystem3 (Яндекс), но удовлетворяющая требованиям и имеющая более свободную лицензию. Для синтаксического анализа на этапе формирования словаря используется парсер, входящий в библиотеку NLTK, который допускает использование задаваемых пользователем набора категорий и словаря. В качестве базы данных в проекте используется реляционная СУБД sqlite. В ней хранятся слова, их формы (в виде допустимых падежей, рода числа и т.п.) и соответствующие каждому слову категории с оценками вероятностей. Схема базы данных приведена на рисунок 1.

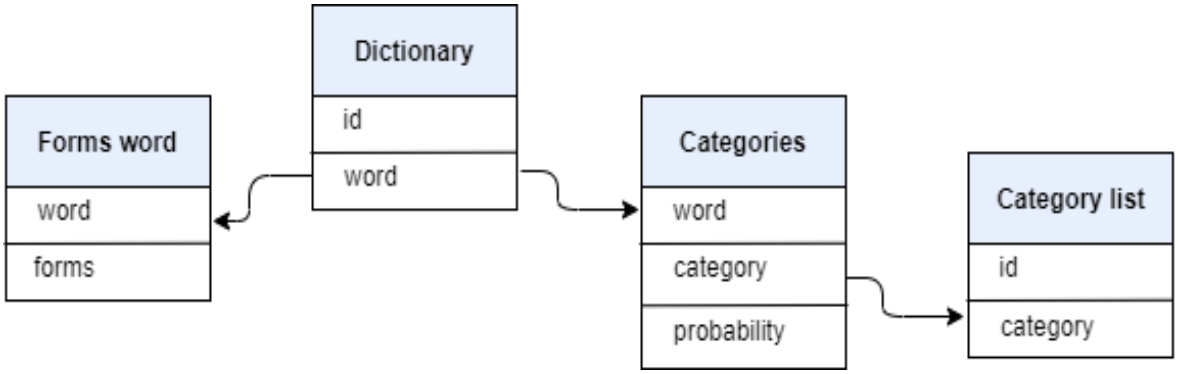


Рисунок 1 – Схема базы данных

ЛИТЕРАТУРА

1. Yonatan Bisk и Julia Hockenmaier. «Simple Robust Grammar Induction with Combinatory Categorial Grammars». В: Proceedings of the TwentySixth AAAI Conference on Artificial Intelligence. 2012, с. 1643—1649.

УДК 004.42

Булатов Д.Е. (1 курс бакалавриата)
Леоньтева Т.В., к.т.н., доцент

ЭКОСИСТЕМА УМНОГО ДОМА НА БАЗЕ ESP8266

С каждым годом все чаще люди задумываются об удобстве эксплуатации жилья и применении технологии «умный дом» [1]. На производствах системы автоматизации применяются уже давно, а до жилища человека они добрались сравнительно недавно, поэтому их выбор не так велик, и устройства имеют недостатки.

Целью данной работы является разработка элементов системы умный дом и приложения на смартфон с учетом устранения недостатков существующих решений и добавления новых полезных функций. На рынке представлено множество компаний, производящих компоненты умного дома или полностью готовые решения. Среди них можно выделить таких производителей как Xiaomi[2] и Redmond[3].

Таблица 1 – Сравнительная таблица

	Мой проект	Xiaomi	Redmond
Приложение на смартфон	+	+	+
Китайский стандарт «умной» розетки	-	+	-
Корректная русификация интерфейса	+	-	+
Встроенные «умные» розетки	+	-	-
Возможность подключения датчиков температуры, влажности и т.д.	+	+	-
Сценарии	+	+	+
Локальный сервер	+	-	-
Работа с голосовым помощником «Алиса»	+	+	+
Работа без подключения к интернету	+	-	-
Функция «кто дома»	+	-	-

Из сравнительной таблицы видно, что в своем проекте я взял лучшее от известных производителей, избавился от их недостатков и добавил новые функции.

За основу проекта был взят микроконтроллер esp8266[4] со встроенным модулем WiFi. Этот микроконтроллер имеет множество плюсов по сравнению с аналогами. Во-первых, данный модуль имеет низкую стоимость, что позволяет снизить конечную себестоимость продукта. Во-вторых, esp8266 очень легко «подружить» с другими модулями и микроконтроллерами. В-третьих, он имеет сравнительно большой объем памяти, которого хватает для размещения сервера.

Разработана программа на C++, обрабатывающая запросы и организующая работу с периферийными устройствами. Работа системы организована на локальном сервере, размещенном на esp8266. На этот сервер приходят запросы и команды, которые пересылаются по сети WiFi на другие элементы системы. После выполнения операций сервер посылает ответ клиенту. Система работает на http запросах. Этот протокол позволяет не поддерживать постоянное соединение со всеми устройствами, что позволяет заметно снизить нагрузку на сеть и не замедлять ее работу.

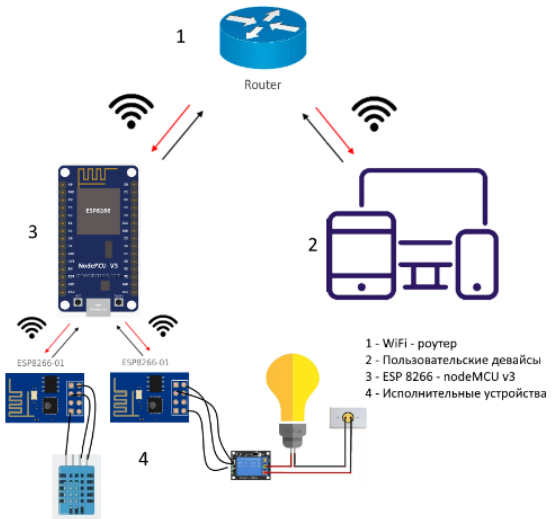


Рисунок 1 – Схема подключения

Приложение на мобильное устройство разработано на языке Java с помощью инструмента Android studio. Для работы программы нужно android устройство с операционной

системой android 5.0 и выше, с возможностью подключения к сети WiFi. Возможно управление системой с помощью web-интерфейса, разработанного на языке JavaScript.

Среди всех возможностей, которые дает готовая система умного дома стоит выделить информационную безопасность. Так как вся экосистема построена вокруг локального сервера, данные не могут быть перехвачены третьими лицами, а отключение от интернета не приведет систему в не рабочее состояние. Безопасность обеспечивается внешним роутером с протоколом wpa2/wpa3, таким образом к системе имеют доступ только те устройства, которые подключены к этому роутеру.

ЛИТЕРАТУРА

1. Орлов Л.О., Тышкевич А.И. Управляющая система для умного дома // Современные технологии в теории и практике программирования: сборник материалов конференции, СПб, 2019.
2. https://www.mi.com/ru/mi-smart-led-bulb-warm-white
3. https://redmond.company/ru/products/redmond-smart-home
4. https://www.espressif.com/en/products/modules/esp8266

УДК 004.93

Власенко Н. А., Дусаева А. И. (3 курс бакалавриата)
Эйзенах Д. С. (2 курс магистратуры)
Маслаков А. П., ст. преподаватель

РЕАЛИЗАЦИЯ СИСТЕМЫ ИЗМЕРЕНИЯ РАЗМЕРОВ ОБЪЕКТОВ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ КОМПЬЮТЕРНОГО ЗРЕНИЯ.

В наше время машинное зрение [1] — это одна из самых перспективных направлений в сфере информационных технологий. Оно позволяет обрабатывать большие объемы данных с точностью недоступной человеку, позволяет увидеть то, что человеческому глазу трудно заметить. Особенно это актуально в промышленности, например, при обнаружении брака. Процесс определения разметов объекта является трудоемким и требует навыков работы со специализированными измерительными прибора. Все эти факторы негативно влияют на погрешность измерений, в то время как компьютерные системы и программные комплексы позволяют добиться большей точности.

В работе проведено исследование методов обнаружения объектов и вычисления их размеров на фото [2]. В ходе эксперимента среди них был найден и применен наиболее эффективный.

Одним из способов обнаружения объектов и вычисления их размеров служит метод определения габаритов объекта относительно другого с известными размерами (эталона), например, монеты или листа формата А4. Метод делится на два этапа: нахождение и выделение предмета вычислений, определение его размеров.

В первом шаге были протестированы два алгоритма: каскады Хаара [3] и нейронная сеть yoloV4 [4]. Первый вариант показал себя не очень эффективно, так как точно выделения объекта сильно зависела от различных факторов: от освещения, однородности фона, формы и цвета исследуемого объекта. Второй вариант - нейронная сеть позволяет наиболее точно, из всех рассмотренных в данной статье способом определять объекты, но не выделять их точные контуры.

На втором шаге, относительно заранее известных параметров образца определяются приблизительные размеры предметов. Погрешность измерений сильно зависит от ракурса съемки. Решить эту проблему можно используя карты глубин [5]. Они позволят определить каким образом объекты расположены на изображении относительно друг друга по оси аппликат и выделить их точную границу. В этом случае необходимо использовать

дополнительное оборудование, т.к. потребуется стереопара изображений. Она позволяет эмитировать стереоскопическое зрение человека, т.е. способность одновременно чётко видеть изображение предмета обоими глазами.

В качестве дополнительного оборудования используется Raspberry Pi 4 и подключенные ультразвуковой датчик и камера. Для измерения объекта необходимо точно определить его границы (найти “крайние точки”) и вычислить длину и ширину объекта в пикселях, а также найти расстояние до него. По приведенным ниже формулам, с использованием физических характеристик камеры, можно рассчитать реальные размеры объекта:

$$\text{Высота объекта на сенсоре (мм)} = \frac{\text{Высота сенсора (мм)} * \text{Высота объекта (пиксель)}}{\text{Высота сенсора (пиксель)}}$$

$$\text{Размер объекта} = \frac{\text{Расстояние до объекта} * \text{Высота объекта на сенсоре (мм)}}{\text{Фокусное расстояние (мм)}}$$

Обычно в системах стереозрения для получения стереопары используют изображения с двух камер, расположенных на определенном расстоянии друг от друга. Мы решили разработать оптическую установку, позволяющую нам достичь похожего эффекта.

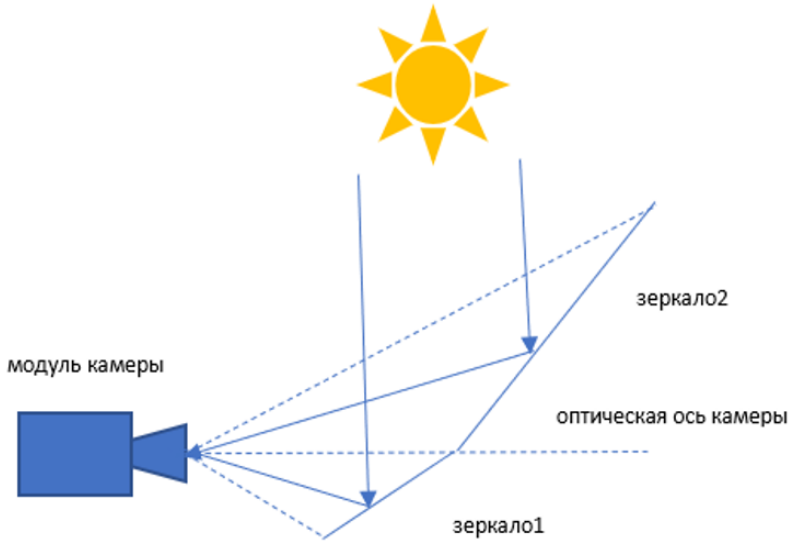


Рисунок 1 – Схема установки

В ней используются зеркала, находящиеся под большим углом, которые и позволяют получить нам два изображения одного и того же объекта в один момент времени. Далее высчитывается расстояние между одной картинкой и смещением на другой. Таким образом, карта глубин представляет из себя матрицу, в которой каждый элемент хранит значения диспаратности точек изображения в диапазоне от 0 до 255. Калибровка происходит с помощью шаблона изображения шахматной доски, с заранее определенными размерами. Эта фотография должна удовлетворять ряду условий: доска должна целиком попадать в кадр, она не должна находиться параллельно граням на полученном фото и т.д. Это важно для конечных результатов калибровки, от ее качества зависят карты глубин. Для дальнейших вычислений необходимо измерить расстояние до объекта при помощи ультразвукового датчика и применить вышеуказанные формулы.

Несмотря на то, что объект выделялся, граница была неточной даже после добавления дополнительных настроек для карт глубин. В процессе работы у нас получилось уменьшить погрешность измерений с первоначальной 30% до 16%. Для улучшения полученного результата, в будущем можно использовать нейронные сети и обработку видеопотока.

Полученные результаты мы объединили в следующую таблицу:

Таблица 1 – Результаты

Методы	**Наша оценка**	**Сложность реализации**	**Определение размеров (точность, %)**
YoloV4	Хорошо классифицирует объекты, точность выделения контура очень низкая	Необходимо установить ПО	23.5
Каскады Хаара	Плохо определяет и выделяет объекты	Требуют дополнительной тренировки на конкретных датасетах	>30
Выделение крайних точек при помощи OpenCV	Хорошая точность, но необходимо изображение предмета на однородном фоне при хорошем освещении	Необходим дальнометр	16
Карты глубин	Хорошая точность, но необходимо хорошее освещение	Необходим дальнометр и оптическая установка	16.6

ЛИТЕРАТУРА

1. С. А. Молодяков, Компьютерное зрение: лабораторный практикум, Санкт-Петербург: Политех-Пресс, 2019.
2. С. Э. Сараджишвили, В. В. Леонтьев и Н. В. Воинов, Введение в обработку изображений на языке Python, Санкт-Петербург: Политех-Пресс, 2020.
3. P. Viola и M. Jones, «Rapid Object Detection using a Boosted Cascade of Simple Features,» 2013.
4. A. Bochkovskiy, . W. Chien-Yao и M. Hong-Yuan, «YOLOv4: Optimal Speed and Accuracy of Object Detection».
5. A. Dubey, «Facing the challenges and seeing the opportunities for ADAS applications».

УДК 004.04

Власов Д.В., Иванов Г.В., Околович Л.Д. (4 курс бакалавриата)
Никифоров И.В., к.т.н., доцент

РЕАЛИЗАЦИЯ ОТЗЫВЧИВОГО WEB-ИНТЕРФЕЙСА ДЛЯ СИСТЕМЫ ТРАНСКОДИРОВАНИЯ ВИДЕОФАЙЛОВ И ВИДЕОПОТОКОВ

Одним из наиболее распространённых типов информации в сети Интернет являются видео и аудио данные [1]. В связи с этим такие компании как: Youtube, Twitch, Amazon, Okko направили своё внимание на работу с медиа данными [2]. Одним из важных аспектов при работе с видео информацией – это транскодирование видео, а иными словами преобразование одного видео в другое с отличными характеристиками (например, другой кодек и битрейт) [3]. Компаниям такое преобразование нужно для того, чтобы вне зависимости от качества соединения с Интернетом, видео информация была доступна конечному пользователю. Каждая организация реализовывает решает данную проблему по-своему. Например, у

компании Amazon есть транскодер собственной разработки, который они дают в аренду [4, 5]. А у Twitch закрытая система транскодирования, недоступная никому, кроме самой фирмы [6].

На рынке есть пять основных предложений по аренде транскодеров на данный период времени: Amazon Elemental, Quencode, Bitmovin, Zencoder, Setaria Cloud. Все они ограничены теми или иными моментами в эксплуатации (например, нет нужного кодека для видео или аудио, не поддерживается склеивание видео из двух в одно), а также требуют серьёзных денежных затрат. Именно поэтому, компания Okko, занимающая одну из лидирующих позиций в области онлайн кинотеатров в России, решила создать собственный транскодер, чтобы в дальнейшем не полагаться на сторонние разработки.

Цель работы состоит в создании системы, отличительной особенностью которой будет скорость транскодирования видеофайлов и видеопотоков, с минимальными потерями качества, а также удобным и отзывчивым пользовательским интерфейсом. Также, вся система будет обёрнуто в контейнер Docker.

В частности, рассмотрим задачу пользовательского интерфейса, поскольку она помогает снизить трудоёмкость работы. Хороший веб-интерфейс должен обеспечить быстрое и простое создание новой задачи на транскодирование, убирая необходимость взаимодействовать с REST интерфейсом пользователю.

Благодаря пользовательскому интерфейсу можно в реальном времени получать следующие наглядные данные:

- процент загруженности жёсткого диска, процессора и оперативной памяти;
- процент выполнения запущенной задачи на транскодирование;
- просмотр итогового видео.

Был облегчён процесс загрузки видео на сервер с транскодером благодаря функционалу по загрузке видеофайлов с локальной машины. Если файлы хранятся на сервисе AWS S3, то загрузить\выгрузить видеофайлы оттуда тоже можно.

Контейнеризация проекта в Docker имеет следующие преимущества [4]:

- безопасность доступа к ресурсам системы;
- масштабируемость;
- простота настройки;
- автоматизация развёртывания;
- уменьшение трудоёмкости.

Благодаря веб интерфейсу, облегчена задача создания нового запроса на транскодирование, так как не надо разбираться в документации к REST Api. Сделано это путём добавления простого диалогового окна, в котором нужно будет просто указать\выбрать нужные данные. В том числе, будет опция выбора профиля транскодирования. Профиль заранее в себе содержит заданные значения параметров для транскодирования. Можно выбрать один из предложенных профилей, или создать новый, после чего выбрать его для задачи. Таким образом, повышается скорость и снижается трудоёмкость создания нового запроса.

В следствии создания пользовательского интерфейса была ускорена работа с транскодером (сравнения приведены с голым REST Api, к которому надо обращаться каждый раз отдельно, например, через “curl”):

- весь список задач сразу перед глазами (в 4 раз быстрее);
- состояние аппаратной составляющей сразу перед глазами (в 4 раз быстрее);
- создание новой задачи на транскодирование значительно упрощается, потому что не надо писать длинные строки с параметрами. К тому же, благодаря профилям, в которых заранее есть все настройки, процесс ускорен ещё больше (в 17 раз, если без профиля, в 25 раз с выбором профиля);
- загрузка\выгрузка видео с AWS S3 быстрее (примерно в 5 раза);
- стало проще и быстрее ставить группы видео на паузу, останавливать, возобновлять и удалять из списка (примерно, в 6 раз быстрее).

Для разработки веб интерфейса был выбран следующий стек технологий: Javascript, React, css файлы, а также различные библиотеки.

Javascript был выбран в связи с тем, что он достаточно прост в использовании, многофункционален, перспективен и популярен (легко найти помощь извне и есть большое количество библиотек).

React – это библиотека Javascript разработанная для того, чтобы не писать код на чистом html. Он крайне гибкий, не вызывает проблем с совместимостью и позволяет декларативно описывать пользовательские интерфейсы и модель состояния [8].

CSS файлы нужны для того, чтобы менять внешний вид элементов на странице.

Работа выполнена в рамках совместного проекта СПБПУ и компании Okko.

ЛИТЕРАТУРА

1. The 50 most visited Websites in the World [Электронный ресурс]. Режим доступа: // https://www.visualcapitalist.com/the-50-most-visited-websites-in-the-world/
2. Сычев В.К., Ленькова Ю.В., Цветкова Е.А., Никифоров И.В., Анализ данных трендов youtube, Современные технологии в теории и практике программирования. сборник материалов конференции. Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. Санкт-Петербург, 2020. С. 131-132.
3. Huifang Sun, Xuemin Chen, and Tihao Chiang; Digital Video Transcoding for Transmission and Storage. Нью Йорк, 2005
4. Лубнин А.В., Ничипорчук А.В., анализ актуальности разработки программного обеспечения для предоставления услуг хостинга Docker контейнеров, Научный электронный журнал «Меридиан». Якутск, 2020. с. 63-65
5. Петров А.А., Селин И.А., Разработка программы по объединению облачных хранилищ данных, Современные технологии в теории и практике программирования. сборник материалов конференции. Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. Санкт-Петербург, 2020. С. 164-165.
6. Amazon Elemental [Электронный ресурс]. Режим доступа: // https://aws.amazon.com/ru/medialive/
7. Live Video Transmuxing/Transcoding: FFmpeg vs TwitchTranscoder, Part I [Электронный ресурс]. Режим доступа: // https://blog.twitch.tv/en/2017/10/10/live-video-transmuxing-transcoding-f-fmpeg-vs-twitch-transcoder-part-i-489c1c125f28/
8. Тузов Г.Д., Воинов Н.В., Реализация асинхронного выполнения событий в высоконагруженном распределенном веб-приложении, Современные технологии в теории и практике программирования. сборник материалов конференции. Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. Санкт-Петербург, 2020. С. 135-136.

УДК 004.051

Губенко Н.О. (4 курс бакалавриата)
Коликова Т. В., ст. преподаватель

ФЕДЕРАТИВНОЕ ОБУЧЕНИЕ И ЧАСТНОЕ МАСШТАБИРОВАНИЕ

Целью работы является изучение нового способа хранения и передачи данных с помощью федеративного обучения и частного масштабирования, выявление достоинств и недостатков этого подхода, а также выводы о целесообразности его применения в проектах.

Федеративное обучение (FL) - это распределенный процесс обучения, в котором модель (веса и контрольные точки) передается на устройства, которые обладают данными, это не классический способ передачи и агрегирования данных централизованно. Таким образом, конфиденциальные данные не покидают пользовательские устройства, что повышает защиту системы и защиту данных каждого пользователя в целом. Федеративное обучение использует

алгоритм федеративного усреднения (FedAvg), который обучается методом усреднения итеративной модели для неидентифицированных и несбалансированных распределенных данных, независимо от количества данных.

Некоторые проблемы чистого федеративного обучения:

1) отсутствие масштабируемости, поскольку модель итеративно обучается на всех устройствах, что усиливается с падением числа устройств;

2) безопасность и конфиденциальность - процесс обучения все еще недостаточно надежен.

3) общая эффективность связи и стоимость выше.

Чтобы смягчить эти проблемы, представляется архитектура федеративного обучения и частного масштабирования (FLaPS), которая улучшает масштабируемость, а также безопасность и конфиденциальность системы. Устройства сгруппированы в кластеры, что дополнительно дает возможность развернуть время до завершения цикла обучения. Таким образом, даже если устройство упадет во время тренировки, весь процесс можно будет запустить снова через определенное время. И данные, и модель передаются с использованием по существу частных отчетов с итеративным переключением, что обеспечивает лучшую торговлю утилитой конфиденциальности. Оценивается FLaPS на наборах данных MNIST, CIFAR10 и TINY-IMAGENET-200 с использованием различных моделей CNN. Экспериментальные результаты доказывают, что FLaPS является улучшенной средой с масштабированием по времени и конфиденциальности, имеющей лучшие и сопоставимые параметры обучения по сравнению с центральной и просто федеративно обучающимися моделями.

Несмотря на все плюсы описанного метода, он все еще имеет ряд проблем, которые пока на данный момент решить не удалось:

1) данные массово распределяются по большому количеству обучающих узлов.

2) данные на узлах могут быть извлечены из другого распределения, которое может не совпадать с распределением на центральном сервере. это явление определяется как не-IID. это сильно влияет на качество обучения переданной модели.

3) узлы могут содержать разное количество примеров для обучения моделей, которые могут варьироваться от очень маленьких до очень больших по размеру, что очень несбалансировано.

Однако, несмотря на недостатки, данный подход к FLaPS дает лучшую гарантию конфиденциальности и полезности за счет внедрения дифференциальной конфиденциальности в сочетании с обычным федеративным обучением в четыре простых шага. Масштабируемость добавляется за счет внедрения кластерного подхода в начале перед обучением. Следом за меньшим количеством устройств, выступающих в роли головок кластеров; повышается надежность за счет более быстрого повторного запуска и за счет меньшего времени на оборачиваемость. Таким образом, в заключение FLaPS строит мост между центральным сервером и федеративно обученной системой устройства с большей надежностью, конфиденциальностью и масштабируемостью.

ЛИТЕРАТУРА

1. Общие правила защиты данных - General Data Protection Regulation – [Электронный ресурс]- https://ru.qaz.wiki/wiki/General_Data_Protection_Regulation
2. Регламент (ЕС) 2016/679 Европейского парламента и Совета от 27 апреля 2016 года – [Электронный ресурс] – eur-lex.europa.eu.
3. H. B. McMahan, E. Moore, D. Ramage, S. Hampson and B. A. y Arcas, ”Communication-E□cient Learning of Deep, Networks from Decentralized Data”, arXiv preprint arXiv:1602.05629 , 2017.
4. Методы минимизации данных и сохранения конфиденциальности в системах искусственного интеллекта Федеративное обучение -[Электронный ресурс]- https://ico.org.uk/about-the-ico/news-and-events/ai-blog-data-minimisation-and-privacy-preserving-techniques-in-ai-systems/

5. ТраскЭндрю. Грокаем глубокое обучение, 2019год.—352стр.
6. K. Bonawitz , V. Ivanov , B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel , … & K. Seth,” Practical secure aggregation for FL on user-held data.” arXiv preprint arXiv:1611.04482.(2016).

УДК 004.932

Власов М.О. (4 курс бакалавриата)
Молодяков С.А., д.т.н., профессор

ЭЛЕКТРОННАЯ СТАБИЛИЗАЦИЯ ВИДЕО НА СМАРТФОНАХ

Современные смартфоны обладают достаточной производительностью, чтобы стабилизировать видео с высокой частотой кадров электронно, не имея оптической или иной механической стабилизации. Однако, не все производители добавляют поддержку электронной стабилизации в свои смартфоны.

Классические алгоритмы стабилизации основаны на отслеживании отдельных признаков ("окон" пикселей) и их перемещении между кадрами – оптического потока. Такие алгоритмы не требуют наличия дополнительных датчиков и могут быть применены к видео в процессе постобработки, поскольку основываются на так называемых "производных" изображения. Широко известен алгоритм вычисления оптического потока Лукаса-Канаде (рис. 1), в котором решается система линейных алгебраических уравнений с неизвестными в виде перемещения одинаковых пикселей по оси абсцисс и ординат [1]. Вводимые ограничения упрощают вычисление системы уравнений, например, вычисление только первой производной изображения. Однако, недостатком таких ограничений является тот факт, что метод эффективно работает только для небольших перемещений пикселей. То есть, сильная тряска или резкая смена экспозиции не может быть компенсирована этим алгоритмом.

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix} \qquad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix} \qquad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}$$

Рисунок 1 – Алгоритм Лукаса-Канаде. A - матрица производных текущего изображения, v - искомая матрица сдвига, b - матрица производных по времени (разница пикселей между кадрами)

Частичным решением проблемы могут быть датчики смартфона – акселерометр и гироскоп. Аккумулируемые данные за период кадра с двух датчиков позволяют получить вектор смещения смартфона в пространстве. Далее рамка изображения смещается в обратную сторону полученного вектора. К сожалению, получаемые данные обладают заметной погрешностью, и даже фильтры сигнала не помогают её компенсировать. Однако теперь сработает классический алгоритм, поскольку итоговая погрешность смещения находится на уровне одного окна поиска признака.

Согласно коду, предоставляемому как основу драйвера камеры компанией Qualcomm, для электронной стабилизации используется гироскоп. Данные с него отправляются в специальный процессор обработки сигнала изображений (ISP - Image Signal Processor), входящего в состав системы на чипе, который вычисляет оптический поток. Применяя данные гироскопа, сдвиг смягчается, чтобы не создавать резких скачков между кадрами. То есть, происходит действие аналогичное предложенному алгоритму в прошлом абзаце, за исключением того, что устранять резкое перемещение специальному процессору не обязательно, поскольку он заточен на операции с изображением и подобные поиски по матрице выполняет моментально даже с большим окном поиска признака. На уровне Android

API у разработчика отсутствует прямой доступ к этому процессору и приходится довольствоваться тем, что есть – центральный процессор и видеочип.

Рассмотрим на упрощённом примере двух последовательно полученных кадров с камеры смартфона (рис. 2). Матрица конкретного экземпляра имеет размер 4000x3000 пикселей, из них вычитается рамка 3840x2160 (4К). Таким образом, на смещение для стабилизации есть 160 пикселей по ширине и 820 по высоте. При получении кадра записываем вектор смещения, полученный с акселерометра и гироскопа. В процессе обработки сначала компенсируем сдвиг с помощью вектора смещения, а затем применяем алгоритм Лукаса-Канаде, который для ускорения может быть выполнен с использованием OpenCV (функция calcOpticalFlowPyrLK). В итоге получаем стабилизированную картинку. Важно обновлять искомые признаки раз в несколько кадров, чтобы адаптироваться под изменения сцены.

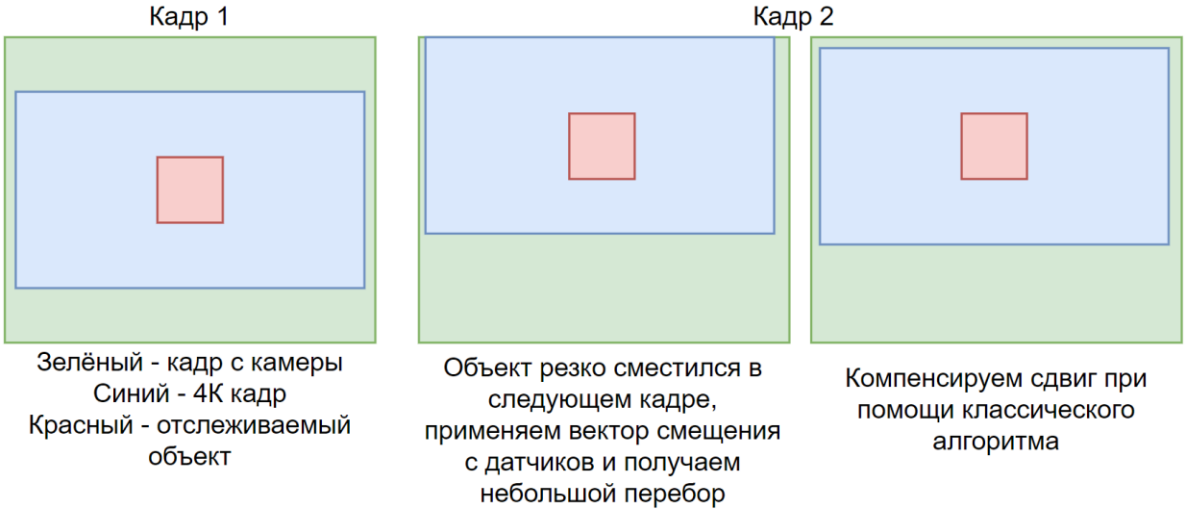


Рисунок 2 – Комбинированный поиск смещения

В докладе представлена программная реализация описанных алгоритмов. Показаны результаты экспериментов, как на компьютере, так и на смартфоне. При работе на компьютере применялись функции компьютерного зрения библиотеки OpenCV [2].

ЛИТЕРАТУРА

1. Калинин М.А., Карпычев В.Ю. Методы вычисления оптического потока для задач анализа трехмерной сущности сцены //Труды НГТУ им. Р.Е. Алексеева, 2016, № 2(113). С.24-29
2. Молодяков С.А. Низкоуровневая обработка изображений в компьютерном зрении. Санкт-Петербургский политехнический университет Петра Великого Санкт-Петербург, 2021. 197 с.

УДК 004.932

Думаев Р.И. (4 курс бакалавриата)
Молодяков С.А., д.т.н., профессор

АВТОМАТИЧЕСКОЕ ОБНАРУЖЕНИЕ COVID-19 ПО РЕНТГЕНОВСКИМ СНИМКАМ

Целью работы является создание системы детектирования и распознавания COVID-19 и других заболеваний легких по рентгеновским снимкам. В условиях ограниченных ресурсов и пандемии возникла нехватка квалифицированных радиологов. Следовательно, существует острая потребность в компьютерных системах диагностики, которые могут помочь рентгенологам в обнаружении различных типов заболеваний по рентгеновским снимкам грудной клетки сразу после их получения. Также, часто такие системы являются закрытыми и не доступными для частных пользователей. Обнаружение COVID-19 и других заболеваний

легких относится к задаче классификации в машинном обучении. Среди методов глубокого обучения сверточные нейронные сети (CNN) показали большие перспективы в классификации изображений и поэтому широко используются исследовательским сообществом. Методы машинного обучения с глубоким обучением на рентгеновских снимках становятся популярными, поскольку имеется множество данных для обучения [1].

Общая архитектура системы. Система состоит из клиентского веб-приложения, с помощью которого можно загружать рентгеновские снимки легких и просматривать отчет о классификации и серверной подсистемы, которая предобрабатывает полученные изображения легких и содержит набор нейронных сетей для сегментации и последующей классификации изображений.

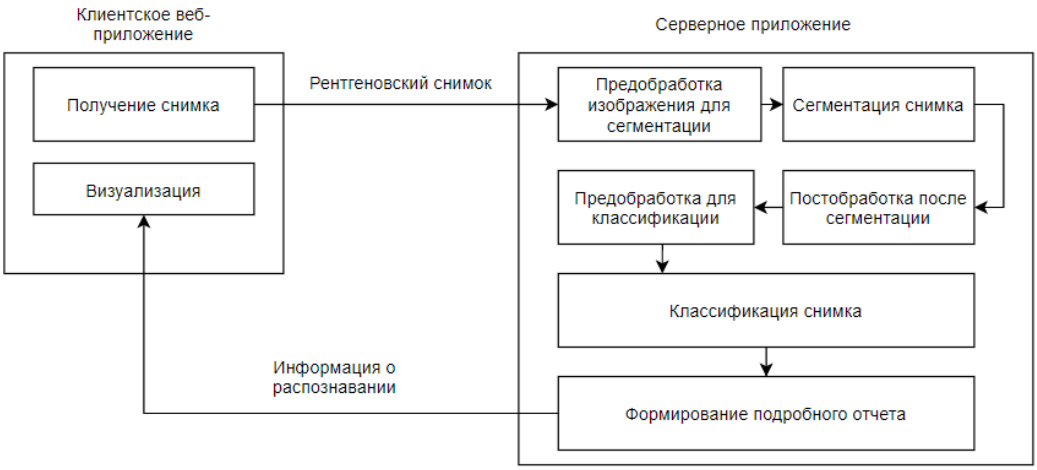


Рисунок 1 – Архитектура системы.

Задача сегментации. Обнаружение заболеваний легких по рентгеновским снимкам можно разделить на две задачи: задачу сегментации и задачу классификации [2]. Для решения задачи сегментации используется сверточная нейронная сеть, предварительная обработка изображений и постобработка изображения, полученного в результате работы модели. Набор данных состоит из снимков легких и бинарных масок легких. Предварительная обработка изображений занимает важную роль в сегментации легких в связи с тем, что некоторые снимки для различных заболеваний имеют помутнения. Без предобработки изображений модели, обученные на наборах данных (многие из которых содержат большую часть снимков легких без заболеваний) без предобработки, могут оказаться неспособны сегментировать аномальные снимки с помутнениями. Для решения этой проблемы был разработан алгоритм предобработки, состоящий из следующих шагов:

– изменения размера до 512x512 пикселей;
– частичное выравнивание гистограммы изображения;
– нормализация значений пикселей;
– случайное добавление шума в виде размытых овалов с различными параметрами.

Для сегментации была выбрана сверточная нейронная сеть U-Net, созданная для сегментации биомедицинских изображений. Архитектура состоит из стягивающего пути для захвата контекста и симметричного расширяющегося пути, который позволяет осуществить точную локализацию. Для этой архитектуры характерно достижение высоких результатов в различных реальных задачах и использования небольшого количества данных для достижения хороших результатов.

Для предотвращения на выходных масках у модели некоторых артефактов (небольшие отверстия на легких, ложные срабатывания) была реализована постобработка, основанная на морфологических операциях [3].

Для оценки точности модели был введен коэффициент Дайса, который показывает меру сходства (отношение площади пересечения к площади объединения). В ходе обучения на наборе данных, состоящих из 5756 изображений, модель после 10 эпох достигла значения

Dice, равного 0.9524. Что говорит о хорошей точности сегментации легких по рентгеновскому снимку.

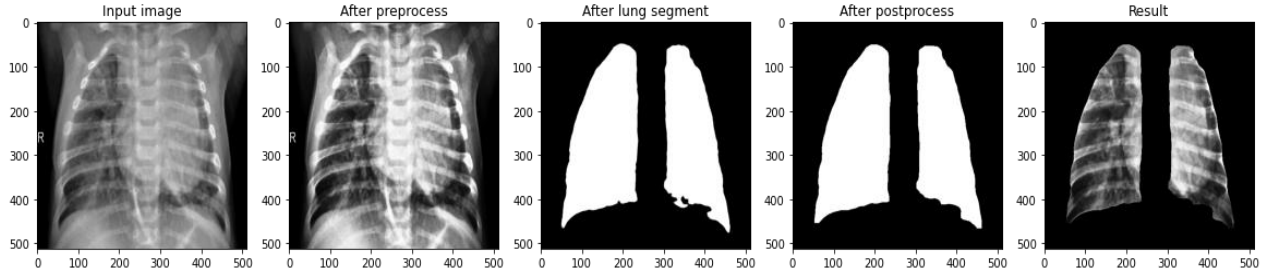


Рисунок 2 – Результат работы всех стадий сегментации.

Задача классификации. Следующим шагом в распознавании заболеваний является классификация изображения, полученного после обработки моделью для сегментации. Для классификации используется классическая архитектура сверточной нейронной сети, которая состоит из свертки, функции максимума и полносвязных слоев. Такая архитектура называется LeNet. Предварительная обработка изображений для обучения состоит из следующих шагов:

– изменения размера до 512x512 пикселей;
– случайное увеличение изображения;
– случайный сдвиг яркости;
– случайный поворот изображения;
– частичное выравнивание гистограммы изображения;
– нормализация значений пикселей.

В ходе обучения на наборе данных, состоящих из 1670 снимков с COVID, 1341 без заболеваний и 3875 с пневмонией, модель после 12 эпох достигла значения точности, равного 0.8743. Ниже представлена матрица ошибок для обученного классификатора.

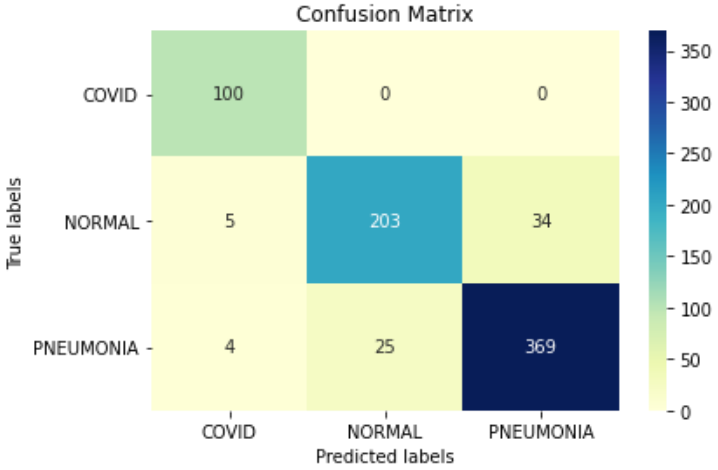


Рисунок 3 – Матрица ошибок классификации.

ЛИТЕРАТУРА

1. Hsin-Jui Chen, Shanq-Jang Ruan, Sha-Wo Huang, Yan-Tsung Peng Lung X-ray Segmentation using Deep Convolutional Neural Networks on Contrast-Enhanced Binarized Images, Mathematics 2020, 8(4), 545; DOI: 10.3390/math8040545
2. Молодяков С.А. Низкоуровневая обработка изображений в компьютерном зрении. Санкт-Петербургский политехнический университет Петра Великого Санкт-Петербург, 2021. 197 с.
3. Raghavendra Selvan, Erik B. Dam, Nicki S. Detlefsen, Sofus Rischel, Kaining Sheng, Mads Nielsen, Akshay Pai Lung Segmentation from Chest X-rays using Variational Data Imputation. arXiv preprint arXiv:2005.10052, 2020

УДК 004.415.002

Загороднев А.А., Приступ М.В., Русcков М.О. (4 курс бакалавриата)
Маслаков А.П., ст. преподаватель

РАЗРАБОТКА ОНЛАЙН ПЛАТФОРМЫ ДЛЯ МЕНЕДЖМЕНТА ГОНОК

Ежедневно на платформе «GTA Online» [0] проводится много различных мероприятий, которые собирают большое количество участников. Однако, сейчас не существует удобных инструментов для организации и мониторинга таких мероприятий. Одним из самых успешных видов мероприятия являются гонки. Одно из самых популярных сообществ в СНГ – Grand Racing League, как раз занимается организацией таких мероприятий, но все действия приходится выполнять вручную, используя таблицы Excel, как это было несколько лет назад при организации различных локальных игровых соревнований. Соответственно, возникла необходимость в оптимизации данного процесса и интеграции в него различных алгоритмов. Для решения данной проблемы было разработано веб-приложение для организации и проведения онлайн соревнований по игре «GTA Online».

Для выявления проблем и потребностей пользователей был организован опрос среди сообщества Grand Racing League и по его результатам сформировано техническое задание для программного продукта.

Основные требования к программному продукту:

1. Возможность создания событий (гонок) с полной информацией о трассе и доступных машинах, даты проведения и ограничения кол-ва участников.
2. Система регистрации на гонки.
3. Пользовательская база машин с возможностью автоматического дополнения посредством парсинга открытых источников и удобной системой навигации.
4. Пользовательская база доступных трасс со ссылкой на официальный ресурс (Social Club [0]).
5. Возможность полуавтоматического добавления результатов, прошедших ивентов.
6. Единая база результатов.
7. Рейтинговая система участников.
8. Система мониторинга и модерации контента и событий.
9. Личный кабинет для пользователей.
10. Система оповещения о предстоящих гонках.
11. Возможность отслеживания погодных условий на сервере в момент проведения гонок.
12. Интеграция с discord сервером.

Анализ существующих на рынке решений показал, что нет аналогов, которые бы полностью отвечали требованиям сообщества из-за специфичной тематики.

Для реализации приложения была выбрана клиент-серверная архитектура, что добавит большей гибкости и масштабируемости. В качестве языка клиентской части был выбран язык JavaScript и фреймворк Vue.js [0], который повышает быстродействие системы, оптимизирует повторный рендеринг страниц и позволяет гибко управлять состоянием приложения.

При создании гибкого UI использован фреймворк Vuetify [0], основанный на концепции Material Design и созданный специально для Vue.js. Он обладает огромной библиотекой компонентов, благодаря чему удалось быстро реализовать весь необходимый функционал.

В качестве языка программирования для серверной части был выбран Kotlin [0], который является современной альтернативой языка Java, что также позволило сократить время реализации проекта. В связке с Kotlin использовался Spring framework для обеспечения удобного доступа к данным.

Общение между двумя модулями приложения будет происходить с помощью HTTP-запросов в соответствии с REST API [0] (от англ. Representational State Transfer – «передача состояния представления»). Так как основные виды взаимодействия с данными не будут

выходить за рамки действий CRUD (Create-Read-Update-Delete), поэтому принципов REST будет достаточно. Итоговая архитектура приложения представлена на рисунке 1.

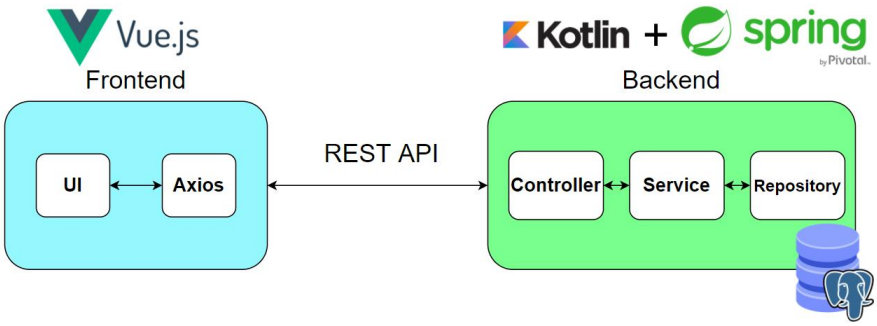


Рисунок 1 – Архитектура приложения

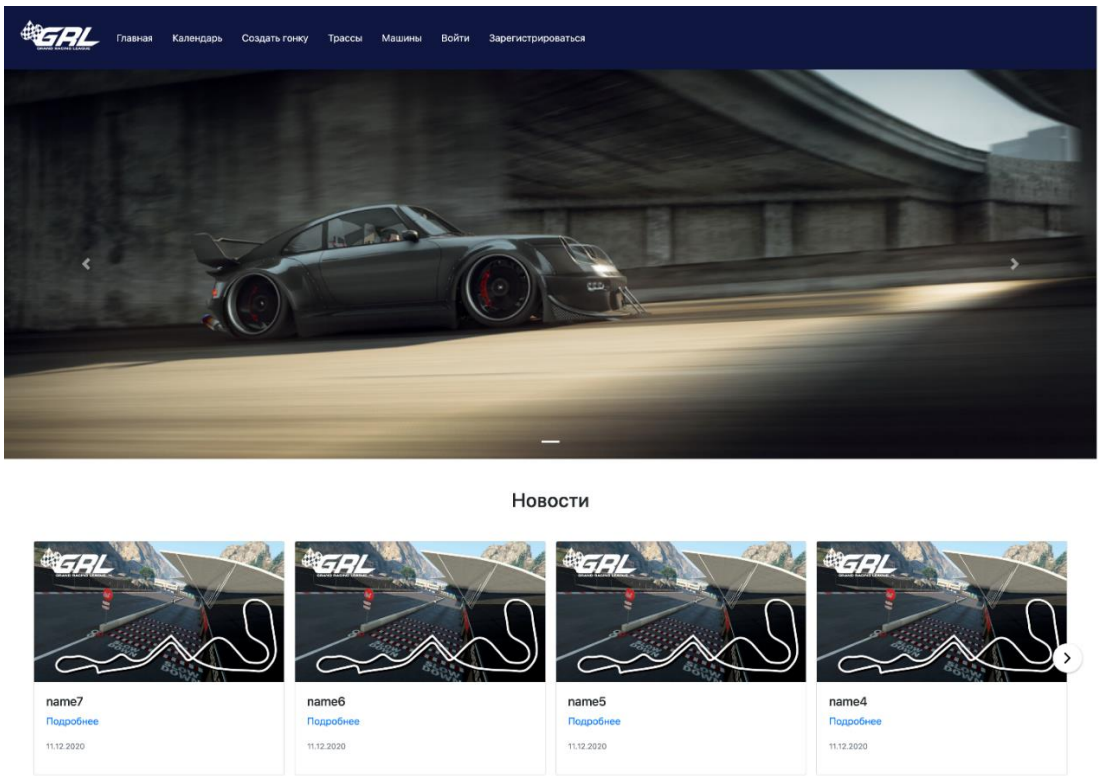


Рисунок 2 – Главная страница

Для прототипа был разработан дизайн веб-интерфейса (рисунок 2) и 25 базовых эндпоинтов для авторизации и CRUD-операций над гонками, трассами и машинами. Также были написаны unit-тесты с использованием Vue Test Utils и JUnit5. В качестве сборщика отчетов был использован Codecov [0].

В результате был реализован и протестирован программный прототип с частичным функционалом. В дальнейшем планируется добавить поддержку расширенного личного кабинета и более гибкую рейтинговую систему.

ЛИТЕРАТУРА

1. Официальный сайт Kotlin [Электронный ресурс] – URL: https://kotlinlang.org/ (дата обращения: 11.03.2020).
2. Официальный сайт Spring Framework [Электронный ресурс] – URL: https://spring.io/projects/spring-framework (дата обращения: 11.03.2020).

3. Официальный сайт Vue.js [Электронный ресурс] – URL: https://vuejs.org/v2/guide/ (дата обращения: 11.03.2020).
4. Официальный сайт Vuetify [Электронный ресурс] – URL: https://vuetifyjs.com/en/ (дата обращения: 11.03.2020).
5. REST API: https://www.redhat.com/en/topics/api/what-is-a-rest-api
6. RESTful Web Services Cookbook: Solutions for Improving Scalability and Simplicity 1st Edition, Yahoo Press, 2010, 320 с.
7. Официальный сайт Codecov [Электронный ресурс] – URL: https://about.codecov.io/ (дата обращения: 11.03.2020).
8. Официальный сайт Social Club [Электронный ресурс] – URL: https://socialclub.rockstargames.com/ (дата обращения: 11.03.2020).
9. Официальный сайт GTA Online [Электронный ресурс] – URL: https://www.rockstargames.com/GTAOnline/ru (дата обращения: 11.03.2020).

УДК 004.416.6

Зайцев Д.И., Пиккио П.Ф. (4 курс бакалавриата)
Графеева Н.Г., к. ф.-м.н., доцент

РАСШИРЕНИЕ ФУНКЦИОНАЛЬНОСТИ ANDROID ПРИЛОЖЕНИЯ

Целью работы является добавление в уже готовое Android приложение “Музыкальный плеер” возможность ранжирования композиций. Это расширение включает в себя настройку системы рейтинга аудиозаписей.

Весь проект представляется в виде архитектурной модели Model-View-Presenter, изображенной на Рисунке 1. View – часть, в которой обрабатываются действия, связанные с экраном, Presenter – отвечает за загрузку данных из модели и вызов соответствующих методов на уровне представления, когда данные загружены. Model – уровень данных. На Рисунке 2 приведен пример упрощенной UML диаграммы проекта.

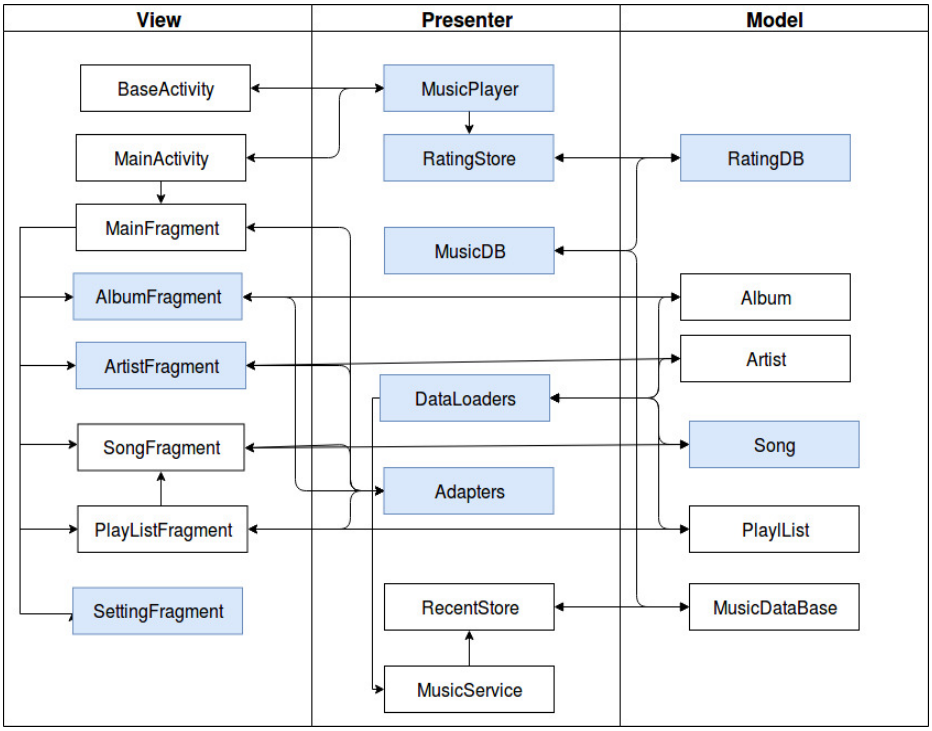


Рисунок 1 – Диаграмма MVP проекта. Синим цветом отмечены классы, в которые вносились изменения.

Используемые технологии:

Язык программирования Java, на котором было написано изменяемое приложение. Для добавления новой функциональности была использована распространенная среда разработки для Android приложений AndroidStudio 3.4. При разработке проекта использовался Android SDK 28.0.3. Для работы с базами данных музыкального плеера было решено использовать компактную встраиваемую СУБД SQL Lite.

Особенности реализации системы управления:

В качестве изначального приложения было выбрано “Timber”, поскольку оно являлось приложением с открытым исходным кодом на языке программирования Java.

Рейтинг, как и остальная информация о композиции, хранится в базе данных и представляет собой натуральное число от 0 до N. Во время тестирования приложения было проведено 4 эксперимента. В трех из них было решено использовать N равное 20, в четвертом – N равное 100. В первом случае пользователи тестировали приложение 4 дня, во втором неделю. Результаты показали, что при N равном 20, композиции ранжируются быстрее, но менее точно, чем при N равном 100. В готовом проекте был использован встроенный контент-провайдер ContentResolver. Он представлял ресурсы телефона в виде таблицы, которая использовала фиксированный набор столбцов, встроенных в библиотеку Android. Для добавления столбца с рейтингом, опираясь на принцип open-closed SOLID, было решено создать новую таблицу, связанную внешним ключом со старой.

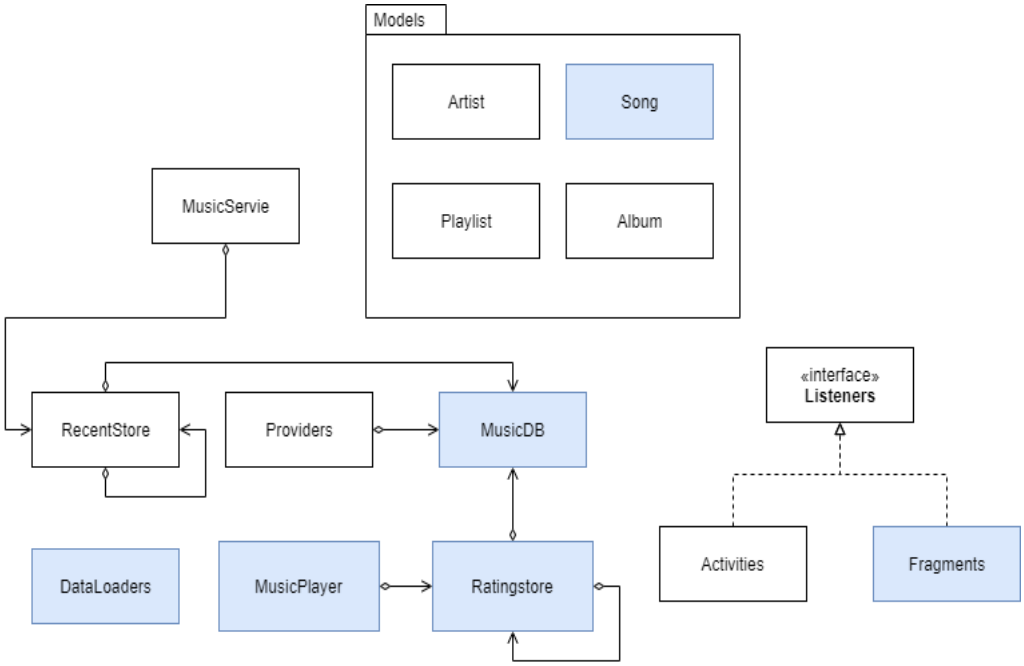


Рисунок 2 – Упрощенная UML диаграмма классов проекта. Синим цветом отмечены классы, в которые вносились изменения.

Во время разработки алгоритма было решено передать ответственность за изменение рейтинга кнопкам, отвечающим за управление воспроизведения композиции. Песню можно отнести к четырем непересекающимся группам: “очень нравится”, “нравится”, “не нравится” и “безразлична”. Рейтинг будет меняться только для тех композиций, которые относятся к первым трем группам. Если пользователь переходит к следующей композиции в течение 10 секунд прослушивания текущей, то было решено отнести ее к категории “не нравится”, и тогда рейтинг композиции уменьшается на 1. Если пользователь прослушивает композицию еще раз, не услышав начала следующей, то песня ему “очень понравилась”, и ее рейтинг увеличивается на 2. Если пользователь после конца песни решил послушать начало следующей композиции, а потом вернуться к предыдущей, то это означает, что первая песня ему “понравилась”, и ее рейтинг увеличивается на 1. Изменение рейтинга в последнем случае

сделано так, что при ошибочном переключении композиции пользователем рейтинг для двух песен остается неизменным.

Если рейтинг песни опускается ниже минимального значения, то на экране появляется диалоговое окно, предлагающее пользователю удалить композицию. Если пользователь решает ее не удалять, то рейтинг песни увеличивается на 1, чтобы быть равным минимальному значению допустимого рейтинга.

В процессе разработки возникла проблема обновления списка композиций. Предположим, человеку не понравилась композиция, которую он прослушал последней, после автоматической сортировки песня окажется ниже по списку трек-листа. Это говорит о том, что пользователь рано или поздно прослушает эту композицию еще раз. Чтобы этого избежать, было решено передать право упорядочивания списка композиций пользователю.

Таким образом, в приложение было успешно добавлена возможность упорядочивания композиций по рейтингу, Проведен эксперимент, показывающий качество работы алгоритма ранжирования.

ЛИТЕРАТУРА

1. Ян Ф. Дарвин Android. «Сборник рецептов. Задачи и решения для разработчиков приложений» – – 2017
2. Дэвид Гриффитс, Дон Гриффитс «Head First. Программирование для Android» — 2018
3. Климов Александр: Android Tutorial URL:http://developer.alexanderklimov.ru/android/
4. Климов Александр: SQLite URL:http://developer.alexanderklimov.ru/android/sqlite/

УДК 681.51

Иванов Д.А. (4 курс бакалавриата)
Петров А.В., ст. преподаватель
Каплан Е.В., ст. преподаватель

РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ УМНЫМ ДОМОМ С ПОМОЩЬЮ МОБИЛЬНОГО ПРИЛОЖЕНИЯ И СИСТЕМЫ АВТОМАТИЗАЦИИ HOME ASSISTANT

Целью работы является создание программного продукта, применяемого для управления системой умного дома (отдельными ее элементами: умными светильниками, датчиками и другими устройствами) посредством мобильного приложение на базе операционной системы IOS и системой автоматизации Home Assistant.

Большинство современных систем управления умным домом работают при помощи облачных сервисов производителей этих самых систем и обрабатывают запросы пользователей. В связи с этой особенностью можно выделить несколько недостатков:

– Долгая задержка между запросом и его выполнением
– Зависимость от наличия соединения с сервером
– Проблемы с безопасностью и возможность несанкционированного доступа к управлению
– Невозможность гибкой настройки обработки запросов

Для решения данных проблем было принято решение разработать собственную систему управления умным домом, достаточно защищенную и независимую от сторонних серверов. При анализе существующих продуктов было принято решение реализовывать систему на основе приложения Home Assistant.

Home Assistant - популярное приложение с открытым исходным кодом для организации умного дома. Приложение разрабатывается с 2013 года и распространяется по лицензии Apache License. И само приложение Home Assistant, и его программные расширения написаны на Python.

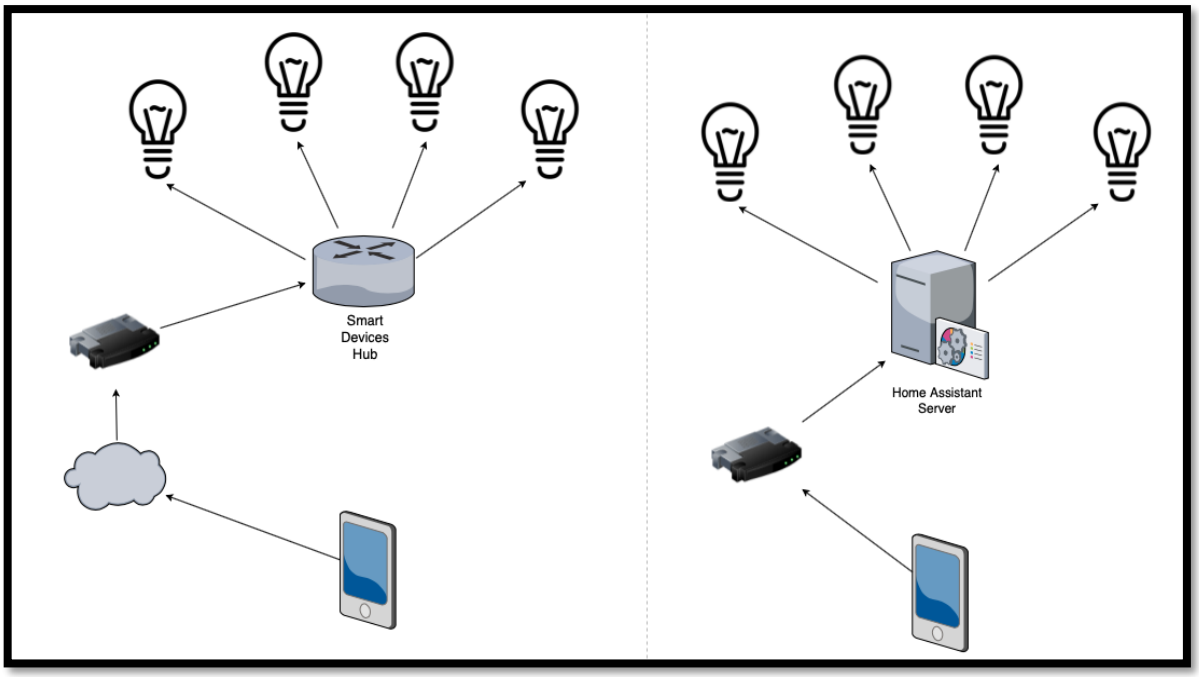


Рисунок 1 – Схемы управления элементами умного дома

На рисунке выше представлены 2 схемы управления умным домом: традиционная, предлагаемая большинством производителей (слева) и предлагаемая мной система (справа), с приложением на основе Home Assistant, запущенном на отдельном сервере. Создаваемый вариант не требует обращения к сторонним сервисам и при использовании в зоне действия локальной сети Wi-Fi остается работоспособной даже при полном отсутствии интернета.

Для работы Home Assistant в качестве сервера был выбран крайне популярный на данный момент одноплатный компьютер Raspberry PI, производительности которого вполне достаточно для решения поставленной задачи, вместе с тем он обладает низким энергопотреблением, малыми размерами и не требует активного охлаждения, что позитивно сказывается на его использовании. Данный компьютер обладает всеми необходимыми интерфейсами для взаимодействия с сетью, которые будут использоваться при разработке нашей системы.

Отличительной особенностью созданной системы управления является возможность отправки запросов и получения ответов от сервера с помощью СМС-сообщений, что позволит иметь возможность контролировать умный дом даже при отсутствии интернета на смартфоне.

Создание собственного программного продукта решает проблемы, возникающие при использовании сторонних сервисов контроля умного дома, и позволяет более гибко настраивать сценарии управления, открывая большие возможности для модернизации системы: улучшение безопасности за счет более совершенных алгоритмов шифрования, добавление систем, основанных на нейронных сетях, обучающихся в процессе эксплуатации, и многое другое.

ЛИТЕРАТУРА

1. Home Assistant [Электронный ресурс] URL: https://en.wikipedia.org/wiki/Home_Assistant
2. Home Assistant [Электронный ресурс] URL: https://www.home-assistant.io
   HOW TO SET UP HELPFUL HOME ASSISTANT AUTOMATIONS ON RASPBERRY PI [Электронный ресурс] URL: https://mediaexperience.com/home-assistant-raspberry-pi/
3. Raspberry PI [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Raspberry_Pi
4. Raspberry PI OS Official Documentation [Электронный ресурс] URL: https://www.raspberrypi.org/software/
5. Apache License [Электронный ресурс] URL https://en.wikipedia.org/wiki/Apache_License

УДК 004.051

Иванов В.О. (4 курс бакалавриата)
Малеев О. Г., к.т.н, доцент

ГЛУБОКОЕ ОБУЧЕНИЕ НЕЙРОННЫХ СЕТЕЙ ДЛЯ РАСПОЗНАНИЯ ИЗОБРАЖЕНИЙ

Идея нейронных сетей была позаимствована у природы, а точнее в качестве аналога использовалась нервная система животных и человека. Нейронные сети применяются для решения целого ряда задач, которые традиционными линейными методами трудно решаемы, или даже вообще не решаемы. На практике нейросети используются в двух видах – как программные продукты, выполняемые на обычных компьютерах, и как специализированные аппаратно-программные комплексы.

Целью работы является разработка нейронной сети для быстрого и точного распознавания объекта на изображении.

Нейронная сеть — это последовательность нейронов, соединенных между собой синапсами. Структура нейронной сети пришла в мир программирования прямиком из биологии. Благодаря такой структуре, машина обретает способность анализировать и даже запоминать различную информацию. Нейронные сети также способны не только анализировать входящую информацию, но и воспроизводить ее из своей памяти. Глубокое обучение — это метод машинного обучения. Глубокое обучение позволяет обучать модель предсказывать результат по набору входных данных. Для обучения сети можно использовать как контролируемое, так и неконтролируемое обучение.

Перечислим основные задачи, решаемые нейронными сетями:

1) Распределенная ассоциативная память;
2) Распознавание образов;
3) Адаптивное управление;
4) Прогнозирование;
5) Экспертные системы;
6) Оптимизация (т.е. поиск максимума функционала при наличии ограничений на его параметры),
7) Кластеризация, категоризация.

Некоторые проблемы при обучении нейронной сети:

1) Нужен большой набор данных.

2) Нужно большое количество вычислительной мощности.

Было создано приложение на основе нейронной сети, которая может различать различные виды животных. Данное приложение было создано для помощи человеку в определении вида животного и общей информации о нем. Так же не составит трудности для расширения списка животных, которое нейронная сеть может различить при помощи увеличения набора данных.

Сеть обучалась на примерах фотографий и названия животного на ней. Нейронная сеть состоит из 11 слоев:

- **Conv2D** — это 2D Convolution Layer, этот слой создает ядро свертки, которое является ветром с входными слоями, что помогает получить тензор выходных данных. При обработке изображений ядро представляет собой сверточную матрицу или маски, которые можно использовать для размытия, повышения резкости, тиснения, обнаружения краев и многого другого путем выполнения свертки между ядром и изображением.

- MaxPooling2D – слой подвыборки, снижающий размерность поступивших на него данных.

- Flatten - преобразует формат изображений из 2d-массива в 1d-массив. У этого слоя нет параметров для изучения, он только переформатирует данные.

- Dense - Это плотно связанные или полностью связанные нейронные слои.

В ней присутствует метод оптимизации типа «adam». Он сочетает в себе и идею накопления движения и идею более слабого обновления весов для типичных признаков. Так же использовалась метрика «accuracy». Обучение проходило на 20 эпохах и в среднем одна эпоха проходила в течение 300 секунд. В общей сложности нейросеть обучилась за 1 час 40 минут. При таком обучении ошибка правильного ответа была минимальной.

Модель нейронной сети:

```
Model: "sequential"
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
conv2d (Conv2D) (None, 148, 148, 32) 896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32) 0
_________________________________________________________________
conv2d_1 (Conv2D) (None, 72, 72, 64) 18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64) 0
_________________________________________________________________
conv2d_2 (Conv2D) (None, 34, 34, 128) 73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128) 0
_________________________________________________________________
conv2d_3 (Conv2D) (None, 15, 15, 128) 147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 128) 0
_________________________________________________________________
flatten (Flatten) (None, 6272) 0
_________________________________________________________________
dense (Dense) (None, 512) 3211776
_________________________________________________________________
dense_1 (Dense) (None, 2) 1026
=================================================================
Total params: 3,453,634
Trainable params: 3,453,634
Non-trainable params: 0
_________________________________________________________________
```

В заключении хочется отметить, нейронные сети быстро внедряются в нашу жизнь. Системы на основе самообучения все больше ценятся на рынке.

Для разработки программного продукта было изучено такое средство разработки нейронной сети как TensorFlow и Keras. Были рассмотрены различные алгоритмы и структуры, используемые при работе с картинками. В будущем планируется улучшить нейронную сеть, для повышения точности и увеличить количество распознаваемых животных на рынке.

ЛИТЕРАТУРА

1. Применение нейронных сетей для задач классификации https://basegroup.ru/community/articles/classification
2. Что такое TansorFlow https://www.computerworld.ru/articles/Chtotakoe-TensorFlow-i-kak-eto-ispolzuetsya
3. Нейронные сети: Основы http://synset.com/ai/ru/nn/NeuralNet_01 _Intro.html

УДК 004.93'12

Имхасина И.Х. (4 курс бакалавриата)
Самочадин А.В., к.т.н., доцент

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ЛЮБИТЕЛЬСКОГО НАБЛЮДЕНИЯ ЗА ПТИЦАМИ

Бердвотчинг (birdwatching) — это популярное увлечение, подразумевающее активное наблюдение за жизнью птиц в их естественной среде обитания. Доступность данного хобби позволяет заниматься им в одиночку или компаниями, проводить соревнования по бердвотчингу [1]. Кроме этого, бердвотчинг помогает научным организациям, которые занимаются мониторингом птиц [2].

Целью данной работы является создание мобильного приложения для упрощения процесса бердвотчинга, включающего в себя определение видов птиц, ведение статистики, а также возможность для помощи в соревнованиях.

Существуют несколько аналогов данного приложения, однако ни одно из них не выполняет все перечисленные функции.

Таблица 1 – Таблица сравнения существующих приложений

Название	Язык	Идентификация	Доступность	Картотека	Автоматизация соревнований
Picture Bird	рус/ англ	фото	платно	журнал	-
Merlin	англ	фото, фильтры	бесплатно	журнал	-
BI	англ	фото	платно	-	-
Ptushki	рус	фильтры	бесплатно	-	-

Схема приложения, реализующего всю необходимую функциональность, представлена на рисунке 1.

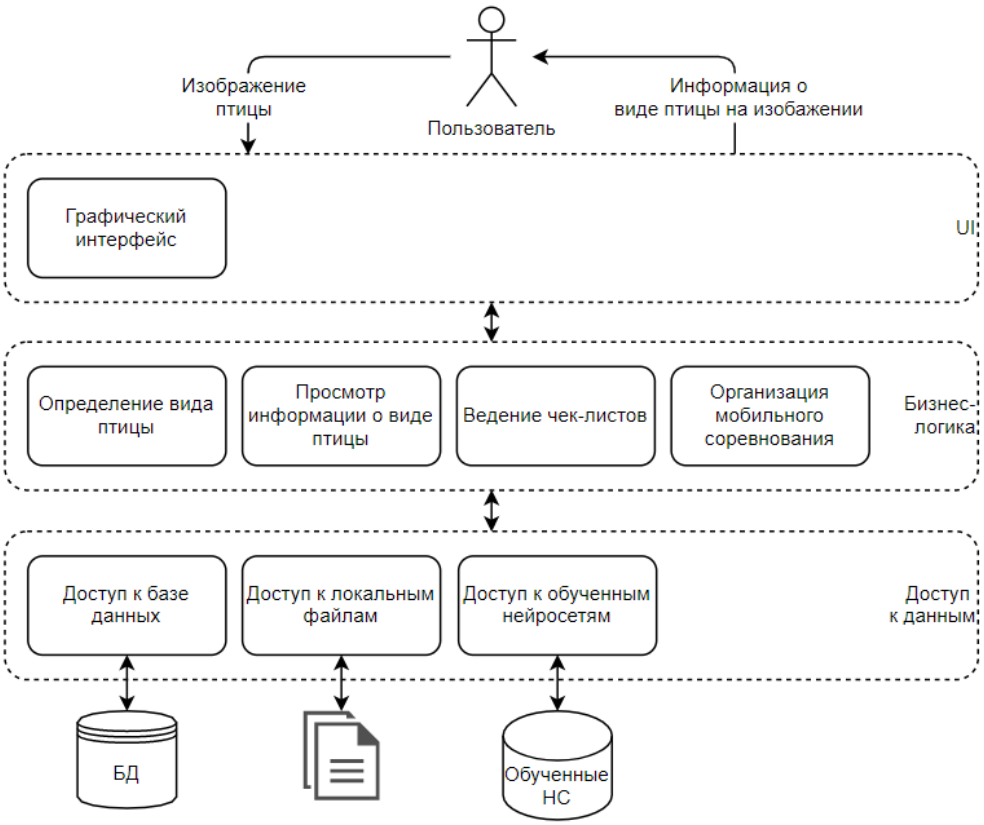


Рисунок 1 – Схема работы приложения

В качестве операционной системы выступает Android, имеющая наибольшее распространение среди пользователей мобильных устройств.

Распознавание птиц происходит с помощью сверточных нейросетей, обученных с использованием бесплатных датасетов от Kaggle и дополнительно собранных [3].

Осуществление работы картотеки происходит с использованием баз данных, имеется возможность к выгрузке.

Проведение соревнований по бердвотчингу осуществляется с использованием технологии GPS.

ЛИТЕРАТУРА

1. Джордж Коннел. Birdwatching, Twitching and Tourism. 2009. С. 203.
2. Ян Чжоу, Анпин Чен. Comparing community birdwatching and professional bird monitoring with implications for avian diversity research [Электронный ресурс]. URL: https://avianres.biomedcentral.com/articles/10.1186/s40657-020-00205-w (дата обращения: 11.03.2021).
3. Хайкин С. Нейронные сети. – М.; СПб.; Киев, 2006. – 89 с

УДК 004.91

Катричко У.Д. (2 курс бакалавриата)
Литвинов Ю.В., к.т.н., доцент

РЕАЛИЗАЦИЯ ГЕНЕРАТОРА РАСПОРЯЖЕНИЙ О РАСПРЕДЕЛЕНИИ СТУДЕНТОВ НА ДИСЦИПЛИНЫ ПО ВЫБОРУ

Каждый государственный университет является организацией, в которой протекают процессы, требующие документального сопровождения, СПбГУ не является исключением. В стенах университета обучаются и работают десятки тысяч человек, поэтому объем делопроизводства огромен. Несмотря на то, что проделана работа по его автоматизации, преподавателям и сотрудникам учебных отделов все еще приходится выполнять трудоемкие и рутинные задачи по составлению различных документов.

Одной из таких задач является создание распоряжения о распределении студентов на курсы по выбору. Каждый год более двухсот пятидесяти учащихся математико-механического факультета в электронном виде выбирают дисциплины для изучения на будущий учебный год. После получения всех заявлений сотрудник учебного отдела приступает к их обработке. Обращаясь к информации из четырех источников, он создает распоряжение, содержащее список дисциплин, с зачисленными на каждую из них студентами.

Чтобы понять, можно ли распределять студентов и генерировать распоряжение с помощью программных средств, был проведен сбор бизнес-требований и анализ имеющихся возможностей автоматизации.

Сделан обзор процесса выбора элективов студентами:

учебные планы содержат вариативную часть обучения, представленную дисциплинами по выбору, которые организованы в блоки. Учащийся должен изучить ровно столько дисциплин из каждого доступного ему блока, сколько указано в учебном плане;

выбор элективов из конкретного блока может быть возможен как всеми студентами направления, так и учащимися исключительно определенной специализации;

чтобы высказать свои пожелания, студенту необходимо зайти в личный кабинет и присвоить дисциплинам приоритеты от 1 до N, где N – число элективов в блоке;

после обработки заявлений учащийся в личном кабинете может увидеть, на какую дисциплину он зачислен. В начале семестра, можно изменить электив для изучения путем отправки заявления, содержащего наименование выбранной ранее дисциплины и желаемой, но оно может быть отклонено, а значит изучаться будет утвержденный изначально электив.

Таким образом студенты подают заявления о выборе дисциплины и о ее изменении.

В ходе интервьюирования сотрудника учебного отдела было выяснено, что для создания распоряжения необходимы следующие документы: учебные планы, заявления учащихся, рабочие программы дисциплин и документ с информацией о каждом из студентов, участвующем в распределении.

Кроме того, были сформулированы требования к процессу распределения учащихся:

число студентов, зачисленных на электив, должно соответствовать нормативам. Если число желающих посещать курс, меньше необходимого, то дисциплина не будет преподаваться. Если желающих изучать электив больше допустимого, то нужно зачислить студентов с наибольшим средним баллом;

требуется распределить всех учащихся конкретного направления, не ушедших в академический отпуск, включая тех, кто не подавал заявления.

Путем опроса сотрудника управления-службы информационных технологий СПбГУ определены возможности имеющихся средств автоматизации. Сделан вывод о том, что хоть они недостаточны для облегчения ручного процесса, их функции, такие как выгрузка заявлений и данных о студентах в форматах docx и xlsx, позволят создать инструмент, облегчающий процесс распределения студентов. Таким образом, задачей данной работы стала реализация генератора распоряжений о распределении учащихся на курсы по выбору.

Инструментом разбора был выбран OpenXML SDK, поскольку в сравнении с NPOI, EPPlus, ClosedXML и другими аналогами, только он имел подробную документацию и умел работать, как с форматом docx, так и xlsx. В качестве системы чтения использован Document Object Model подход, так как, в отличие от Simple API for XML подхода, с его помощью легко оперировать сложной структурой документа, писать наглядный код.

Для реализации генератора распоряжений были созданы следующие модули:

Анализатор учебных планов. Позволяет создать объектно-ориентированную модель учебного плана. Его особенность заключается в том, что он используется не только для генератора распоряжений, но и для автоматического составления кадровой справки государственной аккредитации, редактирования и проверки корректности рабочих программ дисциплин.

Парсер заявлений студентов. Способен разбирать заявления о выборе электива и его изменении. Детектирует три типа ошибок, нарушающих инвариант заявления. Например, студент высказал желания по изучению дисциплин конкретного направления, а затем перевелся на другую программу обучения. Заявления с ошибками сохраняются, при генерации распоряжения информация о них будет сообщена сотруднику учебного отдела.

Парсер документа, содержащего сведения обо всех учащихся факультета, составляет его объектно-ориентированную модель, позволяющую извлекать данные об успеваемости, направлении обучения и специализации студентов, а также списки людей, ушедших в академический отпуск, заявления которых не должны подлежать обработке.

Два алгоритма распределения студентов на элективы. Являются формализацией процесса, распределения осуществлявшегося вручную. Так, алгоритм распределения на основе заявлений о выборе сначала утверждает все заявления с первым приоритетом, затем решает проблемы несоответствия количества зачисленных студентов нормативам, далее распределяет никуда не зачисленных к данному моменту студентов.

Генератор файла распоряжения создает документ, в котором содержится информация о студентах, при разборе заявлений которых возникла ошибка, а также причина ее появления. Далее печатается список дисциплин с зачисленными на каждую из них студентами в требовавшемся формате.

Графический интерфейс создан с использованием технологии WPF. Для его реализации было принято решение об использовании паттерна Model-View-ViewModel, позволяющем отделить бизнес-логику от визуальной части, благодаря чему сопровождение, тестирование, читабельность кода становятся проще.

Апробация была проведена на заявлениях направлений обучения прошлых лет. Распределение либо полностью совпадало с тем, что было в официальном приказе, либо существовали незначительные различия, связанные с неактуальностью данных о студентах, выгружаемых на момент апробации. В апреле 2021 года генератор будет использоваться для решения реальных задач, поставленных перед учебным отделом.

Исходный код приложения доступен по ссылке https://github.com/resueman/decree-generator (дата обращения 16.03.2021).

ЛИТЕРАТУРА

1. Office Open XML, URL: https://ru.wikipedia.org/wiki/Office_Open_XML (дата обращения 16.03.2021)
2. Anatomy of a WordProcessingML File, URL: http://officeopenxml.com/anatomyofOOXML.php (дата обращения 16.03.2021)

УДК 004.414.32

Кауана Хил Л. А. (4 курс бакалавриата)
Коликова Т. В., ст. преподаватель

ПРИЛОЖЕНИЕ ДЛЯ ПРОГНОЗИРОВАНИЯ ПОВЕДЕНИЯ ФОНДОВЫХ РЫНКОВ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ

Целью работы является разработка приложения для прогнозирования поведения фондовых рынков с использованием нейронных сетей. Это приложение должно предсказать тенденцию, которая будет иметь цену закрытия в определенный период времени. В нем будет возможно получить доступ к графическому интерфейсу (GUI), где можно визуализировать данные, сгенерированные в прогнозе в соответствии с обученной сетью.

В графическом интерфейсе (GUI) пользователь также может ввести определенные индикаторы. Помимо этого, для удобства пользователей различной национальности будет предусмотрена поддержка английского языка, являющегося интернациональным. В этом приложении будут использоваться три сети, которые:

- Сети pattern recognition, которые использовались для прогнозирования тенденции цены закрытия.
- Сети Time series analysis, которые будут использоваться для прогнозирования точного значения цены закрытия.
- Сети fitting, которые также будут использоваться для прогнозирования точного значения цены закрытия.

В качестве интегрированной среды разработки выступает MATLAB, разработанная компанией MathWorks. MATLAB также является языком высокого уровня, который включает в себя структуры данных, функции, операторы потока управления, ввод/вывод и объектно-ориентированные возможности. Что касается данных для обучения, мы будем использовать данные, предоставленные Finance Yahoo, период времени данных будет 10 лет, так как использование более длительного периода не удобно динамикой рынка.

ЛИТЕРАТУРА

1. HOWARD DEMUTH AND MARK BEALE, «Neural Network Toolbox for Use with MATLAB®», version 4.
2. The MathWorks, Inc – URL: https://www.mathworks.com/
3. Finance Yahoo – URL : https://finance.yahoo.com/

УДК 004.414.32

Клименко М.А. (4 курс бакалавриата)
Коликова Т.В., ст. преподаватель

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ПОД ОПЕРАЦИОННУЮ СИСТЕМУ ANDROID ДЛЯ ПОДДЕРЖАНИЯ ЗДОРОВОГО ОБРАЗА ЖИЗНИ

Целью работы является проектирование и разработка мобильного приложения для поддержания здорового образа жизни и отслеживания прогресса тренировок. Функционал приложения позволит отслеживать калорийность продуктов питания, уровень потребляемой воды в течение дня и прогресс в потере, поддержании или наборе веса. Также будет реализована возможность использовать таймер, работающий по заданному плану тренировки, возможность включить напоминания о понижении уровня воды в организме.

Похожие продукты уже присутствуют на рынке приложений, но каждая из функций данного проекта реализована как отдельное приложение. Преимуществом данного продукта является объединение функций из разных приложений в одном, доработка функционала, а также добавление нового.

Интерфейс приложения будет содержать несколько вкладок для каждого функционала. Пользователь будет иметь возможность указывать текущий и желаемый вес, отслеживать его, регулярно вводя новые показания, вести дневник питания, выбирая продукты из списка по калорийности и составлять план тренировок из предложенных упражнений. Для списка продуктов и упражнений будет реализована база данных.

В качестве программной среды разработки выступает Android Studio, разработанная компанией Google для работы с платформой Android. Для программирования был выбран язык программирования Kotlin. Для построения базы данных будет использоваться сервис Firebase. Он служит базой данных, которая изменяется в реальном времени и хранит данные в формате JSON.

ЛИТЕРАТУРА

1. Android Developer Guides - https://developer.android.com/guide
2. Kotlin docs - https://kotlinlang.org/docs/home.html
3. Kousen, K. Kotlin Cookbook: A Problem-Focused Approach - O'Reilly Media; 1st edition, 2019. – 254 с.

УДК 004.415

Ладыгина П.А. (4 курс бакалавриата)
Самочадин А.В., к.т.н., доцент

РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА ДЛЯ ПРОГНОЗИРОВАНИЯ ПРОЧНОСТИ БЕТОНА С ПРИМЕНЕНИЕМ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

Прочность бетона – важная характеристика, применяемая при проектировании железобетонных конструкций для строительства различных объектов. Данную характеристику принято определять экспериментальным методом, с помощью измерений минимальных усилий, необходимых для разрушения цилиндрического образца [1].

Применяемые на практике подходы к прогнозированию прочности опираются на эмпирические зависимости, основанные на анализе экспериментальных данных, и дают лишь приблизительный результат. Для получения более точных прогнозных моделей, учитывающих нелинейный характер зависимостей, могут быть использованы современные методы на основе машинного обучения. Применение моделей машинного обучения на достаточно большом наборе данных позволит в перспективе улучшить точность прогноза прочности бетона и сократить количество необходимых экспериментальных проверок при

подборе состава [2]. Реализация программного средства для прогнозирования прочности с применением данных методов поможет инженерам-проектировщикам подбирать необходимый состав бетонной смеси с меньшей затратой ресурсов и в целом автоматизировать процесс проектирования состава.

Таким образом, целью работы является разработка программного средства для прогнозирования прочности бетона на сжатие в зависимости от его состава с применением методов машинного обучения.

В программном средстве имеется возможность применения как традиционных методов расчета прочности, так и прогнозных моделей машинного обучения, на выбор пользователя. С целью повышения точности прогнозов в приложении присутствует возможность дообучения моделей с помощью датасета, загружаемого пользователем. Кроме того, осуществляется хранение результатов прогнозирования в базе данных. Схема архитектуры приложения представлена на рисунке 1.

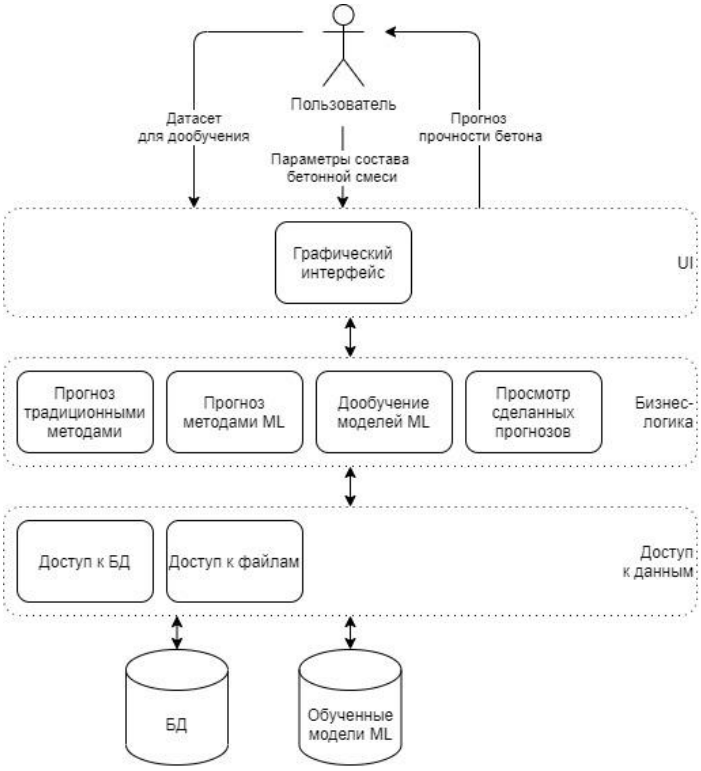


Рисунок 1 – Схема архитектуры приложения

Для обучения моделей используется набор данных из репозитория UCI [3]. Датасет содержит 1030 записей о разных составах бетона с восемью параметрами, которые описывают компоненты бетонной смеси.

Обучение происходит средствами языка Python, а именно – библиотеки sklearn. На вход модели подается набор параметров состава бетона, а также период набора прочности в сутках. Выходным параметром является прочность бетона на сжатие в МПа.

Для реализации приложения был выбран язык C#, а в качестве СУБД выступает MS SQL Server. Графический интерфейс приложения реализован средствами Windows Forms.

ЛИТЕРАТУРА

1. СП 63.13330.2018 Бетонные и железобетонные конструкции. Основные положения. СНиП 52-01-2003
2. P. Ziolkowski, M. Niedostatkiewicz. Machine Learning Techniques in Concrete Mix Design, Materials, 2019
3. Concrete Compressive Strength Data Set [Электронный ресурс]: URL: https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength (дата обращения: 15.03.2021)

УДК 004.891.2

Коновалов И.И. (1 курс аспирантуры)
Воинов Н.В., к.т.н., доцент

РАЗРАБОТКА ИНТЕРАКТИВНОГО ПОМОЩНИКА ДЛЯ КОНСУЛЬТИРОВАНИЯ УЧАЩИХСЯ В ЗАДАЧАХ ЛИНГВОДИДАКТИКИ

Целью работы является создание интеллектуального интерактивного помощника, позволяющего в непрерывном и автоматическом режиме обрабатывать информацию и консультировать изучающих иностранный язык. В соответствии с поставленной целью необходимо решить следующие задачи: анализ современных методов и требований к реализации конечного автомата и нейронной сети; формирование требований к разрабатываемому приложению; разработка программного приложения; тестирование разработанного приложения на определение некоторых интентов и сущностей.

В простейшей интерпретации интеллектуальный помощник (или чат-бот) представляет из себя конечный автомат с встроенным модулем распознавания естественного языка (Natural Language Understanding, NLU). При реализации подхода, названным «Состояние», становится возможным хранить текущее положение системы и набор данных, на основе которых определяется следующее состояние всей системы или одной из ее частей, в частности блока NLU. Благодаря данному подходу реализуется требование обеспечения сохранения контекста в процессе общения с пользователем. [1]

Блок распознавания естественного языка может быть реализован одним из множества методов машинного обучения, начиная от семантических деревьев и заканчивая нейронными сетями. В частности, в данной работе был реализован прототип рекуррентной нейронной сети с использованием метода обратного распространения ошибки для определения и обновления весов при определении слова к той или иной части речи. Для определения интентов и сущностей и оценки весов используется также STS-классификатор, т.е. оценка расстояния между словом и сущностью или интентом происходит на основе значения смыслового содержания, а не лексикографическом сходстве.

Под интентом понимается намерение пользователя, проблема, с которой он обращается в попытке передать боту заботы о ее решении. Чаще всего они выражены глаголами. Например, во фразе «Хочу изучить английский язык» намерением пользователя является слово «изучить», а слово «хочу» выступает эмоциональным окрасом намерению, т.е. сентиментом. Одновременно с задачей определения интента решается и задача поиска именованных сущностей (Named Entity Recognition, NER). Здесь нужно определить, например, что из частей предложения есть дата, имя собственное, местоположение или доменно-зависимый объект. Расширив предыдущий пример (рисунок 1), можно, в частности, определить, что планируется изучение английского языка в городе Москва в апреле месяце за небольшую цену. При этом определяются как общеизвестные сущности (город, месяц), так и доменные сущности (качественное определение стоимости)

Рисунок 1 – Распознавание интентов и сущностей

Создание «разметки» для лексем естественного языка – дорогое удовольствие. Для русского языка, в частности, существует единственный корпус FactRuEval 2016. [2] Он описан в нескольких статьях, является специфичным и включает всего 50000 токенов. [3] Целью данной работы не является участие в соревнованиях по разметке, практическое же применение корпуса в работе невозможно, поскольку предполагает лишь определение локации. [4] Для работы в данном домене можно сформировать ограниченный корпус, либо же решить задачу NER с помощью нейронных сетей.

Текст запроса разбивается на токены – отдельные слова. Затем вычисляются контекстно-независимые признаки для каждого токена: эмбеддинг (отображение дискретного вектора признаков в вектор заданной размерности, символьные признаки и дополнительные признаки (части речи, морфологические признаки). Например, символьные признаки можно сформировать, используя сверточную нейронную сеть (тип CharCNN) и расширяя эмбеддинги через пробельные символы («паддинги»). Далее они пропускаются через свертку малой размерности, проводится операция “max pooling”. Итоговый вектор размерности количества фильтров содержит информацию о символах слова и их взаимодействии (рисунок 2)

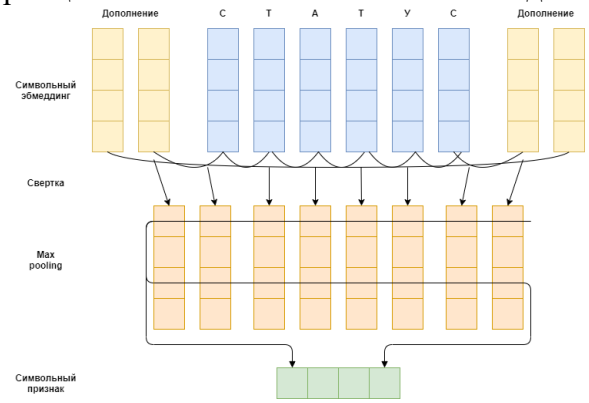


Рисунок 2 – Сверточная нейронная сеть для формирования символьных признаков токена

Конечный автомат реализуется с помощью YAML-подобного доменного языка сценариев, позволяющего задавать состояния и переходы между ними. Блок NLU для распознавания интентов и сущностей в добавление к готовым наборам дополняется доменными сущностями и обучается на нейронной сети. Результаты для тестового предложения приведены на рисунке 3.

Рисунок 3 - Результаты для тестового предложения

Таким образом был реализован прототип программного сервиса для дальнейшего исследования в определении частей речи и консультирования учащихся при изучении естественного языка. Дальнейшая работа предполагает, в частности, разработку и использования языкового корпуса для ускорения работы прототипа, и увеличение качества распознавания интентов и сущностей.

ЛИТЕРАТУРА

1. Карпов Ю.Г. Теория автоматов. Учебник для вузов // СПб: Питер – 2003 – 208с.
2. factRuEval-2016 // Dialogue Evaluation, Github [Электронный ресурс, репозиторий]
3. Statostin A.S., Bocharov V.V., Alexeeva S.V. et al. FactRuEval 2016: Evaluation of Named Entity Recognition and Fact Extraction Systems for Russian // Moscow - Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference “Dialogue 2016
4. FactRuEval — соревнование по выделению именованных сущностей и извлечению фактов // Блог компании ABBYY, Хабр [Электронный ресурс] // Режим доступа - https://habr.com/ru/company/abbyy/blog/273965

УДК 004.588

Устинов А.Г. (3 курс бакалавриата), Лебедев Д.М. (2 курс бакалавриата)
Эйзенах Д.С. (2 курс магистратуры)
Маслаков А. П., ст. преподаватель

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ПОД ОПЕРАЦИОННУЮ СИСТЕМУ IOS НАПРАВЛЕННОЕ НА РАЗВИТИЕ КОГНИТИВНЫХ СПОСОБНОСТЕЙ

Согласно результатам исследования сервиса “WEB-CANACE” [1], половина пользователей проводит за смартфонами более 5 часов, а 26% — более 7 часов в день. Люди тратят практически половину своего дня на времяпрепровождения в смартфонах, а именно, игры, главный источник траты времени. Более чем у 2 миллиардов пользователей есть IPhone, который может улучшить их жизнь, а не просто тратить время пользователя на игры.

В рамках данной работы была поставлена цель - разработать мобильное приложение на развитие когнитивных способностей. “BrainStorm” – мобильное приложение под операционную систему IOS, которое включает в себя совокупность различных разделов, которые помогают развиваться человеку во всех сферах, что позволяет продуктивно провести время.

Приложение предполагает четыре вида активностей:

1. “Arcade”, представляет собой набор мини-игр для развития когнитивных способностей;
2. “Facts”, позволяет пользователю просматривать интересные факты в разных областях;
3. “BrainStorm” - интерактивная игра вопрос/ответ, в случае ошибки, игра начинается сначала;
4. “School task”- викторина из школьной программы по разным предметам.

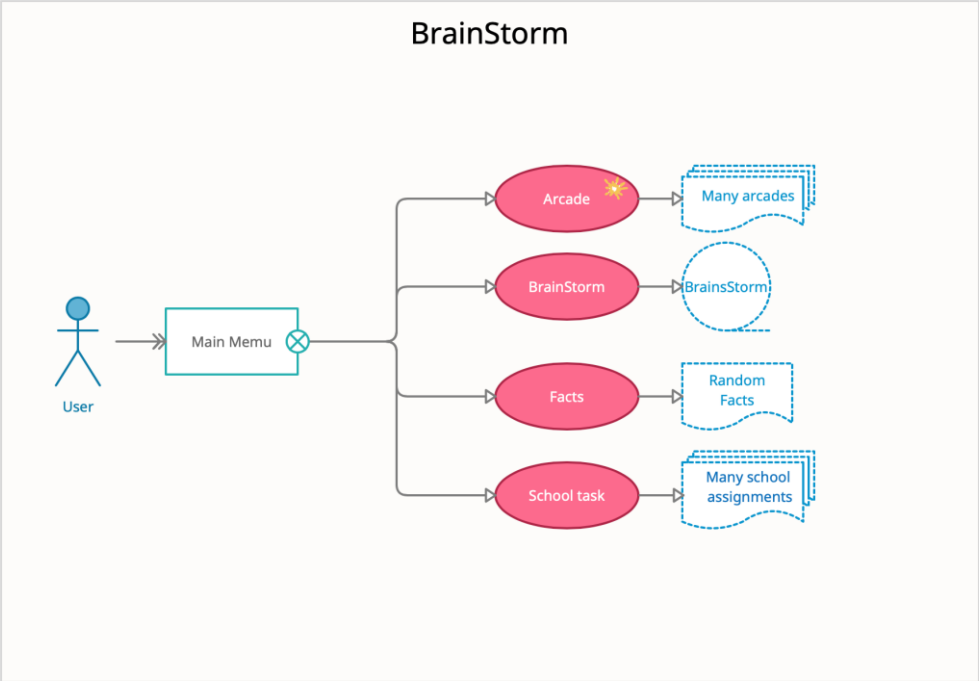


Рисунок 1 – UML диаграмма приложения BrainStorm

В качестве инструмента разработки был выбран нативный язык - Swift, который наиболее оптимизирован для разработки под операционную систему IOS [2]. Для отрисовки компонентов пользовательского интерфейса была выбрана библиотека UIKit, т.к. она предоставляет больший спектр возможностей для работы с UI компонентами по сравнению с другими библиотеками. Упростить процесс написания кода помог фреймворк Foundation [3], предоставляющий доступ к основным типам данных, коллекциям и службам операционной системы. Для корректной работы приложения требуется загружать данные из сети интернет,

для таких разделов приложения как “Facts” и “School task”, для этого мы использовали библиотеку Alamofire [4]. Для удобной работы с изображениями была выбрана библиотека Kingfisher, которая позволяет сразу использовать бинарный тип данных в отрисовки UIImage. Для хранения различных фактов и заданий для разделов “Facts” и “School task” был выбран формат json [5], как один из самых удобных и распространенных на данный момент форматов хранения и обмена информацией. Для работы с json структурой была использована библиотека SwiftyJSON [6], которая позволяет удобно преобразовать данные для дальнейшего использования.

В ходе данной работы было реализовано мобильное приложение под операционную систему IOS - “BrainStorm” для развития когнитивных способностей человека. В планах по развитию приложения - увеличение числа доступных мини-игр и новых видов активностей.

ЛИТЕРАТУРА

1. Вся статистика интернета на 2019 год – в мире и в России/ Юлия Сергеева [Электронный ресурс] – URL: https://www.web-canape.ru/business/vsya-statistika-interneta-na-2019-god-v-mire-i-v-rossii/ (дата обращения: 11.03.2020).
2. Mark, Dave iOS 5 SDK. Development of applications for iPhone, iPad and iPod touch / Dave Mark, Jack Nutting, Jeff Lamarsh. - M .: Williams, 2012 .-- 672 p. [стр. 20]
3. [Электорнный ресурс] - URL: https://developer.apple.com/documentation/foundation (дата обращения: 11.03.2020).
4. [Электронный ресурс] – URL: https://github.com/Alamofire/Alamofire (дата обращения: 11.03.2020).
5. [Электронный ресурс]-URL: https://www.json.org/json-ru.html (дата обращения: 12.03.2021)
6. [Электронный ресурс] – URL: https://github.com/SwiftyJSON/SwiftyJSON (дата обращения: 11.03.2020).

УДК 004.75

Лукин А.А., Мироненков Г.В. (2 курс магистратуры)
Воинов Н.В., к.т.н., доцент
Каплан Е.В., ст. преподаватель

ПРОГРАММНО-АППАРАТНЫЙ КОМПЛЕКС КОНТРОЛЯ КАЧЕСТВА ПИЩЕВОЙ ПРОДУКЦИИ

В настоящее время среднестатистический человека потребляет около 800 килограммов пищевых продуктов в год. Данная практика неизбежно приводит к появлению большого количества пищевых отходов. Кроме того, испорченная еда является прекрасной средой для размножения микроорганизмов, некоторые из которых могут нанести серьёзный ущерб здоровью. К сожалению, человек не всегда может определить, насколько тот или иной продукт безопасен для употребления.

Для решения данной проблемы была разработана концепция программно-аппаратного комплекса для контроля качества пищевой продукции. Данный комплекс состоит из следующих модулей:

1. Набор сенсоров, собирающих данные с образцов. Данные сенсоры должны собирать информацию о цвете, отражающей способности, химических соединениях в атмосфере, спектре и положении.

2. Микроконтроллер, принимающий данные с сенсоров и обрабатывающий их для унификации.

3. Анализатор, на который будут передаваться обработанные данные для определения состояния пищевой продукции.

В ходе работы был спроектирован и собран прототип программно-аппаратного комплекса на основе Arduino UNO [1,2]. Использовались следующие сенсоры:

1. Цифровой датчик яркости TSL2541 для измерения интенсивности света в тестовой камере.
2. Дальномер TMF8701 для определения местоположения тестового образца.
3. Спектрометр AS7341 для изучения спектра цвета тестового образца.
4. Электронный нос ENS160 на основе полупроводниковых оксидов металлов для улавливания изменения состояния воздуха в тестовой камере.

Связь сенсоров с микроконтроллером осуществлялась посредством шины I2C. Полученные данные поступают на микроконтроллер, где они сериализуются для дальнейшей передачи анализатору. В качестве анализатора выступает имитационная нейронная сеть, которая, основываясь на полученной от микроконтроллера информации, делает вывод о состоянии тестового образца.

В ходе экспериментов были получены данные для обучения имитационной нейронной сети.

На графике на Рисунке1 заметен тренд к снижению сопротивления полупроводникового оксида металла, что свидетельствует о росте концентрации этилена в тестовой камере. Пики на графике вызваны перекалибровкой сенсора.

В дальнейшей работе планируется произвести калибровку сенсоров и обучение имитационной нейронной сети.

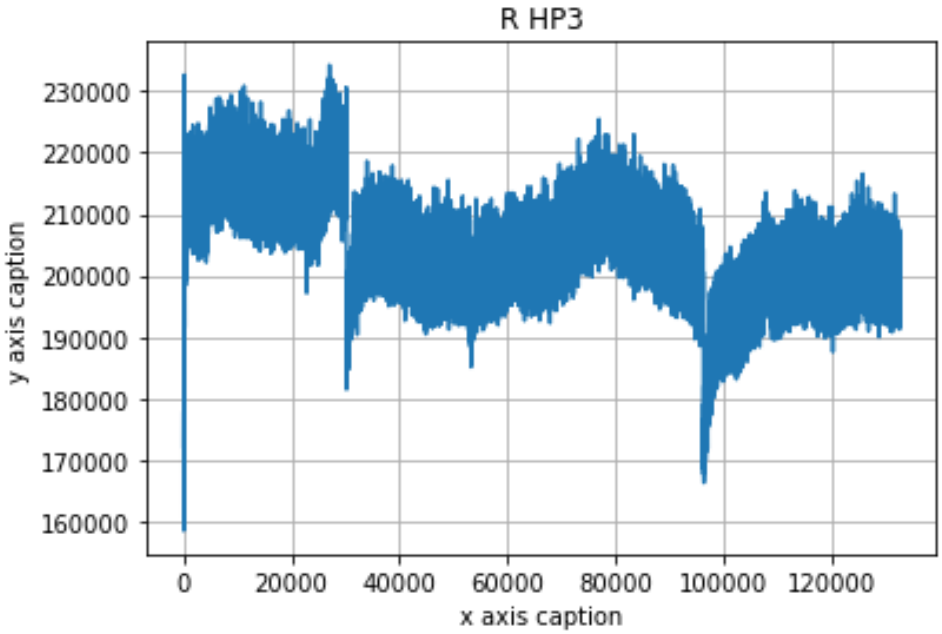


Рисунок 1 – График изменения сопротивления полупроводниковых оксидов металлов.

ЛИТЕРАТУРА

1. П. А. Панков, И. В. Никифоров, Д. Ф. Дробинцев. Исследование эффективности использования одноплатных микрокомпьютеров в качестве обучающей платформы для работы с распределённой обработкой данных. (с. 120) / Современные технологии в теории и практике программирования: сборник материалов конференции, 23 апреля 2020 года / Санкт-Петербургский политехнический университет Петра Великого, [Институт компьютерных наук и технологий [и др.] ; Санкт-Петербург : [ПОЛИТЕХ-ПРЕСС], 2020. 175 с. ISBN 978-5-7422-6935-9.
2. Панков П.А., Никифоров И.В., Дробинцев Д.Ф. Программно-аппаратный комплекс обработки данных для исследовательских и научных целей с использованием микрокомпьютера Raspberry Pi
3. Труды Института системного программирования РАН. 2020;32(3):57-69. https://doi.org/10.15514/ISPRAS-2020-32(3)-5.

УДК 004.454

Лелюхин Д.О. (2 курс магистратуры)
Федоров С.А., ст. преподаватель

РАЗРАБОТКА ПО В ВИДЕ СЕРВИСА ДЛЯ УПРАВЛЕНИЯ КАМЕРАМИ МАШИННОГО ЗРЕНИЯ XIMEA XIX

Целью работы является разработка программного обеспечения в виде сервиса для использования в блоках обнаружения препятствий беспилотных электропоездов ЭС2Г “Ласточка” камер машинного зрения Ximea xiX. Камеры поставляются вместе со штатной библиотекой xiAPI [5], реализованной в виде программного интерфейса [3] для управления захватом кадров, установкой и получением параметров камеры, световыми индикаторами, створками затвора и параметрами линзы.

Рисунок 1 – Камера машинного зрения Ximea xiX с интерфейсом PCIe Gen2

Разработанное программное обеспечение должно осуществлять взаимодействие с камерой Ximea с учетом особенности управления моделью xiX путем отправки команд в определенной последовательности, проверки доступности в сети, сбора диагностической информации о работе устройства, регулирования программных сбоев [1] и поддержания стабильной частоты выдачи данных при захвате кадров, используя вычислительные возможности CPU/GPU.

Особенности реализации сервиса управления:

Особенности реализации сервиса управления:

Взаимодействие с камерой осуществляется через аппаратный интерфейс подключения PCIe Gen2 с более высокой пропускной способностью по сравнению с остальными. Камера поставляется с библиотекой xiAPI [5], реализованной Ximea на языке C++11, однако, каждая компания, закупающая эти камеры, реализует ПО под свои требования. Предполагается использование сервиса для управления стереопарой из двух камер, работающих в режимах master/slave. Реализация сервиса ведется с использованием стандарта программирования C++17 [2], системой сборки CMake 3.10. Для фильтрации шумов и изменения размера изображения используется библиотека компьютерного зрения OpenCV 4.5.1 [4] и библиотека параллельных вычислений CUDA 10.2 в зависимости от типа захватываемых кадров.

Для передачи данных кадров и сообщений сервисам стереозрения используется микро-фреймворк ZCM. В перспективах развития сервиса - добавление программного модуля контроля параметров кадра и анализа изображения с целью адаптивного изменения в режиме реального времени, что приведет к снижению шумов и уменьшению влияния цветовой слепоты камер при передвижении беспилотного электропоезда по тоннелям и городской среде в разное время дня и года.

ЛИТЕРАТУРА

1. J. Ruttan, The Architecture Of Open Source Applications, 2008, vol. II, ch. OSCAR
2. J. Galowicz, C++17 STL Cookbook, 2017, Packt, 532 p.

3. Allen I. Holub, Enough Rope to Shoot Yourself in the Foot: 40 Rules for C++ and C Programming (Unix/C S.), 1 Oct. 1995, McGraw-Hill Inc.,US, 208 p.
4. R. Shilkrot, D. Millan Escriva, Mastering OpenCV 4: A comprehensive guide to building computer vision and image processing applications with C++, 3rd Edition, December 27, 2018, Packt Publishing, 280 p.
5. URL: https://www.ximea.com/support/wiki/apis/XiAPI_Manual

УДК: 004.43:004.65

Маликов А. Т. (2 курс бакалавриата)
Филиповский В. М., к.т.н., доцент

РАЗРАБОТКА ПРОГРАММЫ ПОСТРОЕНИЯ ГЕНЕАЛОГИЧЕСКИХ ДЕРЕВЬЕВ НА ЯЗЫКЕ C++ В СРЕДЕ MICROSOFT VISUAL STUDIO

История рода и родословная, - эти понятия интересовали человечество с древних времен. Информация о предках и ближайших родственниках передавалась из поколения в поколение, хранилась в памяти потомков и изображалась в виде генеалогических деревьев. Чем бережнее предки относились к ведению и сохранению своей родословной, тем больше информации о родственниках, которую необходимо обрабатывать и сохранять, доходило до потомков.

Для хранения различной информации создаются так называемые базы данных, в основу которых могут положены различные модели информационных систем: иерархическая, сетевая, реляционная [0]. Наиболее удобная с точки зрения хранения информации и доступа к ней является реляционная модель, когда данные хранятся в отдельных таблицах, имеющих горизонтальные связи между собой. По этому принципу многие крупные компании, такие как IBM, Microsoft, Oracle, строят свои программные продукты хранилищ информации – СУБД. Для построения генеалогических деревьев разрабатываются соответствующие программы [0], в основе которых также лежат базы данных.

Объектом данной статьи является программа для создания генеалогических деревьев, основанная на принципе реляционной базы данных и позволяющая выполнять различные действия с данными в ней, включающие в себя как построение иерархического дерева родственных связей, так и выполнение основных функций любой базы [0]. Целью является разработка среды хранения родословной, включающей в себя личные данные о человеке, родителях, браках, образовании, наградах, а также биографические заметки, ссылки на интернет-ресурсы и фотографии, и наглядного отображения информации из неё. Разработка осуществлена на языке C++ в среде программирования Microsoft Visual Studio, являющимися пригодными и удобными не только для проектирования баз данных, но и для создания современных и удобных графических интерфейсов [0].

Для хранения данных использовались односвязный список и многомерное дерево. Односвязный список выступает в роли хранилища данных, реализующего реляционную схему данных. Помимо хранения данных, список обеспечивает их индексацию. Для работы со списком создан ряд методов, обеспечивающих добавление, поиск и обновление записей списка. Связь данных между списками обеспечивается благодаря указателям, которые вместе с индексами списков обеспечивают целостность данных. Наряду с односвязным списком, использовалась параллельная структура данных, представляющая собой многомерное дерево. Её назначением являлось иерархическое представление информации о родственниках в виде дерева. Для работы с деревом класс имеет ряд методов, реализующие добавление данных и поддеревьев, а также получение или обновление данных.

Создавать классы, различающиеся только типами данных – трудоёмкий и не практичный метод изготовления программного продукта. Для облегчения этого были разработаны шаблоны и шаблонные методы, представляющие собой класс или функцию, написанные для заранее не определенных типов данных [0]. Благодаря им, классы, создаваемые для разных

типов данных, наследуют все методы шаблонного класса, что значительно ускоряет процесс разработки. Шаблонами в данной программе представлены классы списка и дерева. Кроме шаблонных для каждого класса методов разработаны специальные функции, предназначенные для работы с объектами данного класса.

Благодаря использованию списков и возможностям языка C++, в программе становится возможным реализовывать любые функции, которые могут быть интересны и необходимы пользователю. Одной из таких функция является определение родственных связей между людьми. Для этого использовалась идея поиска кратчайшего пути. Существующие алгоритмы поиска пути ориентированы на графы, поэтому для их применения на основе данных базы строился граф родственных отношений. К этому графу применялся алгоритм Ли, являющийся одним из наилучших алгоритмов поиска кратчайшего пути между двумя точками графа, если это является возможным [0]. Используемая реализация алгоритма не рекурсивная, так как волна распространяется внутри цикла, что позволяет избежать возможного переполнения стека при большом размере графа.

Наличие графического интерфейса является неотъемлемой частью современных приложений. Среда программирования Visual Studio позволяет в полной мере использовать все возможности оконных интерфейсов, включая и кнопки, и ленточные меню, и контекстные меню для реализации большинства функций. Благодаря возможностям среды и элементам библиотеки Windows Forms можно реализовывать любые функции оконного приложения, а также функции баз данных, к которым относятся ввод, вывод информации, просмотр данных базы, создание отчётов [0] и защита информации. Так, например, отображение генеалогического дерева и родственных связей осуществлялось благодаря иконкам с краткой информацией о человеке (см. рис. 1). Расположение этих иконок соответствовало поколению человека – чем выше иконка, тем к более старшему поколению относится человек.

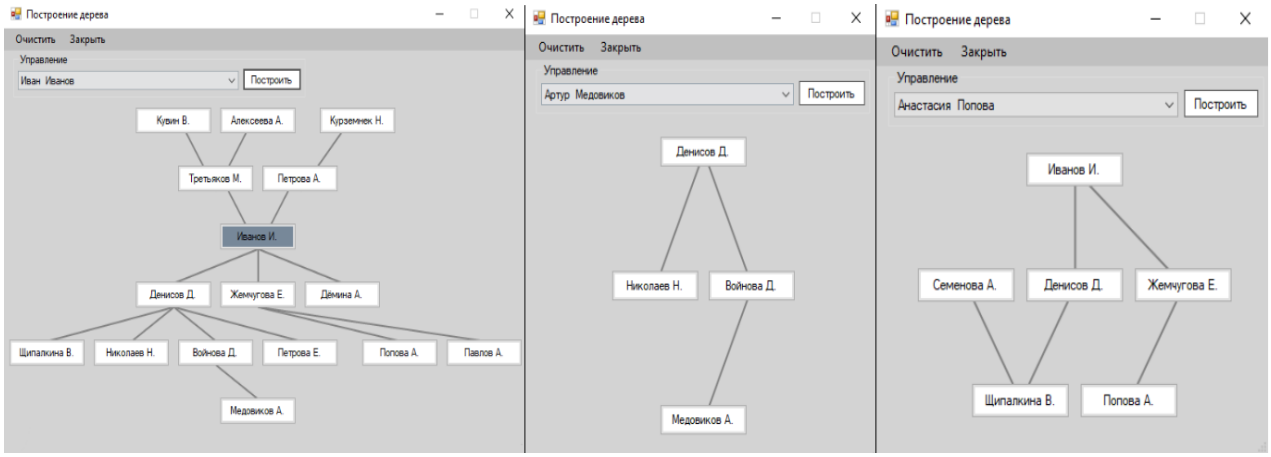


Рисунок 1 – Оконный интерфейс программы

Создание графического интерфейса является заключительным этапом разработки приложения. Благодаря особенностям языка C++, возможностям среды Microsoft Visual Studio и двум информационным моделям – иерархической и реляционной, была разработана программа, совмещающая в себе как возможности и преимущества реляционных баз данных, так и многие функции, присущие оконным приложениям, включая возможность редактирования и защиты информации, а также её отображение в явном и понятном виде, - построение генеалогического дерева или определение родственных связей.

ЛИТЕРАТУРА

1. Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильяме", 2005. — 1328 с.: ил. — Парал. тит. англ.
2. Программы для генеалогического дерева [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/110892/

3. Конюхов В.Г. База данных. Понятие, значение и роль в современном мире // Системные технологии. 2017. №24. Режим доступа: https://cyberleninka.ru/article/n/baza-dannyh-ponyatie-znachenie-i-rol-v-sovremennom-mire/
4. Худяков А.А., Филиповский В.М. Создание базы данных на языке C++ в среде Microsoft Visual Studio // Современные технологии в теории и практике программирования: сборник материалов конференции, 23 апреля 2020 г. – СПб. : ПОЛИТЕХПРЕСС, 2020. – с.78-80.
5. Вандевурд, Дэвид, Джосаттис, Николаи М., Грегор, Дуглас. B17 Шаблоны C++. Справочник разработчика, 2-е изд.: Пер. с англ. — СпБ.: ООО “Альфа-книга”, 2018. — 848 с.: ил. — Парал. гит. англ.
6. Изотова Т.Ю. Обзор алгоритмов поиска кратчайшего пути в графе // Новые информационные технологии в автоматизированных системах. 2016. №19. Режим доступа: https://cyberleninka.ru/article/n/obzor-algoritmov-poiska-kratchayshego-puti-v-grafe/

УДК 004.5

Небогатиков И.Ю. (2 курс магистратуры)
Соловьев И.П., к.ф.-м.н., доцент

СИСТЕМА КОНТРОЛЯ ПРОДОЛЖИТЕЛЬНОСТИ ДЕЙСТВИЙ ОБИТАТЕЛЯ В СРЕДЕ УМНОГО ДОМА

Целью данной работы является разработка и реализация системы мониторинга и контроля продолжительности активности обитателей умного дома с использованием носимых датчиков.

Интерфейс системы умного дома Google Smart Home позволяет задавать и программировать пользовательские сценарии, называемые действиями. Созданные действия доступны для вызова через интерфейс голосового управления Google Assistant [1].

Первый разработанный сценарий среды умного дома предназначен для изменения предела длительности активности с применением ассистента.

Пользователь отправляет запрос ассистенту на изменение параметров продолжительности активности. Ассистент вычисляет действие, которое необходимо запустить для обработки поступившего запроса и вычисляет требуемые аргументы.

Обработчик действия отправляет полученные параметры на сервер через интерфейс REST API. Сервер возвращает сообщение о подтверждении обновления информации.

Второй сценарий разработан для уведомления обитателей о превышении пределов длительности активности. Приложение GadgetBridge, предназначенное для замены приложений производителей фитнес-браслетов и унификации интерфейсов их взаимодействия, используется в среде умного дома для получения информации о сердцебиении и количестве шагов [2]. К полученным показаниям добавляются значения качества сигнала соединения (RSSI) Bluetooth с фитнес-браслетом и качества сигнала соединения смартфона с Wi-Fi роутером. Сформированный набор значений передается на сервер.

На сервере установлен классификатор, определяющий вид активности по указанным выше параметрам. Классификатор реализует модель иерархического распознавания, в которой базовый классификатор применяется для распознавания базовой характеристики действия (пассивное или активное) с использованием алгоритма случайного леса. Полученное значение уточняется двумя классификаторами, предназначенными для распознавания пассивных и активных видов действий, и применяющими методы K-ближайших соседей и случайного леса соответственно.

Длительность распознанного вида активности проверяется на превышение допустимого предела, в приложение возвращается тип действия и характер предупреждения. Пользователь получает уведомление о превышении установленной продолжительности действия.

Разработанная и реализованная система может применяться обитателями умного дома для ограничения длительности заданных видов активности для управления здоровым ритмом жизни.

ЛИТЕРАТУРА

1. Kirola M., Rawat A., Sharma N., Sinha A. K. (2019). Smart Home Automation Using Google Assistant. SSRN Electronic Journal.
2. Rincon M., Montoya J. (2019). Smartwatch-Based User Monitoring for Human-Robot Interaction.

УДК 004.04

Околович Л.Д., Власов Д.В., Иванов Г.В. (4 курс бакалавриата),
Никифоров И.В., к.т.н., доцент

РАЗРАБОТКА МОДУЛЯ ТРАНСКОДИРОВАНИЯ ВИДЕОПОТОКОВ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ FFMPEG

Современный тренд в области медиа пространства направлен на создание и использование систем для обработки и передачи мультимедийных данных [1]. Медиа платформа “ОККО” использует транскодер от сторонней компании, что влечет дополнительные расходы на покупку лицензии. Сейчас перед “ОККО” стоит задача разработать собственный транскодер. Это позволит снизить стоимость эксплуатации данного программного продукта и удовлетворить потребность в ускорении процесса транскодирования.

Система транскодирования должна обладать следующими качествами: масштабируемостью, высокой производительностью, надежностью и безопасностью. Горизонтальное масштабирование [2] позволит получить линейный прирост в скорости транскодирования. Аппаратное ускорения Intel Quick Sync Video позволит ускорить процесс транскодирования [3].

Таким образом, актуальной является задача разработки и использования собственного транскодера для оптимизации использования ресурсов и ускорения процесса транскодирования.

Целью работы является снижение затрат на эксплуатацию оборудования и повышение скорости процесса транскодирования, за счет модуля транскодирования видеопотоков в режиме реального времени с использованием библиотеки ffmpeg, который реализован в программном средстве и внедрен в процесс транскодирования с аппаратным ускорением Intel Quick Sync Video [4].

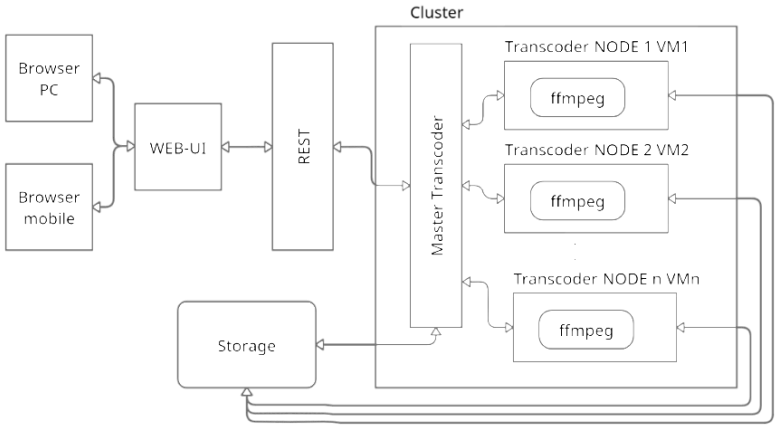


Рисунок 1 – Архитектура транскодера с использованием библиотеки ffmpeg

Предложенная архитектура позволяет осуществить высокопроизводительное преобразование видео аудио потоков с поддержкой горизонтального масштабирования.

Используемые технологии: Python 3.5, ffmpeg, Docker [5].

Сопутствующие средства: Bitbaket, JIRA, Confluence, IDE Pycharm Professional Edition.

Демонстрация повышения скорости и эффективности транскодирования представлена на рисунке 2.

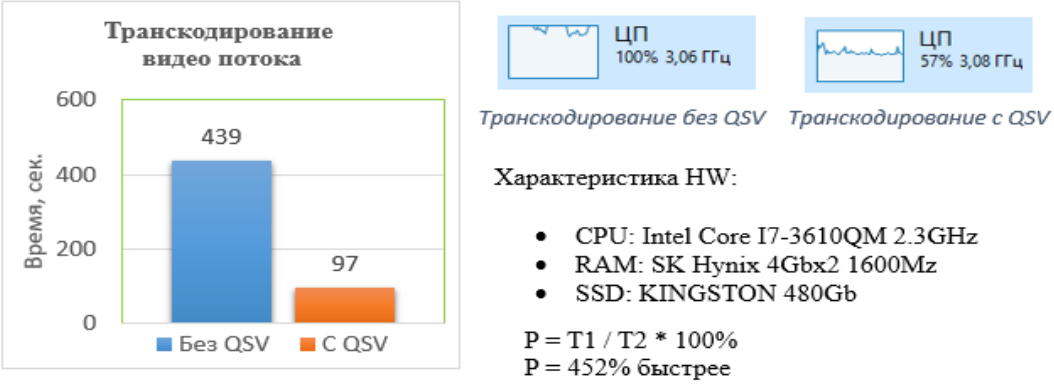


Рисунок 2 – Демонстрация эффективности предложенного решения

Таким образом, с помощью аппаратного ускорения Intel Quick Sync Video было достигнуто повышение скорости и эффективности транскодирования видео потоков.

Работа выполнена в рамках совместного проекта СПБПУ и компании “ОККО”.

ЛИТЕРАТУРА

1. Сычев В.К., Ленькова Ю.В., Цветкова Е.А., Никифоров И.В., анализ данных трендов youtube, Современные технологии в теории и практике программирования. сборник материалов конференции. Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. Санкт-Петербург, 2020. С. 131-132.
2. Птицына Л.К., Сулякаев Т.Р., Исследование методов и средств распределенных регистраций в информационных структурах, Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО 2020). Сборник научных статей IX Международной научно-технической и научно-методической конференции. В 4-х т. Санкт-Петербург, 2020. С. 547-550.
3. Intel quick sync video [Электронный ресурс]. Режим доступа: https://www.intel.ru/content/www/ru/ru/architecture-and-technology/quick-sync-video/quick-sync-video-general.html
4. FFmpeg [Электронный ресурс]. Режим доступа: https://ffmpeg.org/
5. Лубнин А. В., Ничипорчук А. В., анализ актуальности разработки программного обеспечения для предоставления услуг хостинга Docker контейнеров, Научный электронный журнал меридиан. Якутск, 2020. с. 63-65

УДК 004.414.32

Орлов В. В. (4 курс бакалавриата)
Коликова Т. В., ст. преподаватель

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ПОД ОПЕРАЦИОННУЮ СИСТЕМУ ANDROID ДЛЯ ПЛАНИРОВАНИЯ И ОРГАНИЗАЦИИ ПУТЕШЕСТВИЙ

Целью работы является разработка мобильного приложения для планирования и организации путешествий. Данное приложение должно оказаться решением проблемы, связанной с ошибками при планировании путешествий различного типа. В нем будет возможно составлять списки дел, организованным по группам, в чем пользователю будет помогать цифровой ассистент, предлагая необходимые пункты, в зависимости от места

назначения, длительности, типа и прочих факторов путешествия. Если вдруг суммарный вес вещей вдруг окажется больше допустимого веса багажа, приложение обязательно уведомит пользователя об этом и предложит, что можно выложить.

В приложении будет предусмотрена авторизация, для возможности миграции пользователя с одного устройства на другое. Для удобства использования приложения его дизайн будет приведен к стандартам Material Design, разработанным дизайнерами Google, основной идеей которых было то, что у приложений не должно быть острых углов, карточки должны переключаться между собой плавно и практически незаметно. Помимо этого, для удобства пользователей различной национальности, помимо русского языка, будет предусмотрена поддержка английского языка, являющегося интернациональным.

Удобной особенностью данного приложения будет наличие функции поиска отелей поблизости. Для этого будет использоваться GPS Tracker и Google Maps API. Для исключения чрезмерной нагрузки на девайс, локации отелей будут отображаться исключительно в заданном радиусе от пользователя, этот радиус можно будет менять в один клик в настройках карты. После выбора понравившегося отеля появится возможность проложить маршрут до него или перейти в приложение Google Maps и уже там отслеживать маршрут. Для удобства пользования данным функционалом будет разработан виджет на главный экран.

В качестве интегрированной среды разработки выступает Android Studio, разработанная компанией Google. Языком разработки был выбран Kotlin, так как с 2017 года Google официально поддерживает Kotlin как язык разработки под Android. Помимо этого, Kotlin – более молодой язык, чем Java, имеет множество приятных особенностей, а также стремительно развивается, смещая Java с позиции основного языка Android разработки. В качестве СУБД для хранения данных пользователя была выбрана Firebase от компании Google, так как она является СУБД класса NoSQL, изменяющейся в реальном времени, что позволяет разработчику хранить и синхронизировать данные между несколькими клиентами одновременно.

ЛИТЕРАТУРА

1. Филлипс Б., Стюарт К., Марсикано К. Android. Программирование для профессионалов. 3-е изд. – СПб.: Питер, 2017. – 688 с.: ил. – (Серия «Для профессионалов»)
2. Documentation for app developers. URL: https://developer.android.com/docs, дата последнего обращения: 17.03.2021
3. Kotlin documentation. URL: https://kotlinlang.org/docs, дата последнего обращения: 17.03.2021
4. Firebase documentation. URL: https://firebase.google.com/docs, дата последнего обращения: 17.03.2021

УДК 004.05

Неверовский Н. А. (4 курс бакалавриата)
Никифоров И. В., к.т.н., доцент

МЕТОД ПОСТРОЕНИЯ ОБОГАЩЕННОГО МЕТАДАННЫМИ ИЗОБРАЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ REST-МИКРОСЕРВИСОВ

Развитие всемирной сети Интернет ставит перед веб-сервисами и платформами все новые задачи для удовлетворения потребностей конечных пользователей. Растет мультимедийное наполнение сайтов и приложений и, для обработки сотен, а порой и тысяч пользовательских запросов в минуту, нужно создавать высокопроизводительные системы, позволяющие управлять работой с изображениями в такой среде [1]. Процесс необходимо оптимизировать на всех этапах, от работы дизайнеров по медиа-наполнению с держателями

платформ до введения новых технических решений в REST-архитектуре сервиса по обработке изображений [2]. Целью представленной работы, проведенной совместно с компанией «ОККО», является снижение трудоемкости управления медиаконтентом на онлайн-платформе.

Для достижения цели выполнение работы было поделено на два этапа, представленные ниже.

1. Исследование современных эффективных технологий разработки REST-микросервисов.
    a. Исследование языков программирования в промышленной разработке Backend-сервисов (Java, Python, Node.JS и т. п.).
    b. Выбор библиотек для синхронной/асинхронной/реактивной парадигмы программирования и обработки запросов через протокол REST.
    c. Выбор библиотек для работы с мультимедийными данными.
    d. Исследование используемых СУБД на рынке, оценка эффективности их работы в задачах по хранению изображений и метаданных, сохраненных в текстовой разметке.
2. Исследование систем кодирования метаданных для описания формальных правил обработки изображений, веб-инструментария для работы с метаданными [3].
    a. Исследование принципов работы онлайн-редакторов изображений, их способы хранить и обрабатывать метаданные об обработке изображения.
    b. Выбор фреймворка для Frontend веб-приложения.
    c. Изучение способов валидации и верификации метаданных.

Исследования, проведенные в работе, позволили обозначить набор технологий, используемых в реализации (Рисунок 1).

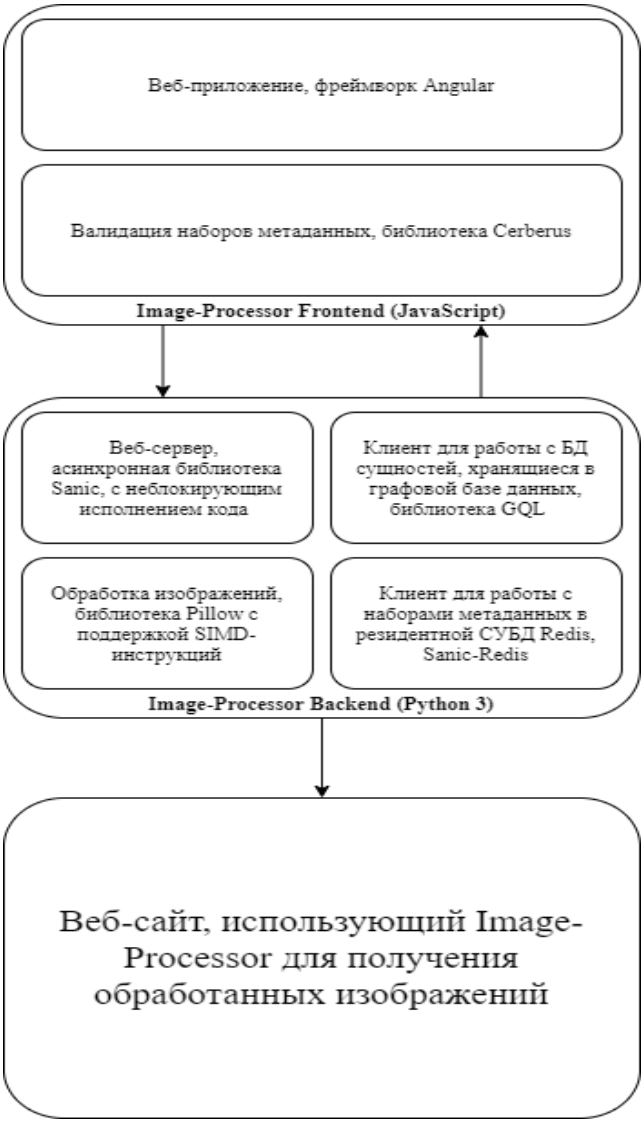


Рисунок 1 – Архитектура и используемые технологии веб-сервиса.

На Рисунке 2 представлен конвейер обработки изображений. Клиенты обращаются к веб-сервису с уникальными идентификаторами изображений в БД и уникальным идентификатором набора метаданных в системе. Данный набор, хранимый в кэше БД Redis [4], содержит список инструкций для обработки изображения. Изображение, вместе со всеми необходимыми сущностями для обработки, достается из графовой БД, в виде единой модели. Набор инструкций выполняется с использованием библиотеки обработки растровых изображений Pillow-SIMD [5]. Конечное изображение возвращается по запросу.

Frontend-часть предоставляет функционал администрирования наборов метаданных в веб-сервисе: добавлять, изменять, удалять наборы метаданных, создавать предпросмотры преобразований до их полноценного введения в систему.

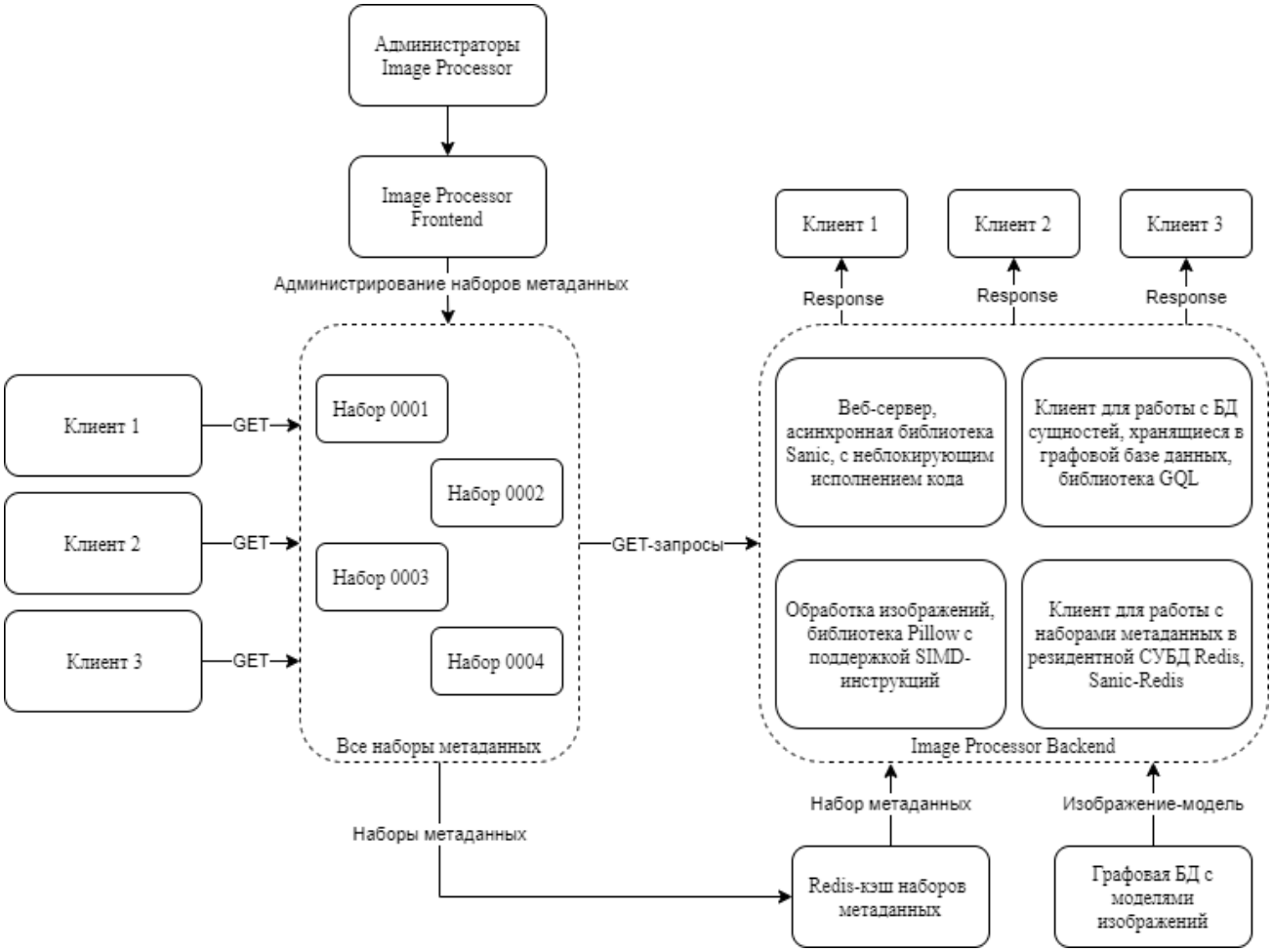


Рисунок 2 – Конвейер работы с изображениями и метаданными

Предложенная в работе архитектура реализует самые новые технологии и использует оптимизированные библиотеки в области построения веб-сервисов и позволяет создать высокопроизводительное решение готовое для применения в промышленных системах, рассчитанных на десятки тысяч клиентов.

ЛИТЕРАТУРА

1. Сычев В.К., Ленькова Ю.В., Цветкова Е.А., Никифоров И.В., Анализ данных трендов Youtube, Современные технологии в теории и практике программирования. сборник материалов конференции. Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. Санкт-Петербург, 2020. С. 131-132.
2. Виничук О.Н., Веб-приложение для обработки изображений больших объемов // Новости науки и технологий. 2020. С. 23-32.
3. Васильевский И. А., Алгоритмы веб-сервиса обработки сборок метаданных для CRM-системы с открытым исходным кодом // Молодой ученый. 2018. С. 125-127.
4. Рубин О.И., Использование СУБД Redis в качестве промежуточного хранилища данных для POSTGRESQL // STUDNET. 2020. С. 1646-1650.
5. Сараджишвили С.Э., Леонтьев В.В., Воинов Н.В., Введение в обработку изображений на языке Python, Санкт-Петербург, 2020.

УДК 004.42

Письмерова В.А. (2 курс магистратуры)
Черноруцкий И.Г., д.т.н., профессор

РАЗРАБОТКА СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЯ ПО ТОРГОВЛЕ ЦЕННЫМИ БУМАГАМИ

Торговля ценными бумагами в последнее время набирает популярность у физических лиц, что подтверждается данными с сайта Мосбиржи [1]. Среди людей, торгующих ценными бумагами, есть как финансовые аналитики, так и люди без экономического образования [2].

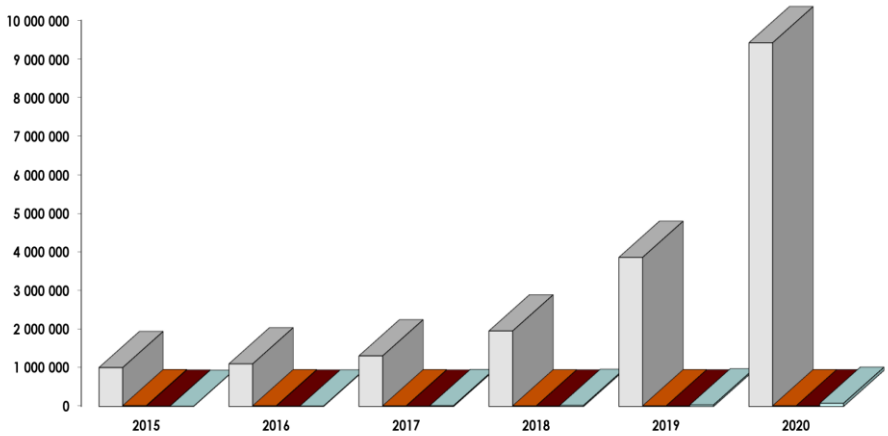


Рисунок 1 – Количество уникальных клиентов в Системе торгов (по состоянию на последний день месяца) (ежегодная статистика). Наблюдается увеличение числа физических лиц с 4 млн до 9,5 млн за последний год. Данные взяты с сайта Мосбиржи.

Результат работы аналитиков сложно актуализировать и пересматривать. Нередко происходит так, что анализ был сделан на основании одних данных, а через несколько дней финансовая ситуация изменилась и аналитические результаты потребовали корректировки (свежий отчет самой компании, сильно отличающийся от ожиданий; политические события, корпоративные изменения и т.д.).

Цель рекомендательной системы – по данным наблюдений предсказать будущие значения измеряемых характеристик изучаемого объекта и дать пользователю системы актуальный отчет с рекомендациями относительно возможной покупки или продажи финансового инструмента. Продуктом системы является отчет по ряду акций с рекомендацией к покупке или продаже.

Существует большое количество исследований на тему прогнозирования котировок акций. Как правило, авторы статей рассматривают различные модели машинного обучения и проводят исследования на самых популярных акциях [3] [4] [5]. В статьях отсутствуют описания применения моделей, не рассматривается рынок российских акций. Проблема часто рассматривается с одной стороны и не учитывается множество факторов.

Возможно выделить несколько задач, которые встают перед исследователем в области прогнозирования котировок финансовых инструментов:

- Прогнозирование цены по историческим данным о ценах.
- Прогнозирование тренда по новостям.
- Прогнозирование тренда по настроению инвесторов.

Для учета разных факторов недостаточно построить одну модель, для разных моделей нужны различные данные, и, соответственно, различные загрузчики и обработчики. Построенные модели могут иметь разную точность предсказаний и могут выгодно дополнять друг друга.

Система спроектирована с использованием микросервисной архитектуры: каждое приложение отвечает за свою часть работы в формировании результата, система может быть дополнена новыми источниками данных и новыми обработчиками. Различные приложения выполняют загрузку сырых данных в БД, прогнозирование с помощью ML-алгоритмов и сохранение результатов прогнозирования; агрегацию прогнозов, полученных из разных источников и составление отчета для потребителя.

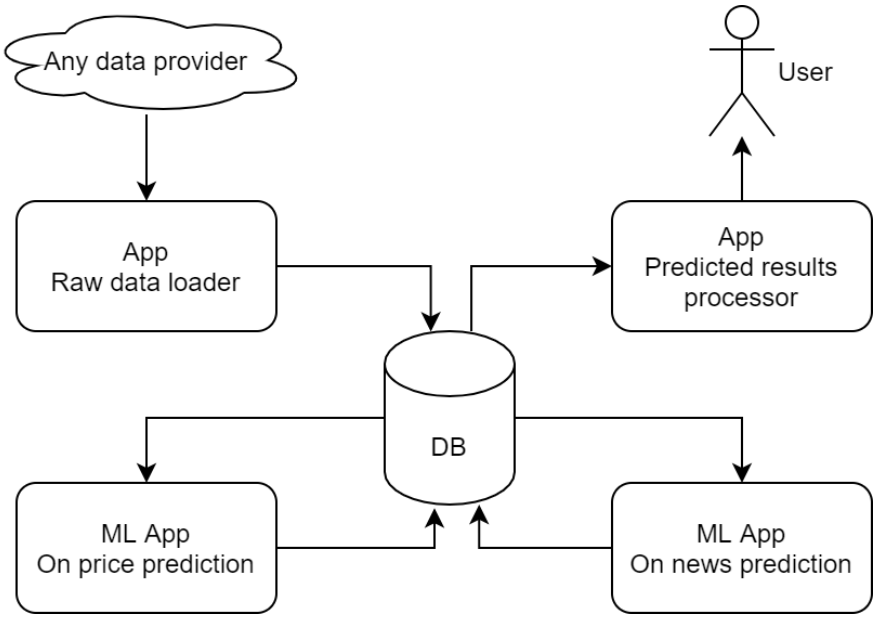


Рисунок 2 – Архитектура системы.

Приложения для загрузки сырых данных написаны на языке Java с использованием фреймворка Spring. Приложения, реализующие модели машинного обучения, разработаны на python использованием библиотек Keras и TensorFlow. В качестве базы данных выбран PostgreSQL.

Данные по расписанию загружаются в базу данных. Загруженные сырые данные обрабатываются ML моделями. Результаты предсказаний выгружаются в БД. После того как все предсказания получены, выгружается отчет в формате csv о возможных тенденциях на рынке в разрезе ценных бумаг.

Преимуществами такого подхода является возможность комбинировать различные методы прогнозирования, включать и исключать блоки в зависимости от их эффективности.

В будущем планируется добавить пользовательский интерфейс для удобства управления системой и визуализации отчетов.

ЛИТЕРАТУРА

1. Статистика по клиентам // Московская биржа. URL: https://www.moex.com/s719 (дата обращения: 14.03.2021).
2. Санкт-Петербургская биржа впервые обогнала Мосбиржу по торгам акциями // Вести. URL: https://www.vesti.ru/finance/article/2519937 (дата обращения: 14.03.2021).
3. G. Bathla, "Stock Price prediction using LSTM and SVR," 2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC), Waknaghat, India, 2020, pp. 211-214, doi: 10.1109/PDGC50313.2020.9315800.
4. B. He and E. Kita, "Stock Price Prediction by Using Hybrid Sequential Generative Adversarial Networks," 2020 International Conference on Data Mining Workshops (ICDMW), Sorrento, Italy, 2020, pp. 341-347, doi: 10.1109/ICDMW51313.2020.00054.
5. Y. Ji, A. W. -C. Liew and L. Yang, "A Novel Improved Particle Swarm Optimization With Long-Short Term Memory Hybrid Model for Stock Indices Forecast," in IEEE Access, vol. 9, pp. 23660-23671, 2021, doi: 10.1109/ACCESS.2021.3056713.

УДК 004.925

Ромицына М.С. (1 курс магистратуры)
Петров О.Н., к. т. н., доцент

ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ЗАДАЧАХ КЛАССИФИКАЦИИ СИГНАЛОВ

Целью работы является разработка нейросети, предназначенной для классификации объектов (пловец/шум). Разработка ведётся с помощью фреймворка DeepLearningToolbox (Matlab).

Магнитоакустическая станция, установленная на дне, излучает сигнал. Получив отражённый сигнал, она формирует набор точек и передаёт их на обработку в систему, которая формирует наборы данных в виде матриц 32x32. Данная матрица содержит значения от 0 до 32, представляющих собой градацию интенсивности сигнала. Верхняя граница (32) влияет на размер матрицы плотности. Данное значение определяется настройками магнитоакустической станции.

По данным матрицам система формирует трассы и присваивает им номера. Задача сети – по матрице из трассы определить класс – пловец или шум.

На вход сеть получает матрицу плотности. На рисунках 1 и 2 представлены изображения сигналов (Рисунок 1 – пловец, Рисунок 2 – шум) на средней дистанции (300-550 метров).

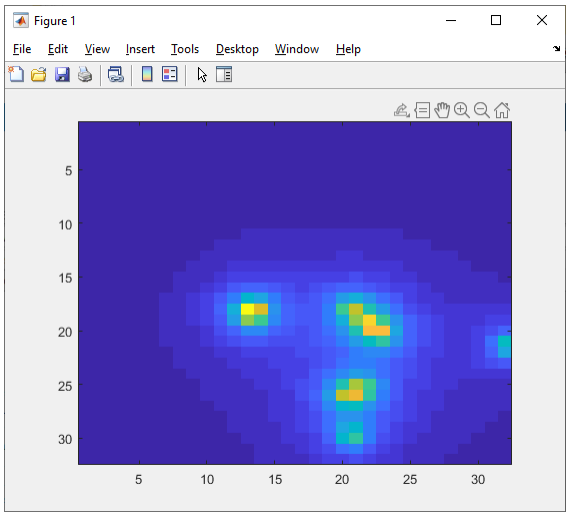


Рисунок 1 – Сигнал пловца на средней дистанции

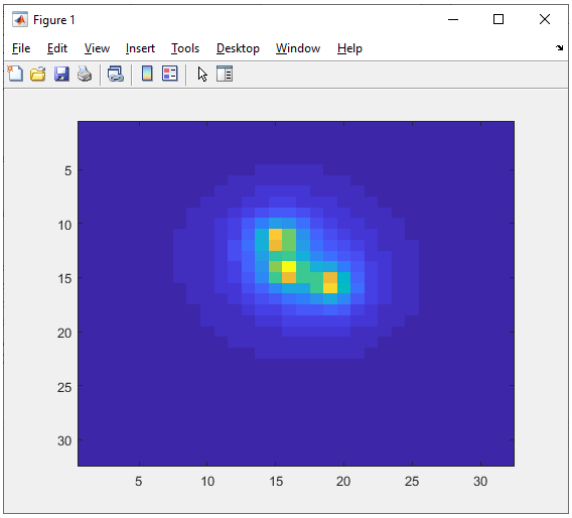


Рисунок 2 – Сигнал шума на средней дистанции

Формируется 3 набора данных:

1) обучающий: содержит матрицы сигналов и шумов в соотношении 50/50; матрицы принадлежат различным трассам;
2) тестовый: содержит матрицы сигналов и шумов в соотношении 50/50; матрицы берутся из тех же трасс, что и для обучения; сеть на этих матрицах плотности не обучалась;
3) контрольный: содержит матрицы сигналов и шумов в соотношении 50/50; матрицы берутся из трасс, не участвовавших в процессах обучения и тестирования; сеть на этих матрицах плотности не обучалась;

Обучение проводится на разных типах дистанций. Для каждого типа выбирается сеть с оптимальными параметрами. Типы дистанций:

1) короткая: [50;300) м;
2) средняя: [300;550) м;
3) дальняя: [550;900) м;

Также проводится обучение на смешанных данных, т.е. нет разделения на дистанции.

В обучении участвует несколько видов сетей:

- многослойный персептрон;
- свёрточная сеть;
- рекуррентная сеть;
- сочетание свёрточной и рекуррентной сетей.

После обучения и тестирования всех сетей производится выбор оптимальных сетей, которые будут переданы в систему обработки данных от магнитоакустической станции.

ЛИТЕРАТУРА

1. Татузов, А. Л. Нейронные сети в задачах радиолокации. Москва : М.: Радиотехника, 2009. стр. 432.
2. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. Погружение в мир нейронных сетей. Санкт-Петербург : ПИТЕР, 2019. стр. 480.
3. Траск, Эндрю. Грокаем глубокое обучение. Санкт-Петербург : ПИТЕР, 2019. стр. 352.
4. Рашид, Тарик. Создаём нейронную сеть. Санкт-Петербург : ООО "Альфа-книга", 2017. стр. 272.
5. DEEP LEARNING TOOLBOX. exponenta.ru. [В Интернете] https://exponenta.ru/neural-network-toolbox.

УДК 004.89

Полетова Н.В. (2 курс магистратуры)
Дробинцев Д. Ф., ст. преподаватель

РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА БЛЮД НА ОСНОВЕ ПИЩЕВОГО КОНТЕКСТА

Актуальность темы обусловлена необходимостью обеспечения удобства людям с ограничениями в питании, так как на данный момент подавляющее большинство рецептов блюд в интернете и в ресторанах не учитывают возможные ограничения людей. Решить эту проблему можно созданием удобного инструмента, позволяющего пользователю отредактировать рецепт блюда в соответствии с медицинскими показаниями или этическими ограничениями, для приготовления или заказа того блюда, что будет являться безопасным для здоровья и соответствовать этическим соображениям.

Целью данной работы является создание и изучение эффективности рекомендательной системы блюд на основе уже существующих, с необходимой для пользователя заменой продуктов на их аналоги, которые не будут существенно влиять на вкусовые качества блюда, но позволят:

1. Вписаться в заданные параметры питательных веществ, исчисляемых в количестве калорий, белков, жиров и углеводов.

2. Заменить продукты, которые пользователь не может употреблять по медицинским показаниям или этическим соображениям.

Для достижения данной цели были поставлены следующие задачи:

1. Сбор и подготовка рецептов различных блюд
2. Подбор подходящих для данной задачи алгоритмов машинного обучения и исследование их эффективности в подборе аналогов продуктов
3. Разработка ПО, позволяющего пользователю:
   - Выбрать в существующих рецептах продукты которые нужно исключить
   - Задать оптимальную калорийность и соотношение белков жиров и углеводов
   - Добавить новый рецепт
   - Увидеть результат работы программы в удобном формате
4. Опрос пользователей с целью подсчета количества удачных или неудачных подборов продуктов

Необходимые данные, представляющие из себя рецепты и таблицы калорийности продуктов были взяты из источников [1] и [2] и преобразованы в единый формат.

Далее были рассмотрены различные способы предсказания наиболее подходящего продукта для замены исключенного.

Одним из перспективных вариантов является нейросеть формата word2vec[3], предназначенная для обучения на тексте. Она использует представление слов в виде векторов в многомерной системе координат, где каждая координата нормализована и представляет из себя какую-либо характеристику слова. Чем больше схожи между собой два таких вектора, тем больше вероятность того, что два соответствующих слова может находиться рядом.

Обучение проходило на рецептах, которые состоят из списка продуктов, в результате были получены вектора слов-продуктов. На рисунке 1 результат обучения, отображенный на двумерное пространство.

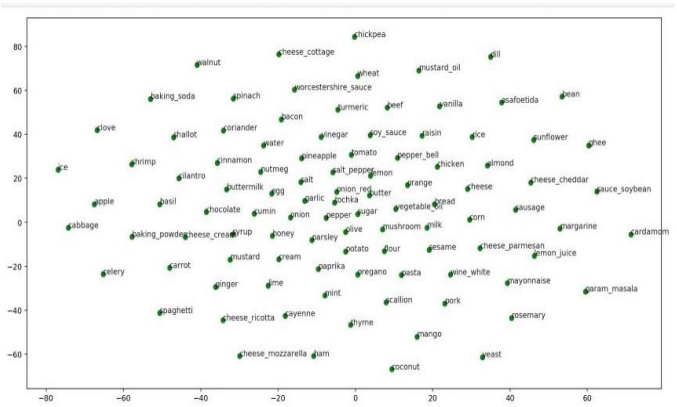


Рисунок 1 – Проекция пространства векторов на двумерное пространство.

В качестве алгоритмов для нейронной сети использовались Skip-gram и Cbow[4].

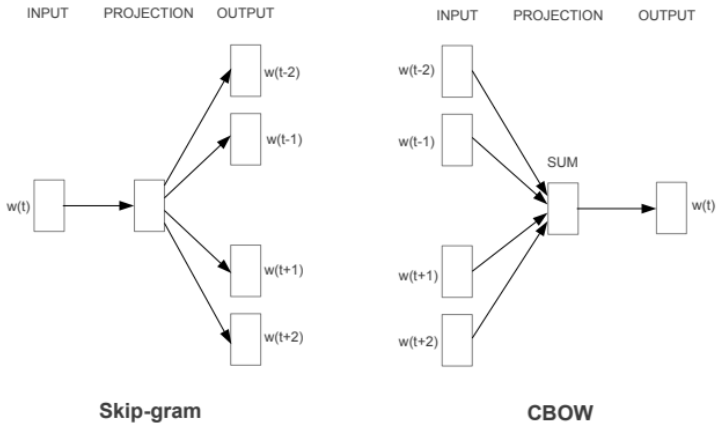


Рисунок 2 – Схемы алгоритмов Skip-gram и Cbow.

Алгоритм Skip-gram позволяет находить слова близкие по контексту, что позволяет подобрать несколько близких по сочетаемости продуктов. Задача алгоритма Cbow состоит в том, чтобы найти продукт по заданному набору продуктов. Совмещение этих двух алгоритмов позволяет подобрать сразу несколько возможных альтернатив для заменяемого продукта в рецепте.

Для того, чтобы организовать удобный доступ к нейросети необходимо реализовать графический интерфейс и сервис для обработки данных поступающих от пользователя.

В результате была подобрана архитектура для реализации которой используются следующие технологии:

- UI - Typescript+React+Material UI
- Data service and Neural network - Python
- Database - Postgresql

В результате пользователь сможет преобразовать любой понравившийся ему рецепт под свои ограничения.

Для проверки была выбрана следующая стратегия: нескольким людям предлагалось попробовать изменить под себя 2-3 рецепта и оценить результат положительно или отрицательно. Всего было проверено около 50 рецептов. Процент результатов, оцененных положительно составил 78%.

ЛИТЕРАТУРА

1. Yummly recipe API, Web: https://developer.yummly.com/
2. Complex systems lab website, CulinaryDB database, Web: https://cosylab.iiitd.edu.in/culinarydb/.
3. Github.io, The Illustrated Word2vec by Jay Alammar, Web: https://jalammar.github.io/illustrated-word2vec/
4. Tomas Mikolov, Kai Chen, G. S. Corrado, J. Dean “Efficient Estimation of Word Representations in Vector Space”, ICLR Workshop, 2013a

УДК 004.052

Скоков Н.С. (3 курс бакалавриата)
Пшеничная К.В., к.т.н., доцент

ВЫБОР СРЕДСТВ НЕЙТРАЛИЗАЦИИ АТАК ВИДА «ОТКАЗ В ОБСЛУЖИВАНИИ»

Согласно отчетам компании «Лаборатория Касперского» количество атак вида «Отказ в обслуживании» (DOS, DDOS) за последний год увеличилось в полтора раза и продолжает расти. Все DDOS атаки подразделяются на следующие группы: объемные атаки, атаки на уровне протоколов, атаки на уровне приложений, другие менее популярные атаки. Существует различные методы нейтрализации перечисленных видов атак, а также средства защиты, которые производятся крупными фирмами в области ИТ.

Целью настоящей работы является разработка приложения на языке C# для выбора средств защиты от атак вида «отказ в обслуживании». Данная проблема рассматривалась в работах [1, 2], где использовались методы теории игр. Однако, возможно применение и других подходов к решению задачи, например, использование метода анализа иерархий Томаса Саати [3]. Метод основан на парных сравнениях вариантов и позволяет пользователю в интерактивном режиме находить решение, соответствующее его пониманию проблемы. Данный метод и был использован при разработке приложения.

Постановка задачи включает рассмотрение четырех групп DOS/DDOS атак: “Group 1” (ICMP-flood, MAC-flood, UDP-flood), “Group 2” (SYN-flood, SSL-flood), “Group 3” (Slowloris, HTTP-flood), “Group 4” (other); и четырех вариантов средств защиты: StormWall (ПО), Selectel DDOS Protection, Kaskpersky DDOS Protection, DDOS Guard.

Критерии сравнения средств защиты: эффективность защиты против каждой атаки, цена, пропускная способность (объем анализируемого входного потока), производительность (время оценки ситуации), нагрузка на центральный процессор системы (ЦП). Дерево иерархии для выбора средства защиты по методу Саати представлено на рис.1.

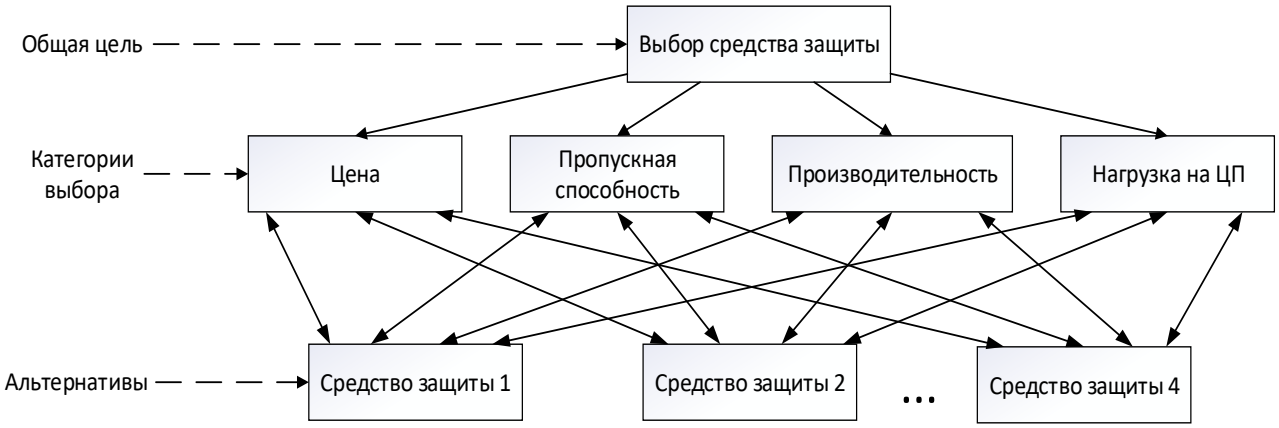


Рисунок 1 – Структурная модель системы выбора оптимального средства защиты.

Средства защиты по перечисленным критериям оцениваются по информации из опубликованных источников (табл.1).

Таблица 1 – Оценка критериальных параметров.

Средства защиты	Цена, р./ мес.	Пропускная способность, Мбит/сек.	Производительность, мин./цикл	Нагрузка на ЦП, %
StormWall (ПО)	4000	10	18	25
Selectel DDOS Protection	2550	10	20	40
Kaskpersky DDOS Protection	5000	15	15	55
DDOS Guard	1250	8	30	22

Основные этапы работы приложения на языке C#.

1) Пользователь задает матрицу значений коэффициентов эффективности средств защиты против каждой атаки путем заполнения соответствующей формы.

2) Пользователь анализирует значимость вредоносных атак, для чего ему предлагается специальный интерфейс.

3) Определяются веса значимости каждого типа атаки. Для этого пользователем выполняется попарное сравнения различных типов атак по некоторым заранее проанализированным параметрам путем заполнения нескольких матриц, (например, для анализа актуальности вектора нападения, анализа разрушительной мощности и т.п.). Столбцы и строки каждой матрицы соответствуют типам атак. Ячейки матрицы заполняются лицом, принимающим решение, следующим образом:

– главная диагональ заполняется единицами, так как сравнение происходит между одним и тем же видом нападения (типом атак);

– остальные ячейки матрицы заполняются значениями в диапазоне от 1 до 9 в соответствии с рекомендациями Саати.

Вес вектора атаки определяется средним значением по строкам матрицы.

4) В результате, по полученным весам исследуемых групп атак и эффективностям средств защиты определяется предпочтительный вариант ПО, который в процентном отношении по своими характеристиками лучше других.

Вычислительная часть способа реализуется посредством разработанного программного продукта. Результаты расчетов представлены в табл. 2.

Таблица 2 – Результаты исследований.

№ п/п	Наименование средства защиты	Оценка средств защиты, %
1	StormWall	25,11
2	Sel. DDOS Protection	24,98
3	Kaspersky DDOS Protection	25,03
4	DDOS Guard	24,88

Необходимо отметить, что результаты во многом зависят от тех числовых коэффициентов, которые пользователь вводил при парных сравнениях вариантов.

Таким образом, результаты исследования помогают обеспечить рациональный выбор средств защиты против интенсивно развивающихся компьютерных атак типа «отказ в обслуживании» (DOS/DDOS), а также анализировать некоторые уязвимости и недостатки продуктов защиты и помочь пользователям интернета в соответствии с их запросами выбирать наиболее подходящие условия безопасности.

ЛИТЕРАТУРА

1. Абденов А.Ж., Заркумова Р.Н. Выбор средств эффективной защиты с помощью теории игр // Вопросы защиты информации. – 2010. – №2. С. 26-31.
2. Т.В. Вахний, А.К. Гуц, В.В. Константинов. Программное приложение для выбора оптимального набора средств защиты компьютерной информации на основе теории игр // Вестн. Ом. ун-та. – 2013. – № 4. С. 201–206.
3. Саати Т. Л. Принятие решений. Метод анализа иерархий. — М.: Радио и связь, 1989. – 316 с.

УДК 004.045

Стариков А.О., Хмыров И.В. (4 курс бакалавриата)
Маслаков А.П., ст. преподаватель

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ТЕСТИРОВАНИЯ ТОРГОВЫХ СТРАТЕГИЙ НА АРХИВНЫХ ДАННЫХ

На сегодняшний день такое направление инвестирования, как торговля на рынке ценных бумаг и валют набирает популярность. Растет количество компаний, которые предлагают свои услуги для совершения сделок на этом рынке, но без возможности автоматизации осмысленной и алгоритмической торговли, выраженной в виде торговых стратегий.

Целью работы является создание веб-приложения, которое позволит тестировать торговые стратегии на исторических данных для их дальнейшего использования на реальном рынке ценных бумаг.

Основной проблемой при разработке торговых стратегий является невозможность быстрой проверки своих гипотез, предположений или торговых алгоритмов на рынке в режиме реального времени [2]. Для решения этой задачи разработанное веб-приложение загружает архивные данные настоящих торгов с биржи ценных бумаг, которые затем обрабатываются и подготавливаются для передачи в модуль тестирования. Модуль тестирования запускает симуляцию реального рынка, используя загруженные данные и торгует на нем, согласно написанной стратегии. В итоге пользователь получает структурированный результат торговли по его стратегии, который может оценить, сравнить с результатами других стратегий и принять решение о выведении данной стратегии на реальную рыночную торговлю.

Приложение разрабатывалось на основе клиент-серверной архитектуры [4]. На рисунке 1 приведено схематическое изображение модулей данного программного продукта. Он разделен на 4 логические части, однако стоит отметить, что 3 из них – Backtrader, Tinkoff API и Django относятся к серверной части.

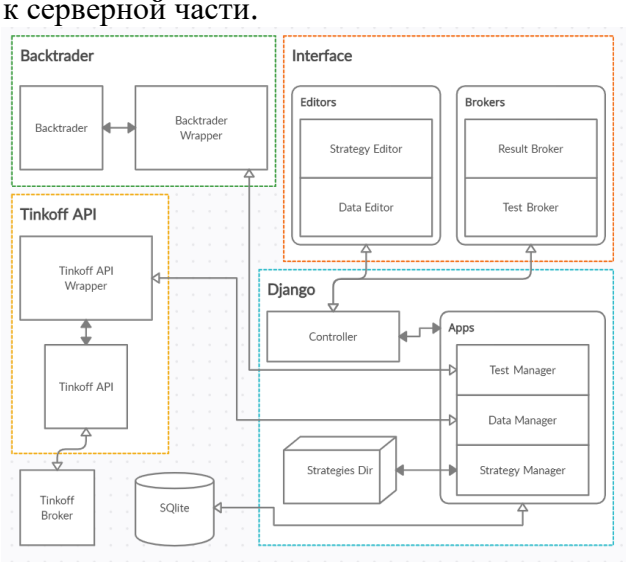


Рисунок 1 – Архитектура приложения

Серверная часть нашего приложения реализована на языке высокого уровня Python 3.8.7 [5] с использованием популярного фреймворка Django вместе с дополнительным фреймворком для построения RESTful API - Django Rest Framework [3]. В качестве системы хранения была выбрана база данных SQlite, взаимодействие с которой происходит напрямую из Django с помощью Django ORM.

Архитектура модуля Django состоит из трех Django apps:

- Strategy Manager - отвечает за работу с данными стратегий, их файлами и валидацией при загрузке в систему;
- Data Manager - отвечает за работу с архивными данными - занимается их получением и обработкой;
- Test Manager - отвечает за работу с тестированием и за распределение задач для модуля тестирования;

Архивные данные торгов приложение получает посредством запросов к асинхронным функция модуля, взаимодействующему с Tinkoff API. Tinkoff API предоставляет необходимые данные о торгах и ценных бумагах напрямую с Московской биржи, таким образом мы получаем актуальную информацию о торговых инструментах и достоверные исторические данные. Данные поступаю в базу данных SQlite, откуда в свою очередь передаются, после дополнительной обработки, в модуль тестирования, использующий асинхронные функции фреймворка Backtrader.

Клиентская часть приложения реализована в виде веб-интерфейса, созданного при помощи языка разметки HTML 5.3, CSS3, языка JavaScript [1] и фреймворка React. Также для упрощения процесса создания GUI были использованы библиотеки Bootstrap v4.0 и jQuery 3.6.0.

Архитектура фронтенда состоит из четырех больших классов, которые отвечают за соответствующие части приложения:

- Strategy Editor - отвечает за создание, обновление, удаление и в целом за менеджмент стратегий;
- Data Editor - отвечает за загрузку и удаление архивных данных;
- Test Broker - отвечает за проведение тестирования (параллелизация процесса тестирования, обновление и визуализация данных по ходу тестирования);
- Result Broker - отвечает за вывод актуальных и архивных результатов тестирования, а также их скачивание и удаление.

Фронтенд и бэкенд приложения обмениваются данными и командами по протоколу HTTP при помощи RESTful API. Все запросы с фронтенда поступают на конечные точки серверной части и передают данные в формате JSON, который также используется в ответах с сервера. Этот формат наиболее удобен для обеих технологий - языка JavaScript на фронтенде и языка Python на бэкенде, так как в них есть соответствующие структуры данных аналогичные формату JSON.

Весь функционал программного продукта был реализован и протестирован в работе. Основная цель получения различных результатов тестирования на исторических данных была достигнута. У пользователей появилась возможность наглядного сравнения свои стратегий после тестирования на различных торговых инструментах и периодах.

Инструменты, используемые в работе, позволяют реализовать дополнительные модули, которые предоставят пользователю возможность запуска протестированных стратегий в реальную торговлю из нашего веб-приложения. Это то направление, в котором планируется продолжить работу для получения полноценной системы осмысленного и алгоритмического инвестирования своих средств.

ЛИТЕРАТУРА

1. Васильев, А.Н. JavaScript в примерах и задачах/ А.Н.Васильев. – М.: Эксмо, 2017.
2. Чарльз ЛеБо, Дэвид Лукас Компьютерный анализ фьючерсных рынков / 3-е изд. – Альпина Бизнес Букс, 2008 - 25 с.
3. Habr// [научные статьи]/ — URL: https://habr.com/ru/post/508100// (дата обращения:12.12.2021).
4. Paul Clements; Felix Bachmann; Len Bass; David Garlan; James Ivers; Reed Little; Paulo Merson; Robert Nord; Judith Stafford. Documenting Software Architectures: Views and Beyond. — Second Edition. — Addison-Wesley Professional, 2010.
5. Python3// [документация]/ — URL: https://docs.python.org/3/ (дата обращения:10.11.2021).

УДК 004.42

Суслов К. А., Шаповал П. В. (4 курс бакалавриата)
Коликова Т. В., ст. преподаватель

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ СТРАХОВАНИЯ ТРАНСПОРТНЫХ СРЕДСТВ ПОД ANDROID ОС И IOS

Введение. Несмотря на то, что каждый год ГИБДД обнадеживают нас снижающимися цифрами в статистике дорожно-транспортных происшествий, по данным на прошлый 2020 год было официально зафиксировано 145073 аварий[1]. К сожалению, никто не может гарантировать вашу абсолютную безопасность на дороге, но можно хотя бы минимизировать финансовые издержки при случае ДТП путем страхования своего транспортного средства.

Область применения. На данный момент почти каждый третий россиянин является владельцем личного автомобиля (на 1 тысячу жителей приходится примерно 309 транспортных средств по состоянию на июль 2020 года[2]) и сохраняется тенденция на увеличение автомобилистов.

Сейчас основным вариантом страхования транспортного средства является каско, но заключить такой договор меньше, чем на год будет сложно и невыгодно.

Разрабатываемое приложение предлагает возможность страхования транспортного средства на выбранный промежуток времени или на определенное количество километров. А также обеспечивает справедливое назначение цены за услуги, путем анализа данных, полученных устройством непосредственно во время вождения.

Android приложение

Нативное Android приложение

Язык программирования – Kotlin 1.3.61

Поддерживаемые версии – начиная с API 21/Android 5.0 (оптимальными являются API 26 - 30/ Android 8.0 - 11).

Приложение написано на базе архитектурных паттернов Single Activity и MVVM (Model-View-ViewModel).

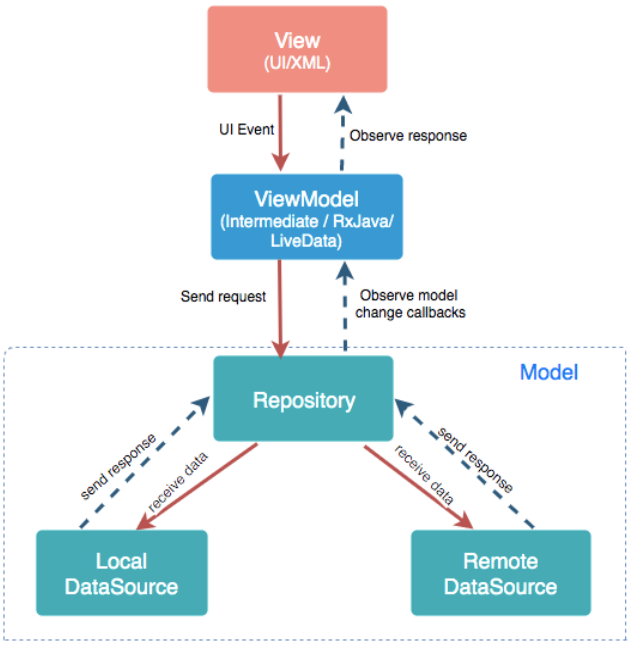


Рисунок 1 – Схема архитектуры Android приложения[3]

iOS приложение

Нативное приложение.

ЯП – Swift 5.0

Поддерживаемые версии iOS – начиная с 12.2.

В приложении используется циклическая архитектура Clean Swift, которая развивает идеи VIPER и построена на принципе разделения обязанностей.

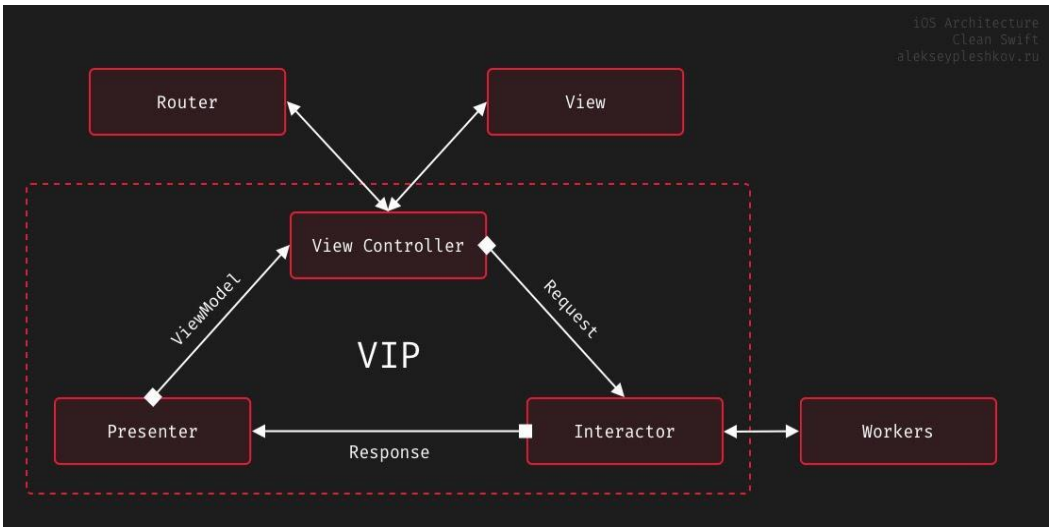


Рисунок 2 – Схема архитектуры iOS приложения[4]

Основные функции. В мобильном приложении есть возможность оформить один из трёх тарифов страхования:

- “Парковка” - оформляется на определенный срок. Указывается тип и место парковки. Местом парковки по умолчанию является местоположение пользователя. Также место парковки можно указать вручную.
- “Поездка” - пользователь указывает точки планируемого маршрута. Отправной точкой по умолчанию является местоположение пользователя. Есть возможность как указания точек маршрута на карте, так и с помощью поиска адреса. При оформленном тарифа данного тарифа осуществляется мониторинг прибытия пользователя в конечную точку маршрута.
- “Пакет километров” - предусматривает страхование ТС на определённый пробег километров.

Пользователь в любой момент может остановить действие тарифа, не дожидаясь его окончания.

При покупке тарифов, подразумевающих передвижение (“поездка” и “пакет километров”) производится сбор данных с сенсоров смартфона. Мобильное приложение собирает данные с акселерометра, гироскопа и географические координаты. Все эти данные записываются в локальную базу данных и отправляются на бэкэнд с некоторой периодичностью.

Для “поездки” сбор данных начинается сразу после покупки страхового полиса и производится до тех пор, пока пользователь принудительно не отменит полис или до момента истечение срока его действия.

Для “пакета километров” сбор данных начинается автоматически – либо при обнаружении bluetooth-устройства, которое “привязано” к ТС, либо при определении факта передвижения пользователя на автомобиле.

Заключение. Сейчас приложения находятся на этапе бета-тестирования и интеграции со страховой компанией, но уже доступны для бесплатного скачивания в магазинах приложений:

- https://play.google.com/store/apps/details?id=ru.flexydrive.app
- https://apps.apple.com/ru/app/flexy-drive/id1536753562

ЛИТЕРАТУРА

1. Официальный сайт Госавтоинспекции [Электронный ресурс] URL: http://stat.gibdd.ru/ (дата обращения 15.03.2021 г.)
2. Официальный сайт Автостат. Аналитическое агентство. [Электронный ресурс] URL: https://www.autostat.ru/press-releases/46332/ (дата обращения 15.03.2021 г.)
3. Android Developers. Guide to app architecture [Электронный ресурс] URL: https://developer.android.com/jetpack/guide (дата обращения 15.03.2021 г.)
4. Clean Swift. Clean Architecture [Электронный ресурс] URL: https://clean-swift.com/ (дата обращения 15.03.2021 г.)

УДК 519.688

Сясько М.В. (2 курс магистратуры)
Соловьев И. П., к. ф.-м.н., доцент

РАСЧЁТ И ВИЗУАЛИЗАЦИЯ ТРАЕКТОРИЙ ДВИЖЕНИЯ ЗАРЯЖЕННЫХ ЧАСТИЦ В НИЗКОЧАСТОТНЫХ ЭЛЕКТРОМАГНИТНЫХ ПОЛЯХ

Компьютерное моделирование и визуализация низкочастотных электромагнитных полей имеют большое значение в образовании, научно-исследовательской деятельности, медицинской практике и промышленности. Процесс визуализации дает представление о природе влияния магнитных полей на различные физические среды, выявляет особенности возникающих явлений и во многих случаях помогает совершенствовать используемые физико-математические модели.

В данной работе представлено гибкое программное решение задачи расчёта и визуализации траекторий движения заряженных частиц в низкочастотных электромагнитных полях, сочетающий в себе как все расчеты, так и визуализацию в одной среде. Разработан и реализован усовершенствованный алгоритм расчета и визуализации низкочастотного магнитного поля. В каждой точке поля вектор магнитной индукции направлен по касательной к линии магнитного поля, таким образом, мы не используем заранее заданную конфигурацию пространства для сетки точек, но, находясь в данной точке, получаем следующую точку для расчета, перемещаясь по касательной на небольшое расстояние.

Целью данной работы была разработка алгоритмов и реализация программного обеспечения для расчета и визуализации траектории движения заряженной частицы в низкочастотных магнитных полях, генерируемых системой осесимметричных цилиндрических катушек в различных средах. Модельная задача рассматривалась в цилиндрической системе координат, решение в виде эллиптических интегралов получается из таблично заданной функции. Задача моделирования траектории для произвольной конфигурации цилиндрических катушек рассматривается как задача последовательного расчета значения поля в точке траектории частицы для каждой катушки и векторным сложением этих значений. Вначале для каждой катушки решается модельная задача, чтобы получать распределение создаваемого ей низкочастотного магнитного поля. По результатам рассчитывается сила, действующая на ион в электрическом поле и сила Лоренца в магнитном поле.

В работе реализован подход, при котором расчет поля и траекторий заряженных частиц и визуализация результатов выполняются в единой программной среде — кросс–платформенном программном пакете Unity, программирование осуществлялось на языке C#. Эта среда широко используется для реализации проектов, требующих итеративных улучшений, например, в циклах прототипирования или тестирования. Разработанное приложение может быть развернуто на любой настольной или мобильной платформе с некоторой оптимизацией и незначительными изменениями в коде.

Были оптимизированы ресурсоемкие математические операции, что привело к оптимизации в целом. Приведены основные алгоритмы и результаты расчетов, а также картины визуализации магнитных полей для нескольких систем катушек, позволяющие упростить процесс постановки экспериментов и проектирование аппаратуры на их основе.

ЛИТЕРАТУРА

1. N. Ampilova, I. Soloviev, M. Syasko, Computer modelling the effect of weak electromagnetic field on charged particles using Unity Engine, Proc. 11 Int. Conf. EPE 2020, 22-23 Oct. 2020, Iasi, Romania.
2. B.Kudrin, I.Soloviev. On interpolation methods of low frequency magnetic field in systems for magnetotherapy — Sofia, Bulgaria, 2014. — 16-18 Oct. — P. 154– 157.
3. D.Dimitrov. Medical Systems for Influence of Electromagnetic Field on the Human Body (in Bulgarian), Sophia, Technical University, 2008.
4. В. Ефремов. Метод и лечебно-диагностическая система на основе низкочастотного магнитного поля малой амплитуды. Дисс. на соиск. ктн, СПб, 2006.

УДК 004.89

Сиверин Д.А. (4 курс бакалавриата)
Самочадин А.В., к.т.н., доцент

ПРИЛОЖЕНИЕ ДЛЯ КЛАССИФИКАЦИИ И ОЦЕНКИ ОБЪЕМА МАТЕРИАЛА

Целью работы является создание мобильного приложения для классификации и оценки объема остатков строительства, сноса и грунтов, вывозимых со стройплощадок.

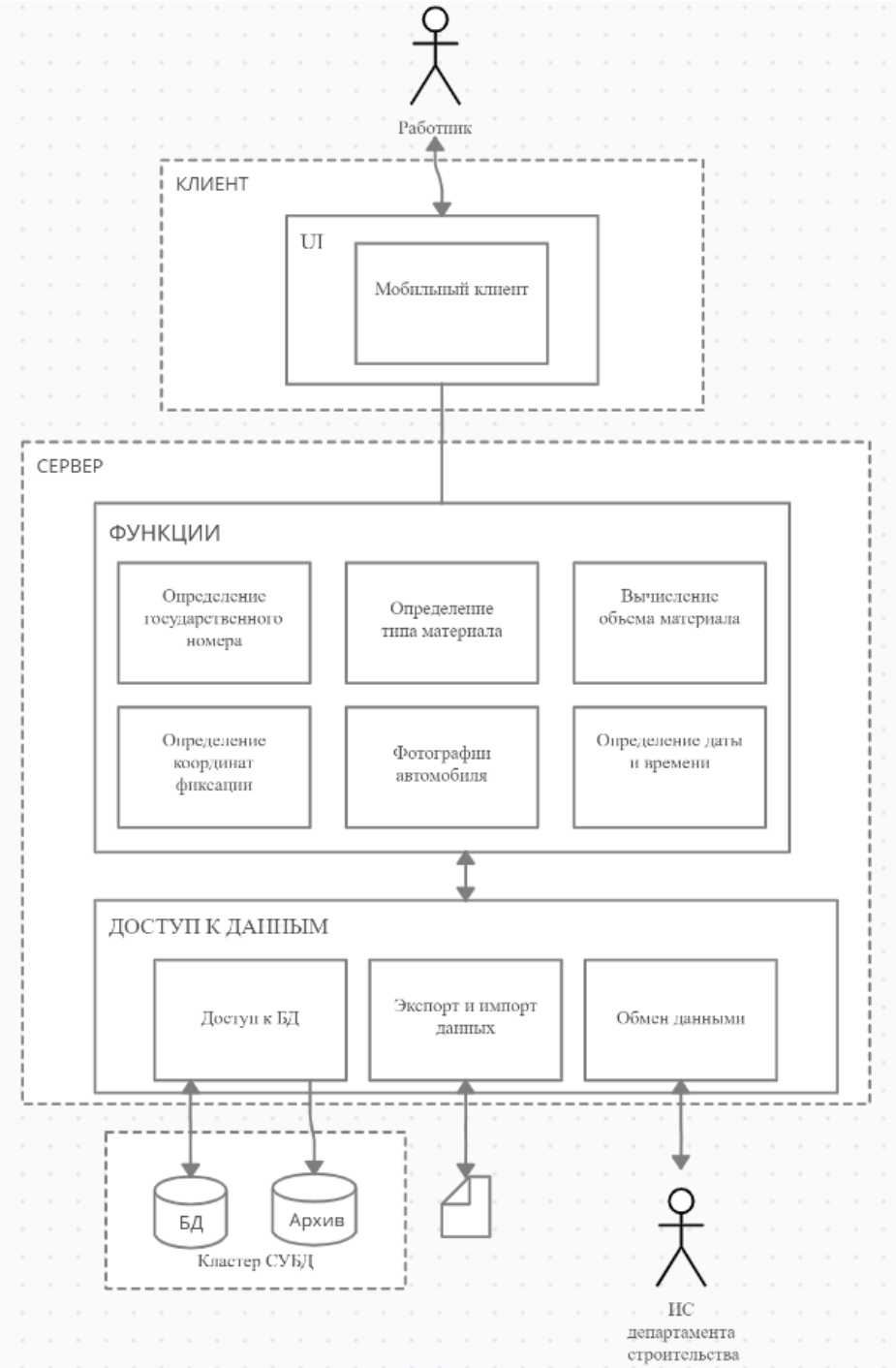


Рисунок 1 – Схема приложения оценки объема остатков строительства, сноса и грунтов.

Приложение состоит из следующих модулей: мобильный клиент, модуль определения ГРН, модуль вычисления объема материала, модуль определения типа материала, модуль определения координат, модуль получения даты и времени.

Мобильный клиент написан на языке программирования Python. В нем предусмотрена работа с камерой, фиксация координат при использовании приложения, а также определение даты и времени.

Модуль определения ГРН. Для этого сервиса используется решение на основе сверточной нейронной сети [2].

Модуль вычисления объема. Реализует методы для вычисления объема материала с использованием нейронной сети.

Модуль определения типа материала использует сверточную нейронную сеть, написанную на языке программирования Python с использованием фреймворка Keras [1, 3].

Модуль получения даты и времени и модуль определения координат. Первый использует запросы к системе мобильного устройства для получения текущих даты и времени. Второй требует при запуске приложения включить геолокацию, это необходимо для фиксации координат места ввоза (вывоза) материала.

Данное приложение на входе должно принимать фотографии грузовых автомобилей с разных сторон, сделанных работником стройплощадки. Используя мобильный телефон делаются следующие фотографии:

- фотографию спереди;
- фотографию кузова автомобиля;
- фотографию сбоку на удалении около 5 метров.

Выходом приложения является государственный регистрационный номер, время, дата и место вывоза, тип материала (приложение определяет около 20 типов различных материалов, таких как щебень, песок, земля и т.п.) и его приблизительный объем.

ЛИТЕРАТУРА

1. Tensorflow [Электронный ресурс]. – Режим доступа: https://www.tensorflow.org/ - (Дата обращения: 13.03.2020).
2. Нейронные сети в детектировании номеров [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/recognitor/blog/329636/ - (Дата обращения: 13.03.2020).
3. Сверточная нейронная сеть [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/348000/ - (Дата обращения: 13.03.2020).

УДК 004.42

Титов Ф.М. (4 курс бакалавриата)
Самочадин А.В., к.т.н., доцент

РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА ПРОТОКОЛИРОВАНИЯ СОВЕЩАНИЙ

Целью данной работы является разработка программного средства протоколирования совещаний на основе аудиозаписи. Имеется следующая программная система стенографирования совещаний на основе аудиозаписи совещания (рис.1). Результатом работы системы является документ, содержащий стенограмму совещания. Стенограмма представляет собой расшифровку аудио в текст с указанием имени говорящего (по возможности) и времени начала его выступления (реплики). Система состоит из следующих основных модулей:

– Модуль разделения дикторов. Необходим для того, чтобы определить кто и когда говорит, чтобы отразить это в документе стенограммы.

– Модуль распознавания говорящего. Необходим для идентификации личности диктора.

– Модуль распознавания речи. Необходим для перевода записанной человеческой речи в текст.

– Модуль формирования стенограммы. Необходим для создания документа стенограммы и приведения к шаблону данных, полученных на предыдущих этапах работы программы.

В рамках работы разработан один из модулей системы – модуль, который формирует протокол совещания.

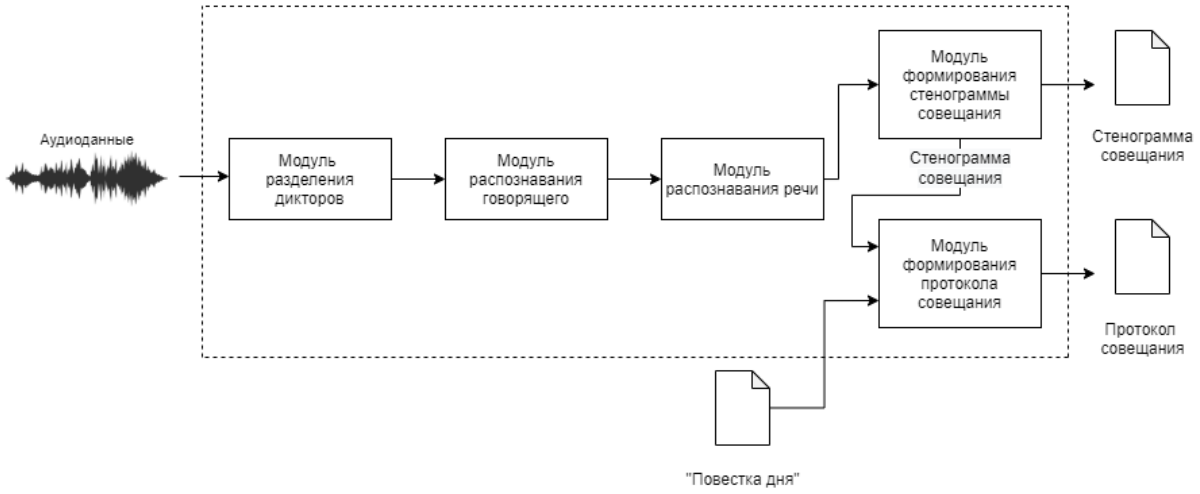


Рисунок 1 – Система стенографирования совещаний на основе аудиозаписи совещания.

Протокол представляет собой документ, содержащий наиболее важные фрагменты совещания, изложенные в краткой форме. Документ подразделяется на блоки. Каждый блок соотнесен с определенным вопросом, который должен быть рассмотрен на совещании. Одним из требований к модулю формирования протокола было формирование документа автоматически с использованием явных маркеров, которые определяют начало и конец блока текста, который необходимо отразить в протоколе.

В соответствии с требованиями было принято решение использовать подход, основанный на использовании явных маркеров (рис.2). Для начала занесения текста в соответствующий блок протокола необходимо произнести команду, означающую началу блока, для завершения занесения текста – команду конца. Команды начала и конца блока должны быть установлены заранее для корректной работы программы и должны произноситься громко и четко во время совещания. Таким образом, задача заключается в том, чтобы распознать эти команды в тексте стенограммы, которая уже сформирована к моменту создания протокола.

Рисунок 2 – Формирование протокола с использованием явных маркеров.

Описанное решение вполне удовлетворяет установленным требованиям, однако является неудобным в эксплуатации, так как для занесения информации в протокол используются явные команды. Человек может забыть произнести необходимую команду, вследствие чего необходимая информация не отразится в протоколе, либо отразится в искаженной форме. Для исправления этих недостатков предлагается алгоритм с использованием неявных маркеров и алгоритмов реферирования.

Общая схема решения представлена на рис.3. На вход алгоритму подается стенограмма совещания, документ «Повестка дня», а также перечень стандартных решений. Стенограмма формируется на предыдущих этапах работы системы. Документ «Повестка дня» содержит в себе вопросы, которые обсуждались на совещании. Алгоритм работает следующим образом. Сначала производится вычленение неявных маркеров из документа «Повестка дня» для каждого вопроса из списка вопросов, необходимых к рассмотрению. Затем производится поиск необходимых смысловых частей на основе статистики вхождений неявных маркеров в рассматриваемом блоке текста стенограммы. Таким образом блок стенограммы соотносится с блоком протокола, в котором необходимо отобразить информацию относительно рассматриваемого вопроса. После этого к выделенному блоку стенограммы применяется алгоритм реферирования, основанный на использовании глубоких LSTM-нейросетей и механизма внимания, и полученные данные заносятся в соответствующий блок документа протокола. Данный алгоритм был реализован на языке программирования Python. Слой механизма внимания был заимствован из репозитория thushv89/attention_keras.

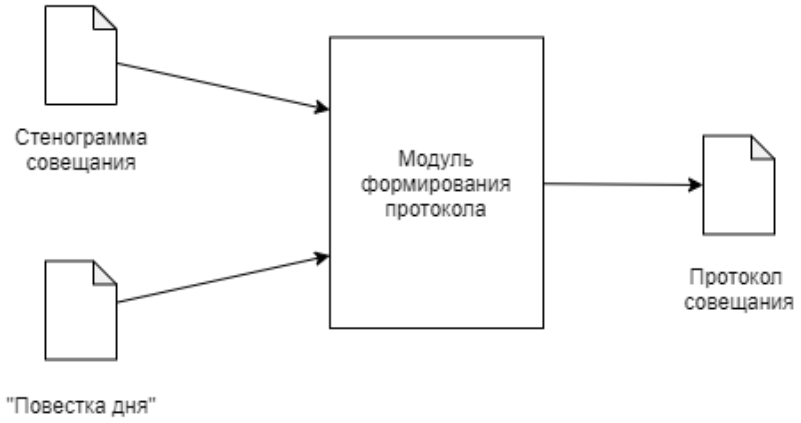


Рисунок 3 – Общая схема решения, основанного на использовании неявных маркеров.

ЛИТЕРАТУРА

1. Мартелли Алекс, Холден Стив Python. Справочник. Полное описание языка. – Изд. «Диалектика-Вильямс», 2018, 896с.
2. Kasimahanthi Divya, Kambala Sneha, Baisetti Sowmya, G Sankara Rao Text Summarization using Deep Learning. - International Research Journal of Engineering and Technology (IRJET), 2020.

УДК 004.051

Шаповалова И.А. (4 курс бакалавриата)
Коликова Т.В., ст. преподаватель

GRAPHQL - АЛЬТЕРНАТИВА REST В РАЗРАБОТКЕ WEB API

Целью работы является изучение основ GraphQL – языка запросов для API – путем сравнения его со стандартом REST API, выявление достоинств и недостатков этого подхода, а также выводы о целесообразности его применения в проектах.

API[1] – это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными. Стандартом для разработки Web API является архитектурный стиль REST[2]. Альтернативным решением является язык запросов

[1] Application Programming Interface
[2] Representational State Transfer

для API – GraphQL. Сравним два этих подхода по базовым критериям клиент-серверного взаимодействия:

Критерий	REST	GraphQL
Протокол передачи данных	HTTP	HTTP, WebSocket, SSH, CLI и др.
Конечные точки	Множество конечных точек, соответствующих определенным ресурсам	Одна конечная точка для любых запросов
Доступные HTTP методы	GET, POST, PUT, PATCH, DELETE	POST
Формат передачи данных	Различные форматы (XML, JSON, HTML, plain text и т. д.)	JSON
Полезная нагрузка HTTP запроса (тело)	Опционально	Обязательно, содержит в себе GraphQL запрос
HTTP статусы запроса	Гибкая настройка на сервере в соответствии со списком кодов состояния HTTP	Если запрос выполнился, то в большинстве случаев код 200 (500 при внутренней ошибке сервера)
Контракт между клиентом и сервером	Для определения явного контракта необходимо использовать дополнительные инструменты (например, OpenAPI/Swagger)	GraphQL схема[1] определяет все доступные операции с API, включая доступные поля, аргументы и ответы
Язык программирования для сервера	Любой	Любой
Язык программирования для клиента	Любой	Любой

Однако у подхода REST есть существенные недостатки:

– отсутствие четко определенных контрактов между клиентом и сервером;

– чрезмерная или недостаточная выборка данных (overfetching и underfetching) – запросы REST зачастую содержат либо слишком много данных, либо недостаточное количество, и создают необходимость в дополнительных запросах.

[1] Описание всех запросов, которые можно сделать к API, включая доступные для запроса данные, аргументы, ответы с явным указанием типов (строгая типизация)

GraphQL решает эти проблемы с помощью языка запросов. Сам запрос содержится в теле посылаемого HTTP метода POST. Запросы имеют определенную структуру, которая показана на Рисунок .

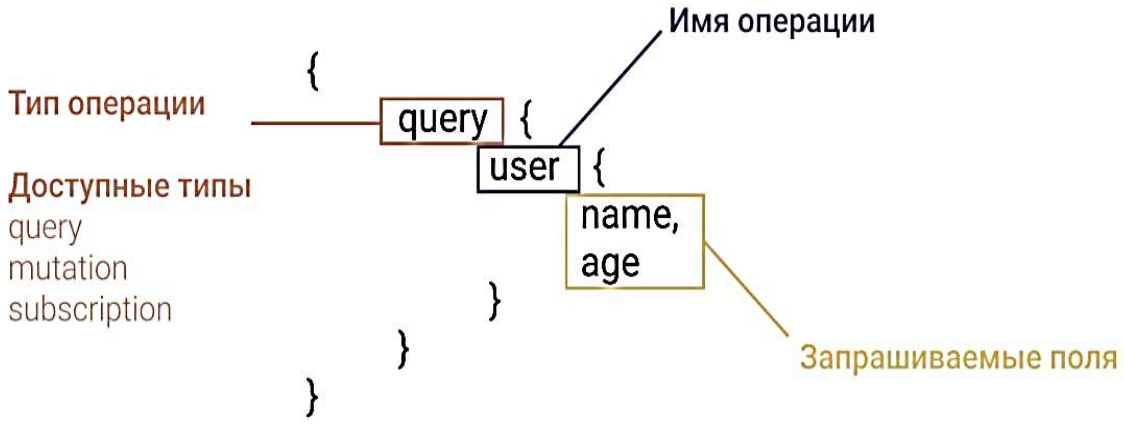


Рисунок 1 – GraphQL запрос.

Имя операции определяется разработчиком API и соотносится с ресурсом и действием, которое выполняется на сервере. Запрашиваемые поля – самое большое преимущество GraphQL. Клиент явно указывает набор данных, с которым хочет работать. С помощью GraphQL схемы могут описываться сложные зависимости и таким образом решается проблема недостаточных или избыточных данных, посылаемых в REST запросах.

Выделим недостатки GraphQL по сравнению с REST архитектурой:

– конечные точки в REST более ясно и понятно отражают суть выполняемых действий;

– необходимость учить язык запросов GraphQL;

– необходимость использовать сторонние библиотеки для внедрения GraphQL в проект;

– нет кэширования (необходимо использовать дополнительные библиотеки);

– если необходимы все поля запрашиваемого объекта, то все они должны быть явно объявлены.

Очевидно, что у GraphQL есть свои достоинства и недостатки, поэтому нельзя называть его полноценной заменой REST API, только альтернативой. Использование того или иного подхода зависит от создаваемой системы и квалификации разработчика. Возможные сценарии применения GraphQL API:

– мобильный API (в этом случае важна производительность сети и оптимизация полезной нагрузки единичного сообщения);

– сложные системы и микросервисы (в этом случае может быть необходимость в объединении данных из нескольких баз данных, сервисов или других источников).

ЛИТЕРАТУРА

1. Бэнкс А., Порселло Е. GraphQL: язык запросов для современных веб-приложений. СПб.: Питер, 2019.
2. Comparing API Architectural Styles: SOAP vs REST vs GraphQL vs RPC // AltexSoft - Technology & Solution Consulting Company URL: https://www.altexsoft.com/blog/soap-vs-rest-vs-graphql-vs-rpc/ (дата обращения: 01.03.2021).

УДК 004

Томилин И. С. (3 курс бакалавриата)
Фёдоров С. А., ст. преподаватель

МЕТЕОСТАНЦИЯ С ВОЗМОЖНОСТЬЮ УДАЛЕННОГО УПРАВЛЕНИЯ И МОНИТОРИНГА

Целью работы является разработка метеостанции, для мониторинга погодных условий окружающей среды (влажность, температура, атмосферное давление), которая превосходит по параметрам и возможностям, решения, предлагаемые рынком на текущий момент.

Предлагаемые устройства на рынке, имеют ограниченный функционал, который в большинстве случаев нельзя выбирать и настраивать. Устройства с большим функционалом, имеют завышенную и неоправданную стоимость, и все равно не имеют возможностей, для гибкой настройки. Разрабатываемый комплекс лишен этих недостатков, его функционал может быть расширен в зависимости от нужд и фантазии пользователя, однако для расширения функционала нужно обладать навыками программирования и понимания аппаратной части устройства.

Аппаратная часть состоит из портативного компьютера Raspberry Pi3 Model B+, основанного на архитектуре ARM и датчика измерения погодных условий BOSCH BME280. Для удобного восприятия человеком, собранные статистические данные визуализируются в виде графиков. Для доступа из любого места, на устройстве развернут веб-сервер, который отображает статистику за указанный период на веб-страницу.

Программно-аппаратный комплекс должен обеспечивать:

– взаимодействие с датчиком BMP280 по шине связи I2C;
– сбор статистики в базу данных;
– отображение визуализированного представления данных на веб-страницу;
– автоматический запуск всего пайплайна при старте устройства.

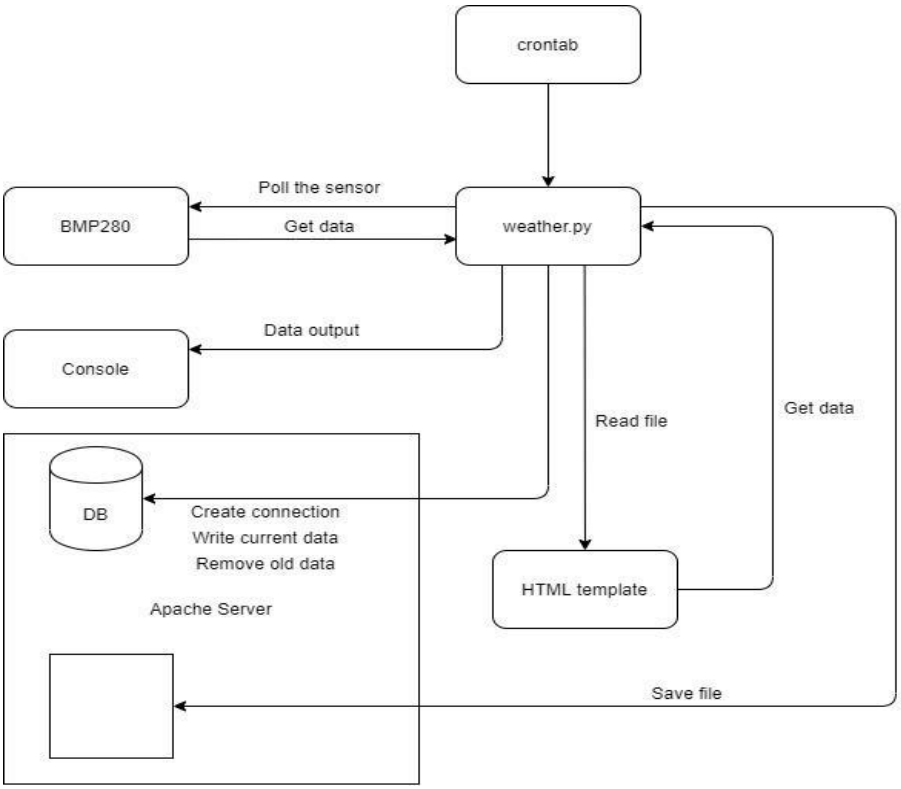


Рисунок 1 – Подробная схема работы приложения в виде UML диаграммы.

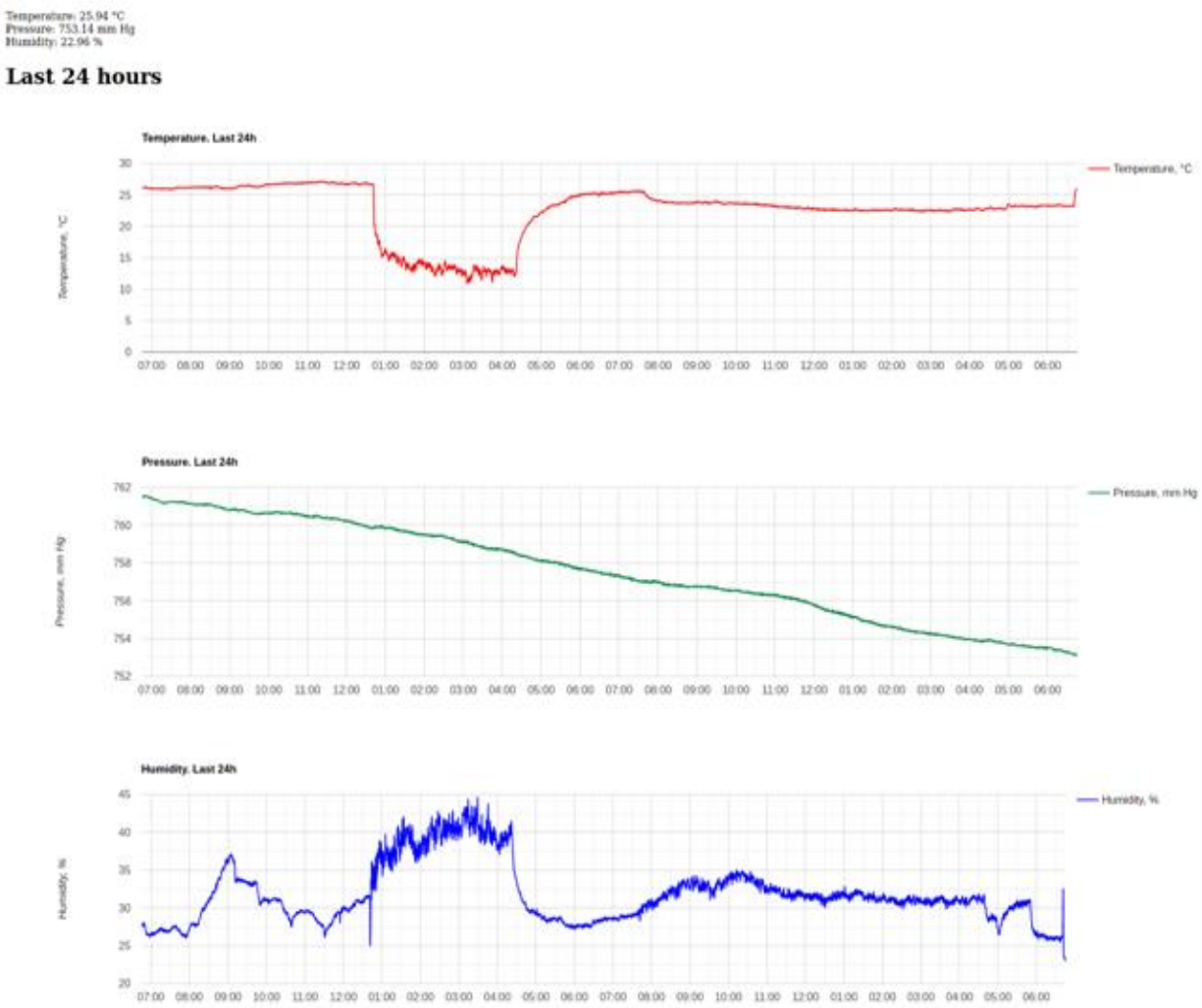


Рисунок 2 – Визуализация полученных данных за временной промежуток.

На текущий момент в проекте реализовано взаимодействие с датчиком, сбор статистики в БД, автозапуск всех необходимых демонов приложений и визуализация собранной статистики на веб-страницу.

В будущем планируется добавить отправку push-уведомлений в мессенджер, для автоматического уведомления пользователя о критических погодных условиях. Взаимодействие с внешними системами, устройствами. Примером внешних систем может послужить включение обогревателя при низкой температуре или более интенсивное проветривание помещения при высокой влажности.

ЛИТЕРАТУРА

1. Lutz M. Python Pocket Reference: Python in Your Pocket : Paperback – Illustrated : O'Reilly Media, 2014. - 266 p.
2. [Электронный ресурс] https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/
3. [Электронный ресурс] https://www.raspberrypi.org/documentation/
4. [Электронный ресурс] https://devguide.python.org/
5. [Электронный ресурс] https://httpd.apache.org/docs/
6. [Электронный ресурс] https://www.sqlite.org/docs.html

УДК 004

Шевченко А.Г. (2 курс магистратуры)
Тышкевич А.И., к.т.н., доцент

РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ ТЕХНОЛОГИЧЕСКОГО ОБСЛУЖИВАНИЯ И УПРАВЛЕНИЯ КОНФИГУРАЦИЯМИ СПЕЦИАЛИЗИРОВАННЫХ РАДИОСТАНЦИЙ

Благодаря развитию технологий, направленных на оптимизацию и упрощение использования существующих устройств, способ передачи данных с помощью радиоволн набирает популярность. В данной работе рассматривается проектирование и разработка распределенных систем технологического обслуживания и управления конфигурациями специализированных радиостанций.

Цель работы заключается в изучении технологий передачи данных в информационной системе между инструментами настройки и управления радиостанциями и программным обеспечением на удаленном ПК, а также в разработке серверной части и подключении базы данных.

Решение данных задач осуществляется с помощью разработки клиент-серверного приложения, в котором пульт управления радиостанциями представляет клиентскую часть и реализован на базе мобильного устройства, а сервер вместе с базой данных разворачивается на удаленном ПК. Для реализации поставленных задач было принято использовать объектно-ориентированный язык программирования - Java, клиентская часть развернута на базе мобильного устройства и разрабатывается в среде Android Studio, серверная часть разрабатывается в IntelliJ IDEA, инструментом разработки баз данных был выбран MySQL.

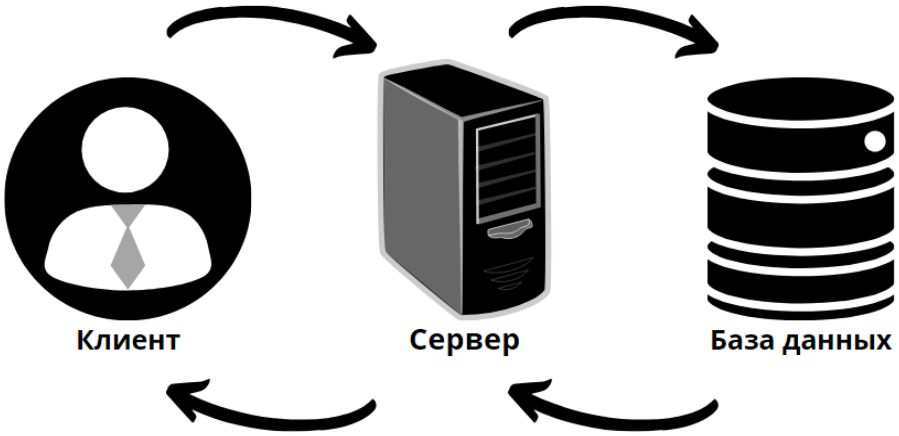


Рисунок 1 – Клиент-серверная архитектура.

На рисунке 1 изображена многоуровневая архитектура клиент-сервер, в которой обработка данных выделяется в отдельное звено и находится полностью на сервере. Плюсами данной архитектуры являются легкая интеграция и доступность из веб-браузеров, а минусами – сложная разработка и дополнительное администрирование.

Таким образом в ходе выполнения работы реализовывается взаимодействие между составляющими информационной системы путем обмена данными о проделанной работе на специализированных радиостанциях. Вследствие чего происходит автоматизация работы технологического обслуживания и управления конфигурациями специализированных радиостанций.

ЛИТЕРАТУРА

1. Шевченко А. Г., 2019. Выпускная квалификационная работа бакалавриата. Пользовательский интерфейс носимого пульта управления радиостанциями на базе мобильного устройства.

2. Майер Р., 2017. Android 2. Программирование приложений для планшетных компьютеров и смартфонов.
3. Бернс Брендан, Русецкий К., 2019. Распределенные системы. Паттерны проектирования.
4. К. Дж. Дейт., 2010. SQL и реляционная теория. Как грамотно писать код на SQL.

УДК 004.925

Шульц С. В. (2 курс магистратуры)
Петров О. Н., к. т. н., доцент

ИССЛЕДОВАНИЕ ПРИМЕНЕНИЯ ТЕХНОЛОГИЙ СМЕШАННОЙ РЕАЛЬНОСТИ В ЗАДАЧАХ ИССЛЕДОВАТЕЛЬСКОГО ПРОЕКТИРОВАНИЯ

Целью работы является исследование возможности применения технологий смешанной реальности в задачах исследовательского проектирования, а также разработка программной реализации прототипа системы исследовательского проектирования для судостроительной отрасли на базе технологий смешанной реальности с целью оценки практической применимости такой системы.

Смешанная реальность является следствием объединения реального и виртуальных миров для созданий новых окружений и визуализаций, где физический и цифровой объекты сосуществуют и взаимодействуют в реальном времени, что уже само по себе говорит о высоких требованиях к аппаратному обеспечению, на котором запускаются такие системы.

В ходе исследования было выявлено, что наиболее производительным, прогрессивным и активно развивающимся решением в сфере создания иммерсивных систем является гарнитура смешанной реальности Microsoft HoloLens 2. В отличие от большинства других устройств виртуальной, дополненной или смешанной реальности, гарнитура HoloLens 2 полностью автономна и не требуют подключения к компьютеру.

Программная реализация прототипа системы позволяет выполнять визуальную компоновку судов различных типов, вариацию форм корпусов и главных размерений, моделировать качку судна при заданном волнении.

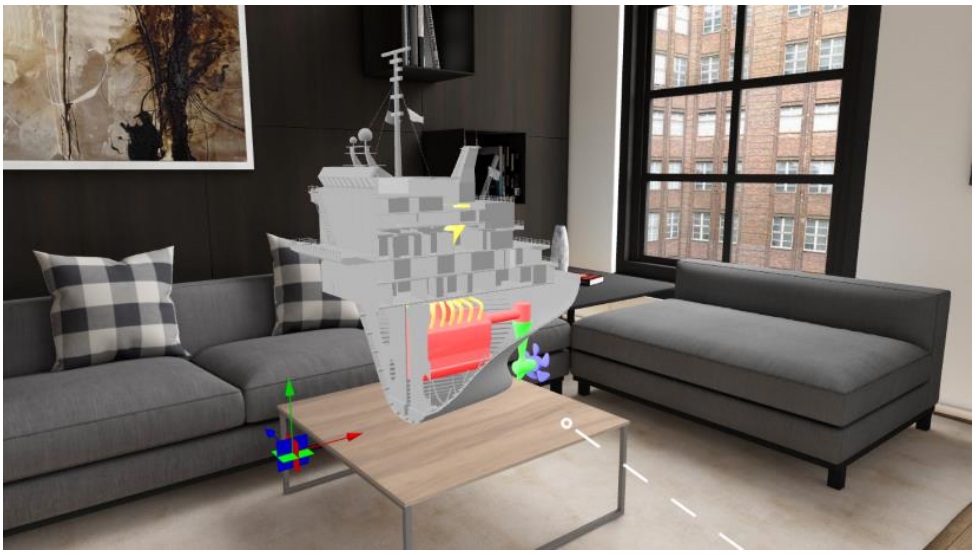


Рисунок 1 – Осмотр кормового отсека и надстройки судна в виртуальном окружении с сечением плоскостью параллельной диаметральной плоскости судна.

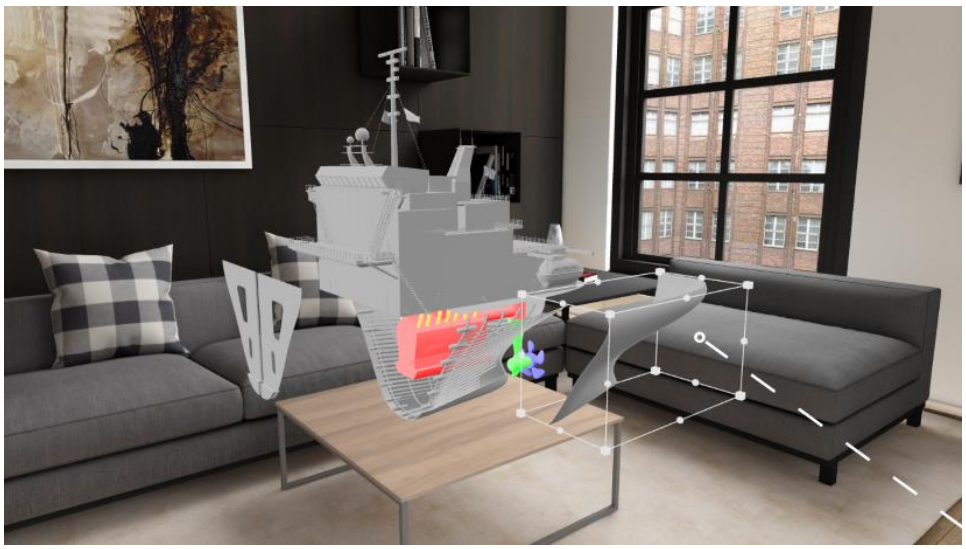


Рисунок 2 – Осмотр и компоновка оборудования кормового отсека судна в виртуальном окружении. Выделены элементы оборудования судна, перемещение которых учитывается при дальнейшем физическом моделировании.

Несмотря на высокие (для данного класса устройств) аппаратные возможности гарнитуры HoloLens 2, они все же недостаточны для совместного выполнения физического моделирования и визуализации. По этой причине система имеет клиент-серверную архитектуру.

Клиентская часть отвечает за визуализацию сцены. Данные для визуализации приходят по беспроводному соединению от серверной части, которая занимается физическим моделированием.

Выполнение физического моделирования на отдельном вычислителе позволяет значительно увеличить время автономной работы гарнитуры.

Дополнительным плюсом такого решения является возможность многопользовательской работы (совместное наблюдение), когда с разных ракурсов несколько пользователей могут видеть разные стороны объекта.

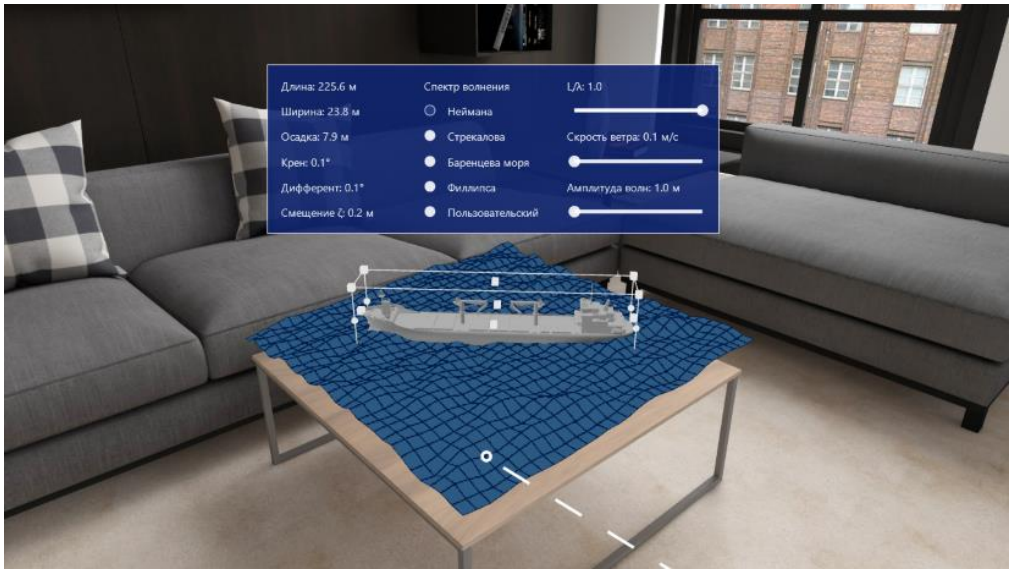


Рисунок 3 – Моделирование качки судна в виртуальном окружении на примере спектра морского волнения Неймана.

Моделирование пространственно-временного поля морского волнения $y(\omega,x,z,t)$ по заданному энергетическому спектру процесса волнения $S(\omega)$ осуществлено с помощью модифицированной модели Лонге-Хиггинса:

$$y(\omega,x,z,t)=\sum_{i=1}^{N}\sum_{j=1}^{M} r_{ij}\cdot\cos\left(k_x\cdot x\cdot\cos\alpha_j+k_y\cdot\sin\alpha_j\cdot z-\omega_i\cdot t+\varepsilon_{ij}\right), \quad (1)$$

где r_{ij} – амплитуды гармоник, вычисляемые по энергетическому спектру $S(\omega)$;

ε_{ij} – случайные фазы, равномерно распределенные по интервалу $[0,2\pi]$;

(k_x,k_y) – волновые числа, связанные между собой через коэффициент трехмерности $\gamma_t=\dfrac{k_x}{k_y}$.

Моделирование основных видов качки (бортовой, килевой и вертикальной) на нерегулярном волнении выполнено с помощью системы нелинейных дифференциальных уравнений:

$$\begin{cases}\zeta''+2\mu_\zeta\cdot\zeta'+n_\zeta^2=\chi_\zeta\cdot n_\zeta^2\cdot y_0\\ \theta''+2\mu_\theta\cdot\theta'+n_\theta^2=\chi_\theta\cdot n_\theta^2\cdot y_x'\\ \psi''+2\mu_\psi\cdot\psi'+n_\psi^2=\chi_\psi\cdot n_\psi^2\cdot y_z'\end{cases}, \quad (2)$$

где $\zeta(t)$, $\theta(t)$, $\psi(t)$ – вертикальное смещение судна, угол бортовой качки (крена) и угол килевой качки (дифферента) в момент времени t;

2μ, n^2, χ – коэффициенты демпфирования, квадраты частот собственных колебаний, поправочные коэффициенты для каждого вида качки (уникальны для проекта судна).

ЛИТЕРАТУРА

1. HoloLens2 hardware [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/hololens/hololens2-hardware
2. What is Mixed Reality? [Электронный ресурс]. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/discover/mixed-reality
3. Нечаев Ю. И., Бухановский А. В., Иванов С. А. «Виртуальное моделирование динамики судна на морском волнении в интеллектуальных тренажерах» // Искуственный интеллект, 2004.
4. Бородай И. К., Нецветаев Ю. А. «Качка судов на морском волнении» – Ленинград: Судостроение, 1969.
5. Лонге-Хиггинс М. С. «Статистический анализ случайной движущейся поверхности»: перевод с англ. // Ветровые волны Москва: Издательство иностранной литературы, 1962.

УДК 004.94

Александров А. В. (4 курс бакалавриата)
Сениченков Ю. Б., д.т.н, профессор

МОДЕЛЬ ОПТИМИЗАЦИИ СВЕТОФОРНЫХ ФАЗ

Транспортное движение стремительно развивалось в последнее столетие, становилось всё обширнее и сложнее. А с ростом городской инфраструктуры загруженность дорог в больших городах стала огромной проблемой. С момента сильной автомобилизации городов, и вплоть до сегодняшнего дня многие страны пытались решить её разными способами, где-то относительно успешно, где-то это только ухудшало ситуацию, но в целом теоретическая база в этой области хорошо развилась за последние десятилетия. Но так можно сказать лишь о теории, на практике же всё становится сложнее, вводить какие-то изменения в структуру города или в экономические и урбанистические законы - занятие очень затратное. Поэтому все эти изменения сначала моделируют с помощью математических и имитационных моделей, которые в современных реалиях стали наиболее эффективным способом продемонстрировать реальное поведение дорожного трафика на искусственной модели.

Целью данной работы является создание модели дорожного трафика, которая поможет решить проблему оптимизации фаз светофоров. Менять длительность фаз на нефункционирующих дорогах не целесообразно, а на дорогах с постоянным трафиком невозможно. Это может привести к огромным заторам и неожиданным последствиям. Поэтому предлагается создать имитационную модель нужной дорожной области, схожую по характеристикам с реальной дорожной системой, и проводить оптимизационные эксперименты в ней. А их результаты уже применять на реальной дорожной системе.

В разных средах моделирования есть возможность реализовать подобную задачу, а в некоторых, таких как AnyLogic [3], данная задача уже была решена и составлена имитационная модель. Мы же собираемся разработать библиотеку, помогающую решить поставленную задачу, для среды моделирования AnyDynamics. На данный момент удобных средств или примеров реализации дорожных библиотек там нет.

Ключевой фактор моделирования дорожного трафика, который надо учитывать при разработке: движение транспорта считается многополосным. В теории дорожного моделирования есть два основных подхода, это макроскопическое моделирование, где транспорт рассматривается как единый поток объектов (подобие гидродинамической системы) и микроскопическое моделирование, где каждый объект рассматривается отдельно, а общие характеристики выводятся из взаимосвязей множества объектов.

Модель должна быть способна выполнять данные задачи:

– Однополосное движение в рамках одного сегмента дороги. В микромоделировании чаще всего решается с помощью моделей типа OVH, клеточного автомата или модели безопасного расстояния на основе теории Гиппса [1]. В данной работе будет использован последний вариант, т. к. конфигурация модели наиболее приближена к реальным дорогам и наиболее понятна пользователю.

– Многополосное движение в рамках одного сегмента. Достигается с помощью модели MOBIL Мартина Трейбера [5].

– Свободное движение в рамках одной дороги (с источника до перекрёстка)

– Работа трёхстороннего и четырехстороннего перекрёстка
– Работа светофорных схем
– Визуализация глобальных характеристик в виде графиков

На данный момент реализовано многополосное движение машин и сделана минимальная визуализация с помощью встроенных средств AnyDymaics (Рисунок 1). После проведения процесса оптимизации фаз нужно сравнить итоговые характеристики с уже имеющимися моделями других сред моделирования (AnyLogic или AimSun, например).

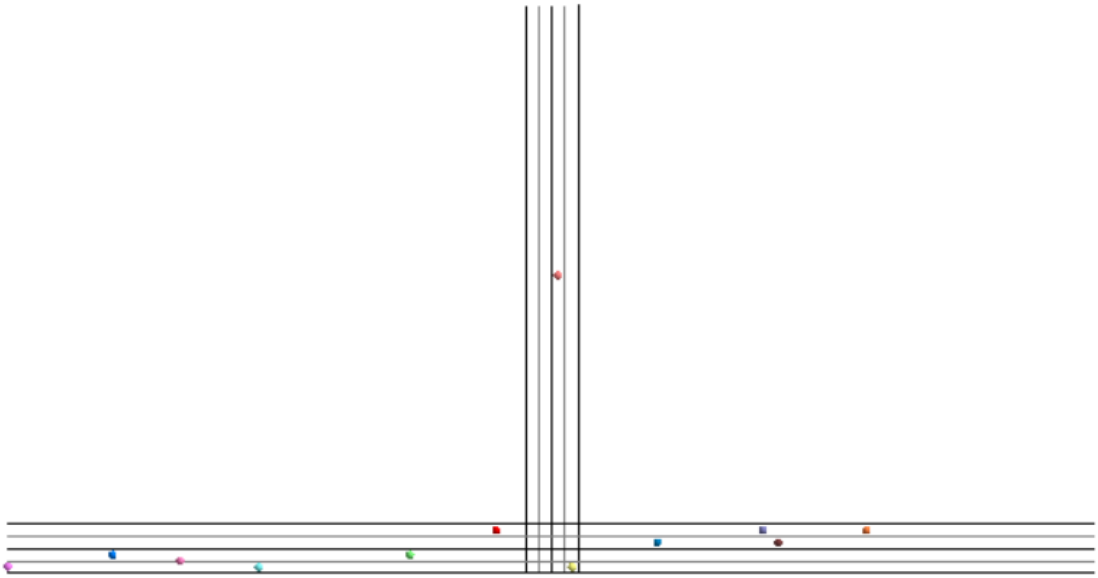


Рисунок 1 – Визуализация движения

ЛИТЕРАТУРА

1. Andrew Lansdowne, Traffic Simulation using Agent-Based Modelling, 2006
2. Куликов Даниил Владимирович, Задачи математического моделирования транспортных потоков мегаполиса, 2018
3. Сайт AnyLogic [URL] https://www.anylogic.ru/
4. Д.Ю. Долгушин, Т.А. Мызникова, Имитационное моделирование автотранспортных поток для оценки альтернативных схем организации дорожного движения в городских условиях, 2011
5. Treiber M., Kesting A. Modeling Lane–Changing Decisions with MOBIL; traffic and granular flow. — 2009. — Part I. — P. 211--221.

УДК 004.056.53

Афанасов А.К. (4 курс бакалавриата),
Литвинов Ю.В., к.т.н., доцент

ИЗВЛЕЧЕНИЕ ДАННЫХ ANDROID-ПРИЛОЖЕНИЯ WHATSAPP

Целью работы является исследование возможных способов получения доступа к данным приложения WhatsApp на Android-устройстве. Экспертам в области криминалистики необходимы инструменты для извлечения данных, которые, в частном случае, можно извлекать из мессенджеров, хранящих информацию о пользователях. Компания Belkasoft занимается разработкой такого инструмента (Belkasoft Evidence Center X), одной из задач

которого является извлечение данных из облачных хранилищ. От компании была поставлена задача исследовать извлечение данных WhatsApp для платформы Android.

Приложение WhatsApp создает зашифрованные резервные копии баз данных пользователя на устройстве и в облаке (Google Drive для платформы Android). Выгружать данные возможно, как при наличии, так и при отсутствии Android-устройства. Для расшифровки баз данных используется key-файл, хранящийся в приватной области памяти файловой системы Android. Этот файл возможно получить двумя способами: либо выгрузить из устройства, либо сгенерировать. В открытом доступе есть проекты, решающие задачи выгрузки зашифрованных баз данных из Google Drive без расшифровки, извлечения баз данных и ключа дешифрования из устройства, расшифровки баз данных по имеющемуся ключу:

– WhatsApp-GD-Extractor-Multithread [1];

– WhatsApp-Key-DB-Extractor [2];

– WhatsApp-Crypt12-Decrypter [3].

Но в открытом доступе нет решений по генерации ключа дешифрования. Для достижения цели получения доступа к данным приложения WhatsApp без использования Android-устройства не хватает самого главного — ключа шифрования, необходимого для расшифровки данных. Подробное изучение ключа дешифрования и его генерации является основным результатом данной работы. Генерация ключа — это один этап “облачного” метода извлечения данных, содержащего следующие шаги:

1. Выгрузка зашифрованных баз данных из Google Drive. Необходимо знать логин/пароль от Google Drive пользователя, данные которого выгружаются. Логин/пароль могут быть получены, например, прямым вопросом пользователю, либо путем анализа личных данных пользователя (чаты, заметки, пароли браузера и т.п.). В рамках настоящей работы логин/пароль являются исходными данными.

2. Генерация ключа дешифрования путем имитации запросов к серверу WhatsApp. Необходим код, отправляемый сервером в СМС-сообщении на номер телефона, привязываемого к аккаунту WhatsApp.

3. Расшифровка базы данных.

Возможные способы генерации ключа дешифрования: получение ключа в чистом или зашифрованном виде от сервера, генерация ключа в чистом виде на устройстве. Исследование генерации ключа в данной работе показало, что WhatsApp генерирует ключ на устройстве. В начале работы приложения WhatsApp скачивается зашифрованная база данных. Далее ключ дешифрования генерируется на устройстве: часть ключа генерируется самостоятельно, часть данных берется из зашифрованной базы, часть получается от сервера.

Необходимость общения с сервером WhatsApp показало изучение файловой системы Android рутованного устройства (от англ. root — традиционное имя суперпользователя в UNIX-подобных системах): ключ создается после загрузки баз данных из Google Drive. Первым этапом исследования был анализ трафика.

Атакой посредника с использованием Burp Suite Community были перехвачены HTTPS запросы: 3 запроса на регистрацию клиента на сервере WhatsApp и запросы на выгрузку баз данных из Google Drive. По причине сложностей в расшифровке XMPP трафика приложение WhatsApp было необходимо модифицировать для логирования информации XMPP общения между клиентом и сервером.

Реверс-инжениринг приложения с использованием декомпилятора jadx показал структуру байтов key-файла. В ходе исследования оказалось, что для расшифровки базы данных достаточно только последнего блока key-файла. В структуру ключа (длина 158 байт) входят: константная метаинформация о файле ключа дешифрования (1-27 байты); версия шифрования (28-29); версия ключа (30ый); байты из зашифрованной базы данных, которые используются для получения последней части ключа по XMPP (31-78); байты с 79 по 110 не участвуют в расшифровке базы, способ их генерации не установлен; последовательность нулей (111-126); байты, генерируемые после XMPP общения с сервером WhatsApp (127-158).

Для логирования XMPP общения была модифицирована iOS версия WhatsApp (модифицировать Android версию невозможно по причине проверки подписи apk), для чего использовались Cydia Substrate и theos. Логирование показало, что клиент iOS в ходе XMPP общения получает от сервера пакет, длина поля данных которого в байтах совпадает с количеством байт последнего блока key-файла. Также в этих запросах фигурировали данные о версии ключа, совпадающие с теми, что имеются в key-файле. Из этого следует, что этот запрос должен предоставить последние данные для генерации ключа дешифрования. Следующим этапом продолжилось исследование декомпилированного apk WhatsApp с целью поиска генерации параметров XMPP запроса.

Декомпилированный код apk показал, что параметры XMPP запроса на получение данных ключа дешифрования, передаваемые от сервера, генерируются по зашифрованной базе данных. Таким образом, чтобы расшифровать базу данных, необходимо иметь данную зашифрованную базу и имитировать XMPP запрос к серверу. Требуется имитировать HTTPS запросы на регистрацию и XMPP запрос на получение данных ключа.

Для имитации XMPP запроса для начала необходимо получить идентификационные данные пользователя. Они генерируются при регистрации пользователя на сервере WhatsApp. Поэтому сначала необходимо имитировать регистрацию клиента. Способы генерации параметров для регистрации клиента были найдены в проекте, позволяющим общаться с пользователями WhatsApp через консоль [4]. После регистрации и формирования идентификационных данных клиента следует XMPP запрос для получения последнего блока key-файла от сервера. В итоге, полученные данные от сервера являются последним фрагментом ключа дешифрования.

Таким образом, было получено решение для генерации ключа дешифрования Android-приложения WhatsApp. На данный момент реализована имитация регистрации клиента. Сейчас продолжается реализация имитации XMPP запроса. Технология реализации — .NET. Планируется внедрение в продукт Belkasoft Evidence Center X.

ЛИТЕРАТУРА

1. YuriCosta. WhatsApp Google Drive Extractor // Сайт GitHub. URL. https://github.com/YuriCosta/WhatsApp-GD-Extractor-Multithread (дата обращения: 15.03.2021).
2. rotbuf. WhatsApp Key/DB Extractor // Сайт GitHub. URL: https://github.com/rotbuf/WhatsApp-Key-DB-Extractor (дата обращения: 15.03.2021).
3. EliteAndroidApps. WhatsApp Crypt12 Database Decrypter // Сайт GitHub. URL: https://github.com/EliteAndroidApps/WhatsApp-Crypt12-Decrypter (дата обращения: 15.03.2021).
4. tgalal. yowsup // Сайт GitHub. URL: https://github.com/tgalal/yowsup (дата обращения: 15.03.2021).

УДК 004.431

Афанасов А.К., Богданов Е.А. (4 курс бакалавриата)
Баклановский М.В., ст. преподаватель

РЕАЛИЗАЦИЯ АСИММЕТРИЧНОГО МАРКЕРНОГО ПРОЦЕССИНГА НА C66X DSP И ARM CORTEX-A15

Задача поступила из области программирования гетерогенных систем, в которых могут одновременно использоваться различные виды архитектур (высокоскоростные, энергоэффективные, быстро производящие сложные математические вычисления). Такие системы позволяют существенно повышать эффективность вычислений путем разделения задачи на фрагменты и их исполнения специализированными ядрами, наиболее подходящими для вычисления конкретного фрагмента общей задачи. Например, такую систему было бы удобно использовать для обработки изображений с камеры: ARM процессор получает задание

на обработку снимка и делегирует вычисления сложных математический функций, необходимых для обработки изображения, процессору DSP.

Однако, важной частью работы гетерогенной системы является организация коммуникации различных ее узлов для этого на математико-механическом факультете СПбГУ был разработан подход к построению таких систем — асимметричный маркерный процессинг [1]. АМП позволяет обеспечить отказоустойчивость системы, коммуникацию между разнородными процессорами в системе, отладку и единую среду проектирования. Данный подход базируется на абстракции маркера — доступа к выполнению задачи. Согласно АМП маркер передается только тому ядру, которое может его принять и которое наиболее эффективно для выполнения текущего фрагмента общей задачи, а для его передачи используется стек протоколов АМП. Стек состоит из 5 уровней: канальный, сетевой, транспортный, уровень представления и прикладной.

Целью данной работы 3го курса является реализация канального, сетевого, транспортного уровней стека протокола асимметричного маркерного процессинга на процессорах DSP C66x (далее — DSP) и ARM® Cortex™-A15 MPCore™ (далее — ARM). Процессоры входят в состав архитектуры KeyStone 2.

Система на кристалле, используемая в данной работе, имеет два семейства ядер (8 DSP и 4 ARM), которые могут общаться по высокоскоростной разделяемой SRAM памяти Multicore Shared Memory (далее — MSM) размером 6 MB [2, с. 5], которую было решено использовать для передачи маркера и загрузки в нее своих программ, так как такой подход позволяет достичь наиболее высокой скорости работы системы. Языками реализации являются язык ассемблера со стороны DSP и язык Си со стороны ARM. Для разработки использовалась IDE Code Composer Studio, предлагаемая Texas Instrument [3].

Система на кристалле имеет ОС Linux. Но если по каким-то причинам доступ к Linux невозможен или не нужен, то исполнять программы возможно путем подключения к ядрам, используя JTAG интерфейс (для отладки программ, запущенных на ядрах) и Board Management Controller (контролирует минимальную настройку используемой платы). В данной работе для обеспечения лучшей производительности был использован второй вариант.

Также с ОС Linux производителем платы поставляется ОС реального времени SYS/BIOS. В данной работе для программ на Си в конфигурационном файле проекта подключались отдельные модули этой операционной системы, а на ядра загружался исполняемый файл, содержащий как написанную программу, так и подключенные модули. Такой подход на первых этапах позволил существенно упростить работу с аппаратными таймерами и организовать ввод/вывод на экран. Однако впоследствии большинство используемых модулей были заменены на ассемблерные вставки, что позволило более тонко настраивать используемые аппаратные компоненты.

Таким образом результатом работы стал набор программ [4; 5] для архитектур ARM и DSP, которые реализуют 3 базовых уровня стека АМП, реализация которых является необходимым минимумом для организации межпроцессного взаимодействия:

– Канальный: позволяет передать байты со стека программы одного процессора в циклический буфер разделяемой памяти другого процессора, а также обнаружить и начать обработку нового пакета, пришедшего от другого процессора.

– Сетевой: занимается проверкой контрольных сумм передаваемых пакетов, формирует заголовки с информацией об отправителе/получателе.

– Транспортный: обеспечивает надежную передачу маркера путем подтверждения получения маркера по типу тройного рукопожатия с некоторыми доработками.

ЛИТЕРАТУРА

1. М. В. Баклановский, Б. Н. Кривошеин, А. Н. Терехов, М. А. Терехов, А. Е. Сибиряков. АСИММЕТРИЧНЫЙ МАРКЕРНЫЙ ПРОЦЕССИНГ // Программная инженерия. — 2018. — Vol. 9, no. 4. — P. 156–162

2. Texas Instruments. 66AK2H14/12/06 datasheet // Сайт Терраэлектроника. URL: https://spb.terraelectronica.ru/pdf/show?pdf_file=%252Fz%252FDatasheet%252F6%252F66ak2h06.pdf (дата обращения: 16.03.2021).
3. Texas Instruments. EVMK2H Evaluation Module // URL: http://wfcache.advantech.com/www/support/TI-EVM/Rev4_0/QuickStartGuide/EVMK2H_QSG_v2.pdf (дата обращения: 16.03.2021).
4. Артем Афанасов. Реализация канального, сетевого, транспортного уровней стека протокола асимметричного маркерного процессинга на C66x DSP// GitHub. URL: https://github.com/ArtyomAfanasov/Asymmetric-Marker-Processing (дата обращения: 16.03.2021).
5. Евгений Богданов. Реализация канального, сетевого, транспортного уровней стека протокола асимметричного маркерного процессинга на ARM// GitHub. URL: https://github.com/jackbogdanov/AMP-for-ARM (дата обращения: 16.03.2021).

УДК 004.4

Барсуков Н.Д., Посметныйс Д. (2 курс магистратуры)
Никифоров И.В., к.т.н., доцент

РАСПРЕДЕЛЕННАЯ МИКРОСЕРВИСНАЯ АРХИТЕКТУРА СИСТЕМЫ АНАЛИЗА ЛОГОВ ПЛАТФОРМЫ «ОТКРЫТОЕ ОБРАЗОВАНИЕ»

Сегодня в связи с эпидемиологической обстановкой в мире практика онлайн-образования набирает все большую популярность. Практически все онлайн курсы реализуются за счет применения онлайн образовательных платформ. Тем не менее, несмотря на свою популярность, онлайн-платформы обладают следующими недостатками [1]:

- недостаточность необходимых технических навыков для проведения занятий в цифровой среде у большой части преподавателей;
- сложность в определении механизмов контроля вовлеченности студентов во время занятий.

В связи с выявленными проблемами перехода образовательных программ на дистанционный формат, появляется необходимость в инструменте, который был помог решить данные проблемы и помочь преподавателям сформировать правильный педагогический дизайн, требуемый для онлайн курсов. Наиболее популярным инструментом на территории РФ является проект «Открытое образование» [3].

«Открытое образование» - образовательная платформа, предоставляющая доступ к онлайн-курсам университетов России. Данная платформа разработана на основе платформы «Open edx»[4] - платформы с открытым исходным кодом. Данный инструмент предоставляет возможности для создания образовательных сервисов. В частности, «Open edx» содержит такой инструмент, как Learning Management System (LMS) - система для управления процессом обучения. LMS даёт возможность публикации учебных материалов [2] (видеолекций, текстов, презентаций), а также проведения аттестации в форме тестирования. Эти возможности использованы платформой «Открытое образование» для публикации курсов российских университетов.

«Open edx» по умолчанию предоставляет инструмент аналитики, который, к сожалению, обладает малым функционалом. Используя «Open edx Analytics REST API» можно получить скромный набор аналитики, и, согласно документации [5], он следующий:

- Получить «активность» пользователя на курсе.
- Получить участников по параметру, в т.ч. пол, дата рождения, по месту принадлежности IP-адреса.
- Получить вопрос, на который чаще всего отвечают неправильно.
- Получить график распределения оценок по курсу

Помимо, скоромного набора аналитики, существует явный недостаток – отсутствие расширяемости аналитических задач. Так же стоит отметить, что аналитический модуль не обновлялся разработчиками с 2015 года.

В связи с представленными проблемами, а также, с открытью исходного кода «Open edx», существует ряд продуктов, которые позволяют проводить аналитику курсов, беря за основу лог-файлы: «Racoon» [6] и «eduNext» [7], «Panorama Openedx» [8]. Все они имеют ряд недостатков, а главным является их проприетарность.

Таким образом была поставлена задача создать программную систему для аналитики с открытым исходным кодом, которую можно было поддерживать студентами местных университетов.

Такую программную систему представили студенты «Санкт-Петербургского Политехнического университета» - «Open Analyzer» [9]. После тестов программного средства на конечном пользователе, были выявлен ряд архитектурных недостатков, которые влияли на скорость обработки аналитических задач. Таким образом, была поставлена новая цель:

Целью работы является повышение производительности анализа активности пользователей за счет распределенной микросервисной архитектуры реализованной в программной системе анализа логов платформы «Открытое образование».

Задачи, которые решает перед собой микросервис:

1. Исследовать архитектуры существующих инструментов анализа активности пользователей на онлайн платформах,
2. Предложить высокопроизводительный и горизонтально масштабируемый подход анализа логов
3. Спроектировать REST - интерфейс программного средства, реализующего подход анализа логов
4. Разработать программное средство
5. Показать прирост производительности предложенного подхода и его реализации в программном средстве на актуальных данных платформы.

Сегодня является актуальным микросервисный архитектурный стиль программирования (МСА). В данном архитектурном стиле, микросервис представляет собой маленькую часть большой бизнес логики и разворачивается независимо от других, что обеспечивает автономность разработчиков и большую гибкость системы [10]. МСА широко используется для распределённых систем, т.к. она позволяет удобно оперировать свойствами модульности, в организации совершенно разных организаций распределенной среды [11].

Каждый микрсосервис необходимо поделить на сервисы, которые будут обрабатывать лог-файлы платформы. которые представляют собой json-документ с описанием событий, происходящих в системе LMS. Событие - это сущность, описывающая отдельную активность пользователя на учебном курсе (например, запись на курс, просмотр видеолекции, отправка ответа в течение тестирования и т.п.).

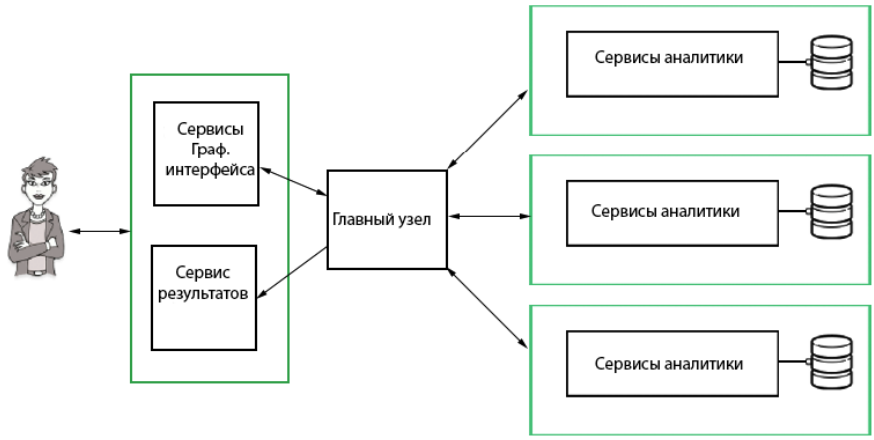


Рисунок 1 – Концепция архитектуры [12].

Концепция представлена на рисунке 1. При получении задачи от сервиса графического интерфейса главный узел начинает отдавать задачи на рабочие узлы. После этого главный узел агрегирует данные и отдает соответствующему сервису результаты.

Для создания новой архитектуры бы выбран язык Java. Он является высокоуровневым, кроссплатформенным и объектно-ориентированным языком программирования, который имеет огромное количество библиотек и фреймворков для удобного, а самое главное, быстрого развертывания микросервисов. Наиболее популярные из них «Spring Boot» и «Micronaut». Так как, фреймворк «Spring» является достаточно тяжеловесным и избыточным для наших задач, выбор пал на «Micronaut». За счет отсутствия некоторых библиотек, а также выполнения «внедрения зависимостей» во время компиляции, «Micronaut» может выиграть в скорости работы микросервиса у «Spring».

Таким образом, была представлена распределенная микросервисная полностью отвечающая поставленным задачам.

ЛИТЕРАТУРЫ

1. Рогачева П.С., Семергей С.В. Проблемы дистанционного образования в период пандемии // Вестник Майкопского государственного технологического университета. 2020. Том 12, №4 С 89.
2. Малышева Е. В. Обучение в системе MOOCs: достоинства и недостатки // Царскосельские чтения, I (XVIII), 358.
3. Никифоров И.В., Толпыгин С.С., Барсуков Н.Д., Сысоев И.М., Разработка микросервиса для получения логов платформы "Открытое образование", НЕДЕЛЯ НАУКИ СПбПУ. материалы научной конференции с международным участием. Санкт-Петербургский политехнический университет Петра Великого. 2019. С. 43-45.
4. Документация «Open Edx» [Электронный ресурс]. - Режим доступа: https://edx.readthedocs.io/projects/devdata/en/stable/internal_data_formats/tracking_logs.html, свободный.
5. Документация «Open Edx Analytics Stack» [Электронный ресурс]. - Режим доступа: https://docs.huihoo.com/edx/open-edx-data-analytics-api/overview.html
6. Документация «Racoon» [Электронный ресурс]. - Режим доступа: https://raccoongang.com/open-edx-services/analytics-integration/
7. Документация «Panorama» [Электронный ресурс]. - Режим доступа: https://www.aulasneo.com/en/panorama-analytics-2/
8. Документация «eduNext» [Электронный ресурс] . - Режим доступа: https://www.edunext.co/articles/open-edx-analytics/
9. Барсуков Н.Д., Сысоев И.М., Перескокова А.А., Никифоров И. В., Посметныйс Д. Анализ Активности студентов на курсах онлайн-обучения на основе логов платформы «OpenEdu» // Труды ИСП РАН, том 32, вып. 3, 2020г., 91-100 (на английском языке)
10. Гольчевский Ю.В., Еромленко А.В. Актуальность использования микросервисов при разработке информационных систем // Вестник Сыктывкарского университета. Сер 1: Математика. Механика. Информатика. 2020. Вып. 2(35). С. 25-36.
11. Опарин Г. А., Богданова В.А., Пашинин А. А. Микросервисы как фундументальная основа распределенного сборочного программирования // ИТНОУ: информационные технологии в науке, образовании и управлении, 2018, №2, с. 21 -25
12. Перескокова А.А., Кольцов А.А., Кочугова В.И., Толпыгин С.С., Никифоров И.В., Архитектура программного аналитического комплекса исследования поведения пользователей на образовательной платформе "Открытое Образование", НЕДЕЛЯ НАУКИ СПбПУ. материалы научной конференции с международным участием. Санкт-Петербургский политехнический университет Петра Великого. 2019. С. 82-85.

УДК 004.4’2

Бетуганов А.А., Хомяков А.А. (4 курс бакалавриата)
Сараджишвили С.Э., к.т.н., доцент

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА МОБИЛЬНОЙ РАЗРАБОТКИ

Целью работы является анализ существующих средств мобильной разработки, выявление конкурентоспособных и перспективных инструментов. При этом будут рассмотрены особенности использования различных инструментов для разработки для наиболее популярных на сегодняшний день мобильных платформ, Android и iOS.

Мы не будем рассматривать нативные средства разработки по очевидным причинам: отсутствие кроссплатформенности, как следствие увеличение штата разработчиков и объема работ.

Анализ существующих на рынке решений показал, что имеются несколько актуальных инструментов для кроссплатформенной разработки, приведенных на Рисунке 1[1]. Рассмотрим следующие из них: Xamarin, React Native и Flutter.

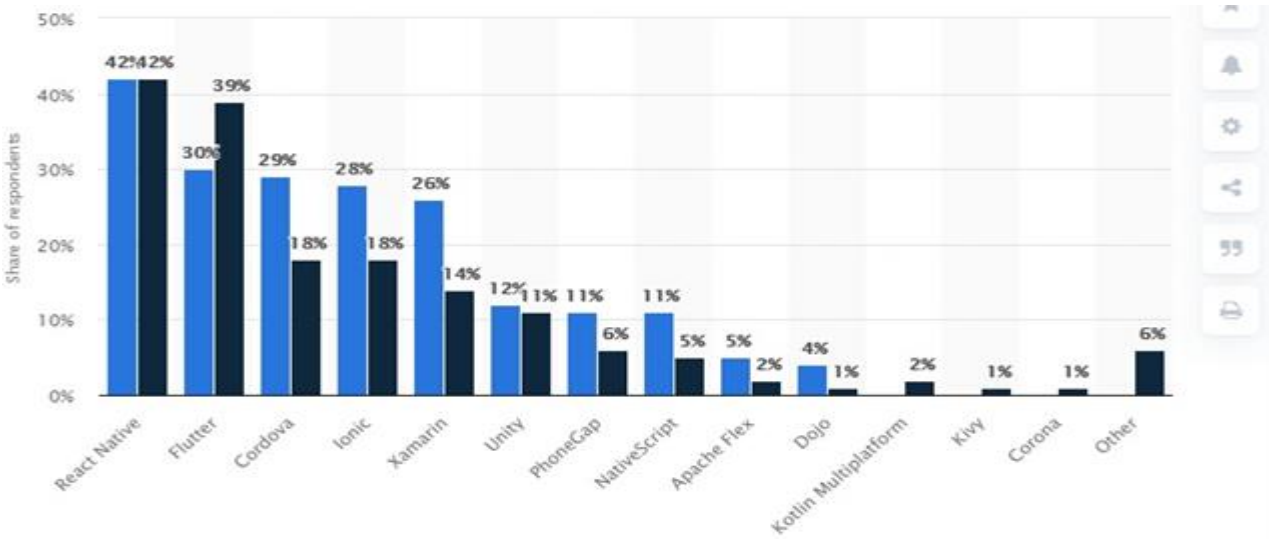


Рисунок 1 – Рейтинг популярности инструментов разработки.

«Xamarin — это платформа с открытым исходным кодом, предназначенная для построения современных производительных приложений для iOS, Android и Windows с .NET.». [2]

На Рисунке 2 [2] приведена подробная схема архитектуры приложения на Xamarin для платформы iOS и Android.

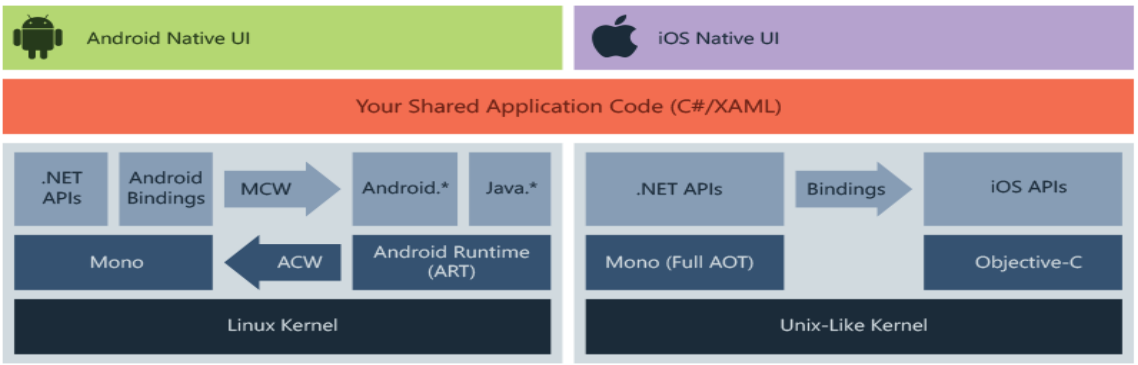


Рисунок 2 – Архитектура кроссплатформенного приложения Xamarin.

Flutter - мобильный SDK с открытым исходным кодом для создания кроссплатформенных мобильных приложений. Flutter-приложения очень похожи на нативные. Рисунок 3 [3] показывает каким образом Flutter взаимодействует с обеими платформами для обеспечения кроссплатформенности.

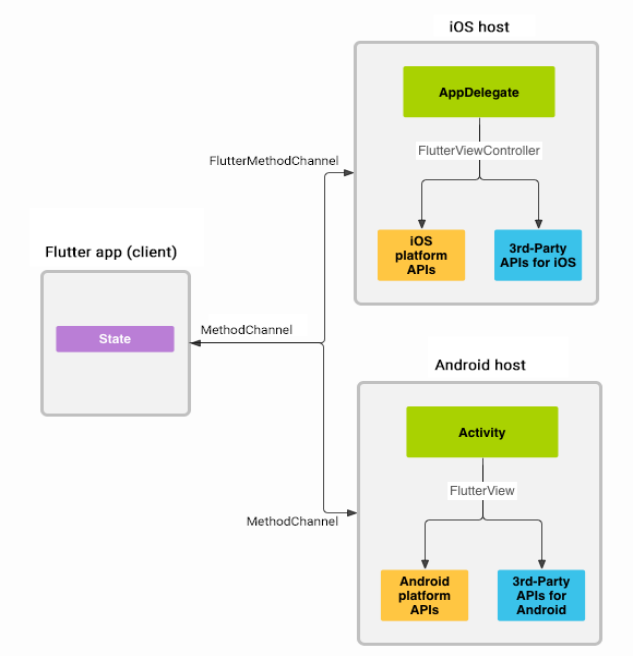


Рисунок 3 – Архитектура кроссплатформенного приложения Flutter.

Вы можете использовать один код для разработки приложения для Android или iOS. Так же Flutter возможно использовать не только для разработки под мобильные платформы, но и для Windows, Linux, Mac и Google Fuchsia.

«React Native — это та же React, но для мобильных платформ. У неё есть ряд отличий: вместо тега div используется компонент View, а вместо тэга img — Image. Процесс разработки остался тем же» [5]. Архитектуру приложения на React Native описывает Рисунок 4[5].

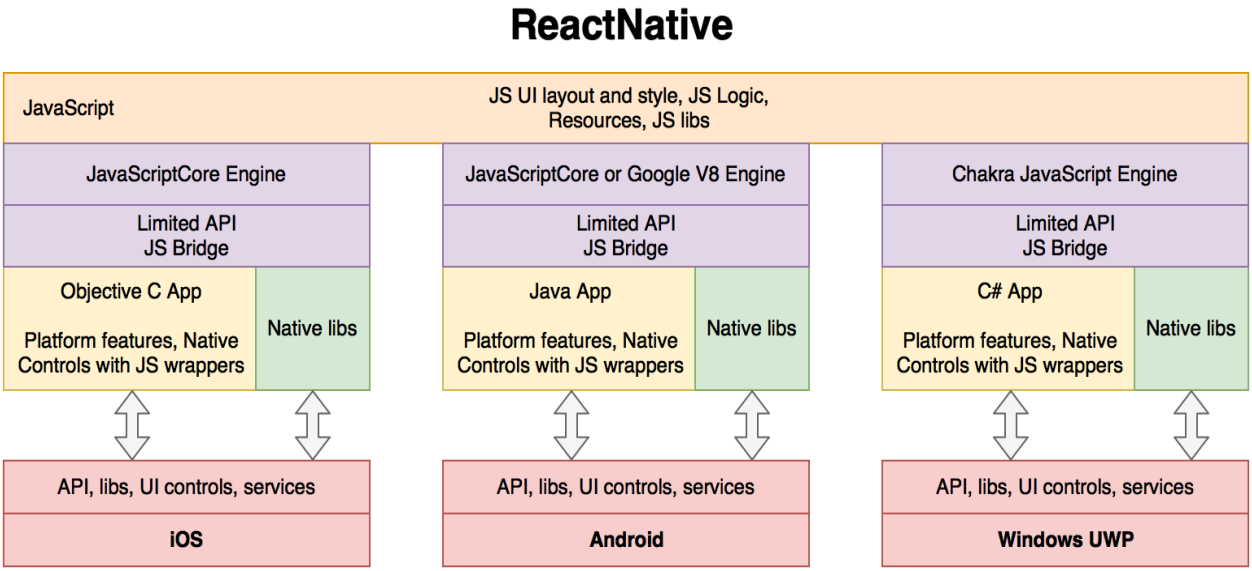


Рисунок 4 – Архитектура кроссплатформенного приложения ReactNative.[6]

Будет производиться сравнение по 7 параметрам: язык разработки, производительность, пользовательский интерфейс, рынок и поддержка сообщества, повторное использование кода, популярные программные продукты, стоимость.

Таблица 1 – Сравнение рассматриваемых инструментов.

Сравнение рассматриваемых инструментов			
Параметры	Xamarin	Flutter	ReactNative
Язык разработки	C# & .net	Dart	JS, Swift, Objective-C, Java
Производительность	Близка к нативной	Сравним с нативной	Близка к нативной
Графический интерфейс	Используются нативные компоненты	Используются проприетарные виджеты	Используются нативные компоненты
Поддержка сообществом	Сильная	Средняя	Очень сильная
Ре-использование кода	Около 90%	Около 90%	Около 90%
Популярные программные продукты	Olo, the World Bank, Storyo	Google ads, Alibaba, SpaceX Go!	Facebook, Instagram, Airbnb
Стоимость	Open-source + оплата за коммерческое использование	Open-source	Open-source

Для кроссплатформенной разработки мобильных приложений все три фреймворка успели зарекомендовать себя среди организаций, стремящихся сократить время и затраты на разработку приложений. Каждый из фреймворков может быть полезен в зависимости от ваших приоритетов и потребностей бизнеса. По совокупности плюсов и минусов, нашим выбором стал Flutter, как оптимальное и перспективное решение.

ЛИТЕРАТУРА

1. Flutter vs React Native: https://codeburst.io/flutter-or-react-native-which-is-the-best-choice-for-2020-355d9473edde
2. Xamarin: https://docs.microsoft.com/ru-ru/xamarin/get-started/what-is-xamarin
3. Официальный сайт Flutter: https://flutter.dev/
4. Flutter — новый взгляд на кроссплатформенную разработку: https://habr.com/ru/company/google/blog/426701/
5. Создание кроссплатформенных приложений с помощью React Native: https://habr.com/ru/company/nix/blog/324562/
6. Architectures of cross-platform mobile frameworks: https://books.binwell.com/architectures-of-reactnative-xamarin-phonegap-qt/3.-architectures-of-cross-platform-mobile-frameworks
7. React Native vs. Xamarin vs. Ionic vs. Flutter: Which is best for Cross-Platform Mobile App Development?: https://www.apptunix.com/blog/frameworks-cross-platform-mobile-app-development/

УДК 004.4'22

Васеха И. А. (4 курс бакалавриата)
Петров А. В., ст. преподаватель

РАЗРАБОТКА ИНСТРУМЕНТА ДЛЯ ГЕНЕРАЦИИ MOCK-СЕРВИСОВ НА ОСНОВЕ ASYNCAPI СПЕЦИФИКАЦИИ

В последние годы широкое распространение получила микросервисная архитектура. Однако при переходе к этой архитектуре доступ к данным становится сложнее, в отличие от монолитной архитектуры. Это связано с тем, что данные, принадлежащие каждому микросервису, являются частными для него и могут быть доступны только через API. Инкапсуляция данных гарантирует, что микросервисы слабо связаны и могут развиваться

независимо друг от друга. При этом встают вопросы, как поддерживать согласованность данных между различными сервисами и каким образом ими обмениваться. Одним из вариантов, решающих эти вопросы, является событийно-ориентированная архитектура, которая включает в себя поставщиков, которые создают потоки событий и публикуют их в каналы, и потребителей, которые прослушивают эти события. Таким образом, у нас появляются понятия событий и каналов, с помощью которых общаются сервисы. При микросервисной архитектуре каждый сервис может разрабатываться разной командой, кроме того, на каналы, в которые публикуются события, может быть подписан не один сервис. Следовательно, необходимо как-то описывать и интерфейс и синхронизировать изменения в нем для упрощения взаимодействия команд. AsyncAPI инициатива, вдохновленная успехом и удобством OpenAPI, используемым при описании REST API, предоставляет язык для описания интерфейсов, а также инструментарий для генерации исходных текстов программ на основе спецификации и наоборот, генерации документации, валидации событий и тестирования. Так как AsyncAPI является open source проектом, то заниматься разработкой инструментов для него может каждый.

Спецификация интерфейса содержит описание каналов и событий для поставщиков и потребителей. Описание событий включает в себя описание заголовков и информационной части, включая ограничения на входящие в состав поля, например, такие как обязательность, минимальная/максимальная длина, диапазон значений, шаблон и другие. Данная информация может быть использована для автоматизации тестирования, например, для генерации допустимых и недопустимых сообщений. В микросервисной архитектуре одни сервисы генерируют данные для других, что, если нам необходимо протестировать функциональность только одного сервиса или некоторой цепочки сервисов? Для таких случаев нередко используются mock-объекты, которые имитируют необходимое поведение недостающих сервисов. При тестировании событийно-ориентированных сервисов необходимо имитировать отправку сообщений в каналы.

Таким образом, в данной работе предлагается реализовать инструмент для генерации mock-сервисов на основе AsyncAPI спецификации. Несмотря на то, что AsyncAPI поддерживает различные протоколы обмена сообщениями (AMQP, MQTT, WebSockets, Kafka, STOMP, HTTP и другие), сперва реализуемый инструмент будет поддерживать только Kafka и AMQP. Основная функциональность инструмента для генерации:

– Создание mock-сервиса, готового к генерации и отправке сообщений, по запросу, содержащему ссылку на AsyncAPI или сам файл спецификации.

– Просмотр информации обо всех сгенерированных mock-сервисах, такой как API для генерации и отправки сообщений и лог отправленных сообщений.

– Отправка запросов к mock-сервису на генерацию и отправку, используя графический интерфейс.

– Запуск в docker-контейнере.

Функциональность mock-сервиса:

– Генерация и отправка сообщений в указанный канал по запросу, содержащему такие параметры, как количество сообщений, тип сообщения (например, допустимое, содержащее все поля и заголовки, допустимое, содержащее только необходимые, поля и заголовки, недопустимое, с нарушением различных ограничений и так далее), установленные пользователем конкретные поля сообщения, формат сообщения (json, xml, string), требуемый канал.

В итоге получим следующие преимущества от использования данного инструмента:

– Отсутствие необходимости работать с отправкой сообщений к брокеру и конфигурировать подключение напрямую из тестов, следовательно, упрощение работы для тестировщиков и концентрация на тест-кейсах.

– При переходе с одного брокера на другой не потребуется вносить изменения в тесты.

– Упрощение поддержки тестов при внесении изменений в интерфейс.

– Гарантированное соответствие тестовых данных спецификации.

Возможные расширения:

– Поддержка других протоколов.

– Генерация сервиса чтения сообщений из каналов с возможностью сравнения с ожидаемым результатом и валидации по AsyncAPI.

ЛИТЕРАТУРА

1. AsyncAPI URL: https://www.asyncapi.com/
2. Microcks URL: https://microcks.io/
3. Apache Kafka URL: https://kafka.apache.org/

УДК 004.42

Васильев Д. С. (4 курс бакалавриата)
Маслаков А. П., ст. преподаватель

РАЗРАБОТКА RESTFUL API НА ОСНОВЕ ЯЗЫКА PYTHON И ФРЕЙМВОРКА FLASK

Сегодня веб-сервисы окружают пользователей повсюду и практически за каждым таким сервисом стоит бэкенд, который реализует бизнес-логику приложения, невидимую для пользователя. Для построения такого компонента программисту необходимо решить несколько важных задач: выбрать способ взаимодействия с сетью, способ передачи и формат передаваемых данных, как получить и не потерять данные от клиента и как их обработать.

Целью данной работы является разработка бэкенд-сервиса, как часть создания собственного многопользовательского клиент-серверного приложения. Серверная часть разрабатываемого продукта должна предоставлять разделенный доступ к данным, хранящимся в базе данных, и соответствовать REST архитектуре [4]. Функционал, который должен быть обеспечен сервисом:

1. Администрирование базы данных, а именно: создание таблиц, связей;
2. Управление данными в базе данных – поддержка CRUD;
3. Предоставление доступа по HTTP-протоколу;
4. Обмен данными, с помощью json-структур;
5. Разделение доступа к данным на основе предоставленных прав конечному пользователю приложения;
6. Поддержка Socket.IO [2] для передачи сообщений в реальном времени.

Для разработки данного решения был выбран язык Python 3, по нескольким причинам: скорость написания кода, простота установки и использования пакетов (библиотек) и отсутствие необходимости компилировать код.

Для того чтобы упростить и ускорить работу разработчика, на рынке в свободном доступе присутствуют программные решения, которые берут на себя реализацию тех или иных аспектов бэкенд-компонента. В данном случае, для обеспечения функционала был выбран фреймворк Flask [3], т. к. в силу своего минимализма фреймворк легок в настройке и способен быстро обрабатывать поступающие запросы, а также гибкости при разработке добавляет большое количество расширений.

Модуль бэкенда, отвечающий за администрирование базы данных, реализован с помощью пакета Flask-SQLAlchemy [2], в котором есть необходимые компоненты для создания таблиц и связей. Для взаимодействия с данными посредством CRUD-операций используется предоставляемый пакетом интерфейс, который абстрагирует работу с базой

данных таким образом, что разработчику не приходится переключаться между написанием SQL-запросов и кода на Python.

Сервис предоставляет RESTful API, то есть внешний интерфейс доступный посредством HTTP-протокола. Взаимодействие осуществляется за счет использования четырёх HTTP методов: GET – получение данных без возможности редактирования, POST – передача новых данных для добавления в базу, PATCH – изменение данных из базы за счет передачи измененных полей, DELETE – удаление данных из базы. REST API представляет из себя список эндпоинтов, то есть URL адресов для обращения к бэкенду, Flask позволяет использовать декораторы для их описания. То есть для определения того, какая функция программы будет вызвана при обработке поступившего запроса, достаточно написать всего одну строчку кода с указанием того, по какому URL данная функция доступна и какой метод реализует (GET, POST и т. д.).

В качестве формата данных для API был выбран JSON, поскольку это простой и удобный формат, поддерживаемый основными используемыми фреймворками.

В базе данных хранятся некоторые персональные данные пользователей и для их защиты введено разделение доступа по двум признакам: факт владения, то есть принадлежит ли информация данному пользователю, и на основе причастности к группе пользователей. Существует 3 группы пользователей: рядовой пользователь, который имеет доступ к редактированию только своего профиля; модератор, который обладает теми же правами, что и рядовой пользователь относительно персональных данных; администратор, имеющий доступ к редактированию данных других пользователей. По умолчанию, все пользователи – рядовые, добавить пользователя в другую группу может только администратор. Для определения права доступа клиент сервиса предоставляет токен авторизации в заголовке запроса.

Клиентское приложение подразумевает создание чатов между пользователями и для обеспечения быстрого обмена сообщениями необходима реализация передачи данных в реальном времени на основе Socket.IO. В этом помогает компонент Flask-SocketIO [1], который добавляет возможность создания и запуска событий на стороне сервера.

В ходе работы был реализован бэкенд-сервис смешанного вида, включающий в себя как реализацию REST интерфейса, так и использование сокетов для взаимодействия в реальном времени. Всего был написан 31 эндпоинт и подготовлено для инициализации 2 события для сокетов. Вышеуказанные компоненты формируют внешний интерфейс бэкенд-сервиса, с которым уже сейчас работает клиентское веб-приложение. На данном этапе для окончания работы над сервисом осталось сформировать достаточный набор тестов, которые помогут определить, что данная разработка не только проста с точки зрения написания кода, но и эффективная с точки зрения производительности.

ЛИТЕРАТУРА

1. Документация Flask-SocketIO [Электронный ресурс] URL: https://flask-socketio.readthedocs.io/en/latest/

2. Документация Flask-SQLAlchemy [Электронный ресурс] URL: https://flask-sqlalchemy.palletsprojects.com/en/2.x/

3. Документация фреймворка Flask [Электронный ресурс] URL: https://flask.palletsprojects.com/en/1.1.x/

4. Описание REST архитектуры [Электронный ресурс] URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

УДК 681.3.06

Воскресенский А.С. (4 курс бакалавриата)
Семёнова-Тян-Шанская В.А., к.т.н., доцент

ИСПОЛЬЗОВАНИЕ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ ПРИ ЗАЩИТЕ ОТ СКЛИКИВАНИЯ

Хорошо известно, что любой, кто рекламируется в интернете, будет заинтересован, чтобы потенциальным покупателям показывалась именно его реклама, а реклама конкурентов «всплывала» реже или и вовсе не показывалась. Скликивание (clickfraud) - намеренное увеличение нецелевых переходов по рекламе в интернете. Начиналось такое мошенничество с создания простейших ботов, чаще всего, бизнесменами при использовании ресурсов собственной фирмы. Лишь позднее появились специальные агентства, профессионально предоставляющие услуги скликивания. В настоящее время рекламные площадки делают небольшие шаги к борьбе против кликеров, но их методы позволяют определить только самых «топорных» ботов, не использующих средства имитации поведения реального человека. Такая защита неэффективна, так как может не срабатывать целые сутки, за которые скликанная реклама не будет показываться.

В данной работе предложен новый подход к идентификации ботов - кластеризация сессий (визитов на сайт) пользователей на бот-сессии и сессии людей при помощи алгоритмов машинного обучения. Существуют большое количество интернет-сервисов, позволяющих собрать много обезличенной информации о сессии (например, Google Analytics, Matomo/Piwik). Их скрипты устанавливаются на клиентском приложении и собирают различные сведения об ОС, браузере, поведенческие характеристики, страну, регион и город захода, название провайдера, подсеть и т.д. Таким образом, можно собрать достаточно большой объем данных для применения машинного обучения.

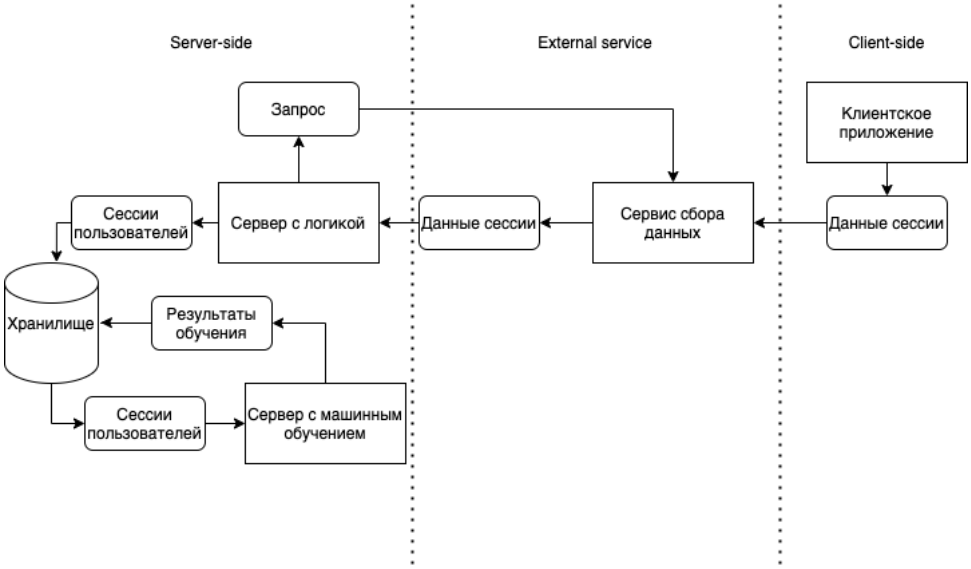


Рисунок 1 – Схема сбора и обработки данных

В работе был использован интернет-сервис сбора данных Matomo. Он оказался наиболее удобным, так как бесплатно предоставляет доступ к необходимым в данной работе инструментам и не использует полученные данные пользователей в своих целях. Скрипт устанавливался на страницу через Google Tag Manager (GTM), так как его код удобно встраивается в сайты, сделанные при помощи популярных CMS (WordPress, Wix, Tilda и другие).

Чтобы найти факты для машинного обучения, можно, например, выделить пользователей, решивших капчу, и считать их людьми, а пользователей, появившихся при заказе скликивания, считать ботами. Также в работе в качестве одного из определяющих

параметров «хорошей» сессии использовался анализ конверсий (покупок) при помощи API Google Ads и Яндекс.Директ. А для определения ботов, использовалась проверка наличия признаков Webdriver, управляющего браузером.

Конечно, не все боты используют Webdriver и не все люди генерируют конверсии, но поставленная цель - поиск похожих, а не идентичных сессий. Из этих соображений решающие факты не подаются на вход алгоритмам ML, они используются лишь для их обучения. Входные данные сессий, на основе которых алгоритм выносит решение, не несут информации о «порядочности» пользователей, а лишь передают некоторые их черты. Анализ этих черт помогает найти ботов.

Из большого набора современных библиотек для машинного обучения была выбрана ML.Net, так как она является частью экосистемы .Net, на основе которой написан сервер обработки данных, и заявлена как одна из лучших библиотек такого назначения. В работе используется перцептрон как алгоритм, хорошо подходящий для целей кластеризации. Архитектура нейронной сети содержит группирование входных нейронов по признакам и по группам признаков и отдельный анализ групп в слоях, близких к входному.

Вряд ли возможно создать алгоритм, который будет вечно без сбоев отлавливать ботов: их методы и подходы к скликиванию постоянно эволюционируют и всегда находятся слабые места, которыми они могут воспользоваться. Единственный выход - симметрично развивать механизмы защиты и следить за трендами web-разработки, работая на опережение.

ЛИТЕРАТУРА

1, Кульгин М. Что такое скликивание рекламы и как защититься от скликивания? // vc.ru 2020 https://vc.ru/marketing/157249-chto-takoe-sklikivanie-reklamy-i-kak-zashchititsya-ot-sklikivaniya
2. Quintanilla L., Olprod How to choose an ML.NET algorithm // docs.microsoft.com 2019 https://docs.microsoft.com/ru-ru/dotnet/machine-learning/how-to-choose-an-ml-net-algorithm

УДК 004.457

Голубева К.А. (4 курс бакалавриата)
Самочадин А.В., к.т.н., доцент

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ КОНТРОЛЯ ПОСЕЩАЕМОСТИ С ИСПОЛЬЗОВАНИЕМ МОДУЛЯ BLUETOOTH

Целью данной работы является создание системы, позволяющей отслеживать посещаемость учащихся школ, студентов, а также посетителей различных мероприятий и удовлетворяющей условию минимизации количества затраченного времени и действий со стороны человека. Приложение разработано для мобильных телефонов, для корректной работы должно быть установлено как у ведущего участника, так и у остальных. Все пользователи должны быть зарегистрированы в корпоративной системе.

В качестве операционной системы была выбрана Android sdk. Проект реализован с использованием объектно-ориентированного языка программирования Java. На рис.1 представлена схема приложения:

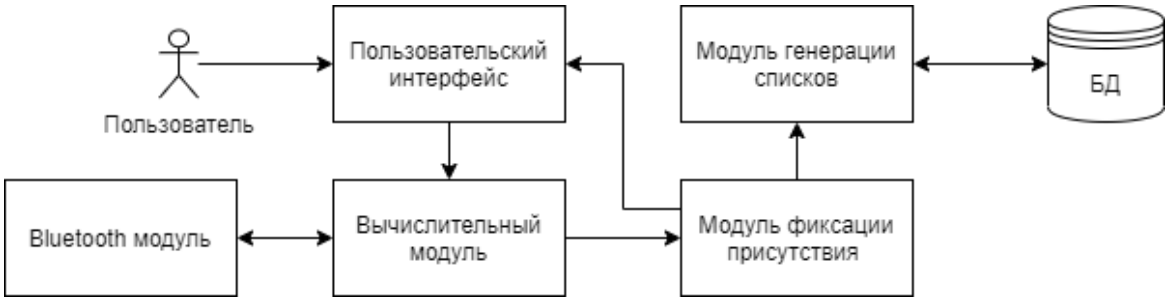


Рисунок 1 – Схема приложения.

Рассмотрим подробнее компоненты схемы приложения:

В пользовательском интерфейсе присутствует возможность включить или выключить Bluetooth, сделать устройство видимым для других, составить список присутствующих, отобразить ранее или только что сохраненный список присутствующих, а также выполнить экспорт списка. Интерфейс реализован с использованием внутренних средств языка программирования Java и языка разметки XML.

Для программной реализации Bluetooth модуля был использован пакет android.bluetooth (классы BluetoothAdapter и BluetoothDevice).

В вычислительном модуле осуществляется измерение силы сигнала Bluetooth (dBm) с использованием константы EXTRA_RSSI класса BluetoothDevice, которая содержит значение RSSI устройства-источника. Полученный результат переводится в системные единицы: метры, с допустимой погрешностью.

В модуле фиксации присутствия на основе полученных данных из вычислительного модуля устанавливается факт присутствия пользователя в условиях заданных характеристик. Пользователь получает push-уведомление о регистрации в списке присутствующих. Уведомление создается при помощи системного сервиса Android NotificationManager. Генерация соответствующего списка производится в одноименном модуле. Доступные форматы: png, jpeg или xls.

Каждый сгенерированный список сохраняется в базе данных, реализованной с использованием языка SQL при помощи системы управления MySQL.

Мною был выбран подход на основе Bluetooth, так как он обеспечивает минимизацию затрат времени как ведущего, так и остальных участников.

В характеристиках Bluetooth указано, что диапазон действия равен 10 метрам [1]. Также диапазон может варьироваться в зависимости от наличия препятствий таких как тело человека, стена, металл [2]. Чтобы уточнить влияние препятствий на силу сигнала Bluetooth были проведены различные эксперименты, в которых в качестве устройства-источника и устройства-приемника были использованы 2 телефона, на каждом из которых была установлена тестовая версия приложения:

Справа представлен график зависимости погрешности от расстояния (присутствует препятствие в виде стены), слева представлен график зависимости погрешности от расстояния и количества людей между устройствами (Желтая кривая – нет людей между устройствами, синяя кривая – нет людей между устройствами, но устройство-источник находится в переднем кармане брюк человека, зеленая кривая – 1 человек между устройствами, черная кривая – 2 человека, оранжевая кривая – 3 человека):

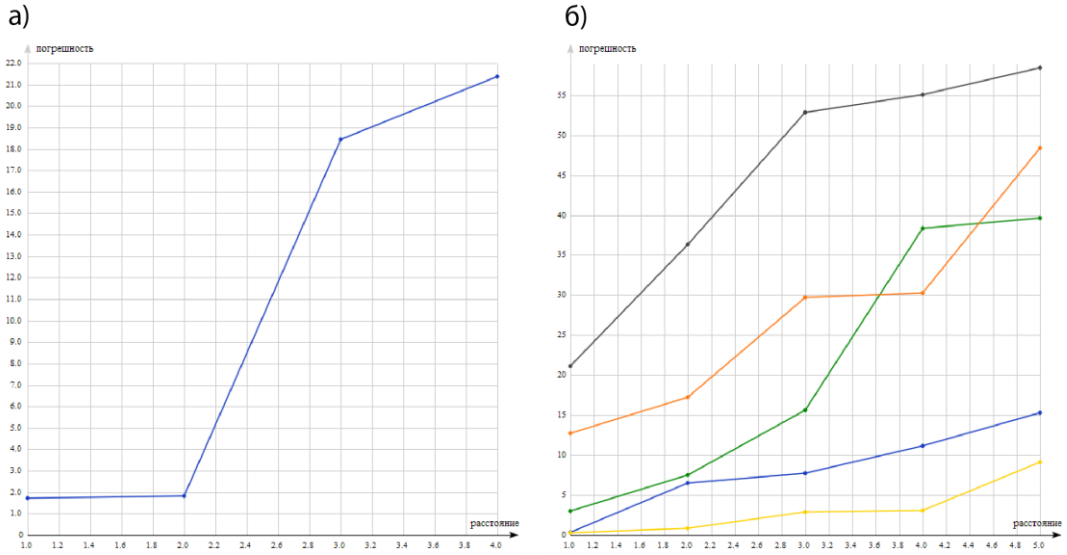


Рисунок 2 – График зависимости погрешности от расстояния с препятствием в виде стены (а) и график зависимости погрешности от расстояния и количества людей между устройствами (б).

Из полученных результатов можно сделать вывод, что система, в которой используются только мобильные телефоны (у ведущего и у участников) может быть использована для небольших помещений (с линейными размерами меньше 10 метров), а для больших помещений необходимы специальные устройства-маяки.

ЛИТЕРАТУРА

1. The range of Bluetooth and how can it be extended [Электронный ресурс] / URL: https://www.scienceabc.com/innovation/what-is-the-range-of-bluetooth-and-how-can-it-be-extended.html (Дата обращения 14.03.2020)
2. Maximum communication range of the Bluetooth connection [Электронный ресурс] / URL: https://www.sony.com/electronics/support/articles/00014085 (Дата обращения 14.03.2020)

УДК 004.42

Доленко Д.В., Шарапов А.А. (4 курс бакалавриата)
Коликова Т.В., ст. преподаватель

РАЗРАБОТКА ПРОГРАММЫ ДЛЯ СИНХРОНИЗАЦИИ КОНТЕНТА МЕЖДУ УСТРОЙСТВАМИ В РАМКАХ ЛОКАЛЬНОЙ СЕТИ

Сейчас проблема взаимной интеграции платформ является весьма актуальной: обычные пользователи хотят, чтобы все их устройства работали как единое целое. Для решения данной проблемы разработчики стали создавать экосистемы. Основная задача экосистемы - без лишних сложностей дать обычному пользователю возможность объединить свои устройства. Например, иметь общий буфер обмена между телефоном и ПК, общую файловую систему и т.д. Весьма распространёнными являются проприетарные решения, например, Apple Continuity или Samsung Flow. Однако главным их недостатком является то, что они доступны на очень ограниченном числе устройств: Apple Continuity работает только с устройствами Apple, а Samsung Flow только между телефонами и ноутбуками Samsung.

Помимо проприетарных решений есть и свободные, которые доступны на большинстве устройств и позволяют интегрировать устройства разных компаний. Самым известным и распространённым решением является KDE Connect [1].

Было проведено сравнение с этими решениями. Проприетарные решения доступны на ограниченном числе устройств, а решение с открытым исходным кодом KDE Connect имеет два недостатка:

– KDE Connect распространяется под лицензией GPL [2]. Главным ограничением GPL лицензии является то, что все производные программы должны выпускаться под той же лицензией. Это означает, что любые изменения, вносимые в данный продукт, а также модули, созданные для данного продукта, должны иметь открытый исходный код.
– KDE Connect использует централизованную систему внесения изменений. Перед добавлением изменений необходимо, чтобы ответственный за проект, утвердил изменения и выпустил новую версию программы, которая будет иметь предложенные изменения. Стоить отметить, что все внесённые изменения будут также распространяться под лицензией GPL.

Разрабатываемый продукт будет распространяться под лицензией MIT [3], которая позволяет сторонним разработчикам создавать модули под другой лицензией. Это может быть полезно для разработчиков, которые не хотят, чтобы исходный код был в свободном доступе, а также для тех, кто хочет иметь возможность коммерциализировать свой модуль.

В разрабатываемом продукте плагины являются независимо компилируемыми программными модулями, которые динамически подключаются к основной программе. Поэтому у разработчиков нет необходимости согласовывать изменения.

В системе присутствуют следующие компоненты:

– Application - основной модуль, который отвечает за связь остальных модулей.

– Plugin manager (менеджер плагинов) - отвечает за подключение плагинов к основной системе, предоставляет возможность плагинам подписываться на события, обеспечивает взаимодействие плагинов между собой. Плагины являются независимо компилируемыми программными модулями, которые динамически подключаются к основной программе. Плагин представляет собой динамическую библиотеку.

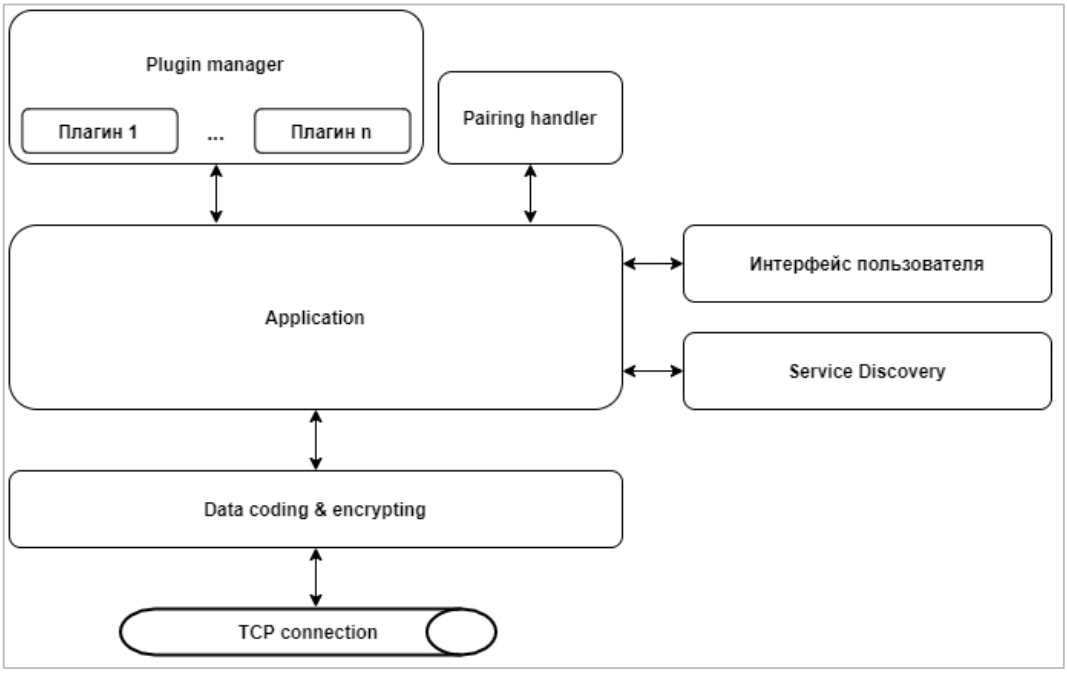


Рисунок 1 – Архитектура системы

– Service discovery – регистрирует сервис в локальной сети, осуществляет поиск других сервисов, определяет IP и порт по имени сервиса с помощью набора технологий Zeroconf [5] (Avahi в случае десктопного приложения, NSD – в случае Android приложения).

– Pairing handler (обработчик сопряжений) - отвечает за создание сопряжения двух устройств и сохранение данных для этого сопряжения. Таким образом образуется пара, для которой применяется автоматическое подключение одного устройства к другому, когда они оказываются в одной локальной сети.

– Data coding & encrypting - отвечает за кодирование при отправке и декодировании при получении данных. Используется протокол безопасной передачи данных TLS [4] с применением самоподписанного сертификата.

ЛИТЕРАТУРА

1. Главная страница проекта KDE Connect [Электронный ресурс]. – Режим доступа: https://kdeconnect.kde.org/
2. GNU General Public License [Электронный ресурс]. – Режим доступа: https://opensource.org/licenses/GPL-3.0
3. The MIT License [Электронный ресурс]. – Режим доступа: https://opensource.org/licenses/MIT
4. Transport Security Layer (TLS) [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Transport_Layer_Security
5. Zero Configuration Networking (Zeroconf) [Электронный ресурс]. – Режим доступа: http://www.zeroconf.org/

УДК 004.4

Егоров М.Э. (4 курс бакалавриата)
Леонтьева Т.В., к.т.н., доцент

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ЗВОНКОВ ПОД IOS НА ОСНОВЕ WEBRTC

Целью работы является исследование протокола WebRTC для разработки приложения для онлайн-звонков под iOS. С помощью этого приложения два пользователя смогут связаться между собой при наличии сети Интернет.

WebRTC — это протокол, который описывает передачу потоковых аудио и видео данных в режиме реального времени. Он работает как с UDP, так и с TCP и может переключаться между ними. Один из главных плюсов этого протокола — это возможность соединять пользователей с помощью p2p (peer to peer) соединения, это значит, что медиа-поток передается напрямую, минуя сервера. Но этот протокол не умеет передавать данные о соединении. Два пользователя не смогут передать друг другу медиа потоки, если не получат нужные данные для соединения друг от друга каким-то другим способом.

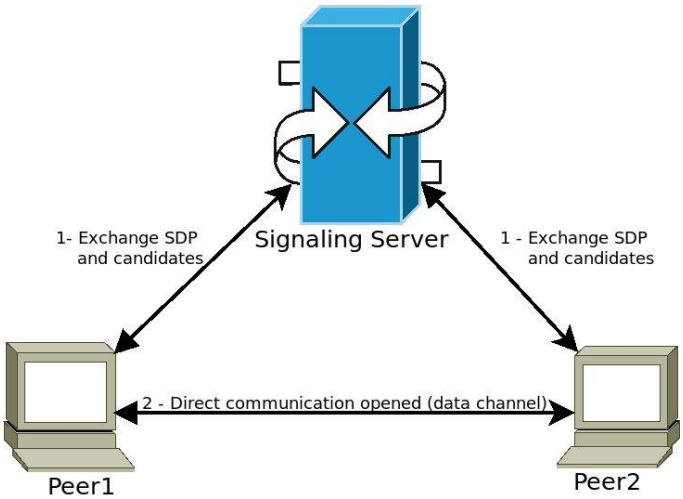


Рисунок 1 – Схема взаимодействия пользователей

Особенности реализации приложения:

Для установки соединения и передачи данных между клиентами используются вебсокеты. Для разработки приложения был использован язык программирования Swift 5. Это язык, с помощью которого можно писать приложения для телефонов, десктопных компьютеров, серверов. [1] Он был разработан компанией Apple как замена достаточно старому языку Objective-C. Swift уже имеет за плечами годы развития, и он не стоит на месте, а включает в себя всё новые и новые возможности.

Для работы с протоколом WebRTC использовался фреймворк WebRTC для iOS. Этот фреймворк целиком разработан на Objective-C, но Swift поддерживает runtime Objective-C. Это позволяет работать с этим фреймворком из Swift среды. В качестве среды разработки был выбран Xcode. Xcode - интегрированная среда разработки (IDE) программного обеспечения для платформ macOS, iOS, watchOS и tvOS. [2]

В итоге получаем приложение, которое работает примерно так: имеем двух клиентов (инициатор звонка и ожидающий звонка). Один из клиентов инициирует звонок и в этот момент он отправляет свой SDP через сигнальный сервер. SDP – дескриптор сессии, который содержит важную для установления соединения информацию о клиенте (набор кодеков и т.д.). Клиент, который принимает звонок, в свою очередь формирует ответный SDP и передает его обратно. Далее клиенты устанавливают физическое соединение путем обмена Ice Candidate. Объекты Ice Candidate содержат информацию о нахождении клиента в сети, с помощью Ice

Candidate клиенты смогут найти друг друга и начать передавать медиа-поток. После этого устанавливается соединение, и пользователи могут общаться друг с другом по видеосвязи.

Вывод: с помощью WebRTC можно разрабатывать приложения для звонков, видеоконференций и тд. Протокол достаточно быстро позволяет устанавливать соединение, что очень важно для онлайн звонков. Он не требует затрат для передачи медиа-потоков, так как делает это напрямую. Однако, существует очень мало информации о нем, и это затрудняет работу.

ЛИТЕРАТУРА

1. https://swift.org/about/
2. https://developer.apple.com/xcode/

УДК 004.414.38

Гусев И. С. (4 курс бакалавриата)
Коликова Т. В., ст. преподаватель

РАЗРАБОТКА МЕССЕНДЖЕРА С ВОЗМОЖНОСТЬЮ ОБМЕНА ГЕОПОЗИЦИЕЙ НА БАЗЕ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ

Целью работы является разработка мессенджера, который, помимо обычных функций обмена сообщениями, имеет возможность обмениваться геопозицией между собеседниками. Этот проект будет полезен людям, которым важно отслеживать месторасположение других людей, например, родители могут узнать, где сейчас находится их ребенок.

Сервер мессенджера будет использовать микросервисную архитектуру и REST Api. Основные преимущества микросервисной архитектуры:

– Автономность и независимость отдельных частей системы. Каждый модуль системы может разрабатываться отдельно разными командами. Модули можно заменять в любое время

– Возможность использовать разные технологии на разных частях системы. Благодаря этому каждый отдельный модуль увеличивает свою эффективность, так как может использовать более подходящие инструменты.

– Простота масштабирования. При потребности можно масштабировать только отдельный модуль, а не всю систему.

– Устойчивость. Учитывая, что элементы системы независимы друг от друга, в случае выхода из строя одного из модулей, остальная система продолжает работать.

Определение месторасположения осуществляется с помощью GPS Tracker и Google Maps Api. Серверная часть проекта будет разрабатываться на языке Java с использованием фреймворка Spring. Сервер будет сохранять данные пользователя, историю переписок и файлы пользователей в базе данных. В качестве основного хранилища данных выбрана свободная объектно-реляционная СУБД PostgreSQL. Передача данных между мобильным приложением и сервером будет осуществляться с помощью Json.

Пользователи смогут пользоваться сервисом с помощью мобильного приложения. Оно будет разработано на языке Kotlin. Дизайн приложения будет создан в соответствии с требованиями стандарта Material Design. В нем пользователи смогут авторизоваться, находить других пользователей по их контактам, создавать чаты, в том числе групповые, обмениваться как текстовой информацией, так и документами и медиафайлами. История переписки будет сохраняться, будет доступна функция поиска сообщений. Также можно будет запрашивать месторасположение других людей и видеть его на карте.

В качестве основных IDE будут использоваться Intellij IDEA для серверной части и Android studio для мобильного приложения.

ЛИТЕРАТУРА

1. Сэм Ньюмен Создание микросервисов. - СПб: Питер, 2016.
2. Микросервисы (Microservices) // habr.com URL: https://habr.com/ru/post/249183/ (дата обращения: 15.03.2021).

УДК 004.416.2

Ерохин Я.Д. (4 курс бакалавриата)
Дробинцев П.Д., к.т.н., доцент

СРАВНИТЕЛЬНЫЙ АНАЛИЗ БАЗ ДАННЫХ ДЛЯ ВЕДЕНИЯ ЖУРНАЛА УСТРОЙСТВ "ИНТЕРНЕТА ВЕЩЕЙ" В ОБЛАЧНО-ОРИЕНТИРОВАННОЙ СРЕДЕ

В этой обзорной статье мы рассматриваем различные opensource базы данных временных рядов для ведения журналов, сравниваем результаты из других источников, объединяем их и формулируем набор рекомендаций, которые можно использовать для ведения журнала в облачной среде.

Инструменты логирования относятся к фазе Operations в статье DevOps [1]. Ведение журнала иногда считают заменой отладки, и программистам рекомендуется выполнять отладку вместо журнала, чтобы они не оставляли следов в приложении. Но с точки зрения DevOps трассировки необходимы для управления приложениями. Правило состоит в том, чтобы создавать постоянные записи событий приложения. В дальнейшем файлы журнала можно использовать для просмотра любых событий в системе, включая сбои и преобразования состояний.

Рассмотрим одну важную характеристику облачно-ориентированных систем - наблюдаемость. Облачно-ориентированные приложения по своей природе труднее поддаются исследованию и отладке. Поэтому наблюдаемость является ключевым требованием к распределенным системам: мониторинг, ведение журнала, трассирование и сбор показателей помогают инженерам понять, чем занимаются их системы, и что они делают не так [2].

Сравнение инструментов.

В этом разделе выполняется сравнение продуктов: InfluxDB, TimescaleDB, Elasticsearch. Для сравнения инструментов выберем следующие критерии: удобство языка запросов к базе данных, является ли база данных распределенной, степень сжатия данных, скорость записи, скорость чтения, скорость выполнения запросов к базе данных, экосистема продукта и поддержка сообщества.

Удобство языка запросов. TimescaleDB имеет полную поддержку распространенного языка запросов SQL, что является проверенным годами решением. Множество разработчиков знают язык SQL, поэтому начать работу с TimescaleDB будет очень просто. InfluxDB имеет поддержку двух языков InfluxQL и Flux. Первый язык является SQL-подобным, поэтому является худшим решением, чем поддержка чистого SQL. Второй язык Flux является более интересным. Flux использует шаблоны функционального языка, что делает его невероятно мощным, гибким и способным преодолеть многие ограничения InfluxQL. Этот язык является более компактным и лаконичным, чем языки SQL, поэтому пользователю будет удобнее работать с ним. Но из-за того, что все его операции строго упорядочены, планировщик запроса не может оптимизировать ход выполнения запроса, поэтому тут встает выбор между скоростью и удобством пользователя [4]. В тоже время, если пользователь уже знаком с языком SQL, то подобный выбор становится намного очевиднее в сторону TimescaleDB. Что касается Elasticsearch, то там язык запросов реализован через JSON, что является далеко не самым удобным решением. Кроме того, HTTP API Elasticsearch допускает множество синтаксически допустимых запросов JSON. Это означает, что сервер может принять запрос с ошибкой, не сообщив об этом [3].

Распределенная база данных. Elasticsearch является распределенной базой данных [5]. InfluxDB и TimescaleDB тоже могут работать в режиме распределенной базы данных, но только на платной основе [6, 7].

Сжатие данных. Лучше всех сжимает данные InfluxDB, она в 9 раз лучше использует дисковое пространство, чем Elasticsearch и в 2 и более раз лучше, чем TimescaleDB [8, 9].

Скорость записи. Если данные записываются с небольшого количества устройств интернета вещей примерно от 1 до 1000, то InfluxDB является лучшим решением, так как на

23 процента быстрее записывает, чем TimescaleDB и в среднем в 3.8 раза быстрее записывает, чем Elasticsearch. Если же речь идет о большем количестве устройств интернета вещей от 1000, то лучшим вариантом будет TimescaleDB, так как в этом диапазоне запись выполняется быстрее в 3.5 раза, чем у InfluxDB [3, 8].

Скорость выполнения запросов. Для запросов агрегации нескольких метрик с устройств интернета вещей TimescaleDB лучше справляется примерно в 1.5 раз, чем InfluxDB. InfuxDB справляется лучше, чем Elasticsearch примерно в 7.7 раз лучше [3, 8].

Использование оперативной памяти. InfluxDB использует в разы больше оперативной памяти, чем TimescaleDB [13]. С увеличением нагрузки потребление памяти очень быстро растет.

Экосистема. TimescaleDB имеет очень хорошую экосистему, превосходящую InfluxDB, так как у TimescaleDB имеется большее количество сторонних продуктов, поддерживающих её официально [8]. У Elasticsearch экосистема тоже достаточно хорошая, так как продукт входит в Cloud Native Computing Foundation (CNCF) и имеет огромное количество различных инструментов для бесшовной интеграции вроде Logstash, Fluentd, Kibana и многие другие.

Поддержка компании и сообщества. Самая большая поддержка сообщества у Elastichsearch, так как продукт входит в CNCF. На втором месте стоит TimescaleDB, потому что они унаследовали поддержку сообщества PostgreSQL. InfluxDB создает свое сообщество буквально с нуля, поэтому является менее привлекательным в данном критерии [8].

Заключение.

Найти решение, которое является лучшим на рынке не удалось, поэтому нужно искать компромисс в зависимости от решаемой задачи. Сообщения, вносимые в базу данных, от устройств интернета вещей обычно не превышают в длину 100 - 200 символов, поэтому для такой задачи одинаково хорошо могут подойти все 3 продукта. Если смысловой поиск по тексту логов нам несильно нужен, то Elasticsearch можно пренебречь, так как база данных работает достаточно медленно и довольно сложна в настройке. Исключением может стать только случай, если нужна распределенная база данных на бесплатной основе. Если есть средства на распределенную версию InfluxDB и TimescaleDB, то стоит внимательнее посмотреть в их сторону. Если количество устройств интернета вещей, с которых мы хотим собирать логи, примерно равно 1000, то InfluxDB очень хорошо подходит, так как запись в базу данных проводится быстрее своих конкурентов. Если же устройств значительно больше 1000, появляется выбор: либо использовать экономно дисковое пространство с InfluxDB, либо использовать экономно оперативную память, быстрее писать в базу данных, быстрее выполнять запросы и иметь более продвинутую экосистему с TimescaleDB.

ЛИТЕРАТУРА

1. DevOps / C. Ebert [и др.] / IEEE Software. 2016. С. 94-100.
2. Arundel J., Domingus J. Cloud Native DevOps with Kubernetes. Sebastopol: O'Reilly, 2019. 315 с.
3. InfluxDB vs. Elasticsearch for Time Series Data & Metrics Benchmark // InfluxData. 2020. URL: https://get.influxdata.com/rs/972-GDU-533/images/InfluxDB_vs_Elasticsearch.pdf (дата обращения: 14.03.2021).
4. SQL vs. Flux: Choosing the right query language for time-series data // Timescale. 2018. URL: https://blog.timescale.com/blog/sql-vs-flux-influxdb-query-language-time-series-database-290977a01a8a/ (дата обращения: 14.03.2021).
5. Elasticsearch // Elastic. 2020. URL: https://www.elastic.co/elasticsearch/ (дата обращения: 14.03.2021).
6. InfluxDB Time Series Platform // InfluxData. URL: https://www.influxdata.com/products/influxdb/ (дата обращения: 14.03.2021).
7. TimescaleDB Products // Timescale. URL: https://www.timescale.com/products (дата обращения: 14.03.2021).
8. TimescaleDB vs. InfluxDB: Purpose built differently for time-series data // Timescale. URL: https://blog.timescale.com/blog/timescaledb-vs-influxdb-for-time-series-data-timescale-influx-sql-nosql-36489299877/ (дата обращения: 14.03.2021).
9. Высокопроизводительный TSDB benchmark VictoriaMetrics vs TimescaleDB vs InfluxDB // Хабр. 2020. URL: https://habr.com/ru/post/508362/ (дата обращения: 14.03.2021).

УДК 004.6

Засядько В.В. (4 курс бакалавриата)
Воинов Н.В., к.т.н., доцент
Прокофьев О.В., ст. преподаватель

ВЕБ-СЕРВИС ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕССОВ В ОТДЕЛЕ МАРКЕТИНГОВЫХ ИССЛЕДОВАНИЙ

В наше время компаниям приходится работать с огромным объемом информации, который продолжает расти ежедневно. Для успешной работы необходим качественный и эффективный анализ получаемых данных, так как результаты проведенного анализа имеют критическое влияние на принятие дальнейших решений [1-3]. Визуализация данных позволяет быстро и легко замечать и интерпретировать связи и взаимоотношения, а также выявлять различные тенденции, которые не привлекли бы внимания в виде необработанных данных [4,5]. В большинстве случаев для интерпретации графических представлений не требуется специальное обучение, что сокращает вероятность недопонимания.

Цель работы - создание инструмента для эффективного визуального представления информации для работы с данными опросов. Технически данный инструмент должен быть реализован в виде веб-сервиса и предоставлять широкие возможности по визуализации полученных данных.

К данному моменту в рамках работы проведены исследования в области современных эффективных технологий разработки веб-приложений: промышленного стека разработки (TypeScript, React, Effector) и методов проектирования архитектуры веб-приложений для выбранной задачи; исследование существующих систем и инструментов визуализации данных для выбора наиболее эффективных вариантов визуального представления данных и наиболее простого и интуитивного пользовательского интерфейса для взаимодействия с системой. Создан прототип архитектуры веб-сервиса. В ближайшее время планируется реализация веб-сервиса, его тестирование и анализ полученных результатов.

Результатом всей работы станет реализованное веб-приложение с функционалом для конструирования дашбордов для визуализации результатов различных опросов в наглядном и доступном для анализа данных виде.

ЛИТЕРАТУРА

1. В. А. Ивлев, И. В. Никифоров. Обработка данных в геоинформационных системах для выбора местоположения рекламы. (с. 27) / Современные технологии в теории и практике программирования: сборник материалов конференции, 19 апреля 2019 года / Санкт-Петербургский политехнический университет Петра Великого, [Институт компьютерных наук и технологий [и др.] ; Санкт-Петербург : [ПОЛИТЕХ-ПРЕСС], 2019. 247 с.: ил., табл. ; 20 см. ISBN 978-5-7422-6508-5.
2. Е. В. Тимощук, И. В. Никифоров, В. В. Шаляпин. Программная система поддержки принятия решений и оценки финансовых рисков в области транспортного лизинга. (с. 79) / Современные технологии в теории и практике программирования: сборник материалов конференции, 19 апреля 2019 года / Санкт-Петербургский политехнический университет Петра Великого, [Институт компьютерных наук и технологий [и др.] ; Санкт-Петербург : [ПОЛИТЕХ-ПРЕСС], 2019. 247 с.: ил., табл. ; 20 см. ISBN 978-5-7422-6508-5.
3. В. К. Сычев, Ю. В. Ленькова, Е. А. Цветкова, И. В. Никифоров. Анализ данных трендов Youtube. (с. 131) / Современные технологии в теории и практике программирования: сборник материалов конференции, 23 апреля 2020 года / Санкт-Петербургский политехнический университет Петра Великого, [Институт компьютерных наук и технологий [и др.] ; Санкт-Петербург : [ПОЛИТЕХ-ПРЕСС], 2020. 175 с. ISBN 978-5-7422-6935-9.
4. А. А. Перескокова, А. А. Кольцов, В. И.. Кочугова, С. С. Толпыгин, И. В. Никифоров. Архитектура программного аналитического комплекса исследования поведения пользователей на образовательной платформе "Открытое образование". (с. 82) / Материалы научной

конференции с международным участием "Неделя науки СПбПУ", 18-23 ноября 2019 года. Институт компьютерных наук и технологий / Санкт-Петербург : ПОЛИТЕХ-ПРЕСС, 2019. 144 с.: ил., табл. ; 20 см. ISBN 978-5-7422-6804-8.

5. Барсуков Н.Д., Сысоев И.М., Перескокова А.А., Никифоров И.В., Посметныйс Д. Анализ активности студентов на курсах онлайн-обучения на основе логов платформы «OpenEdu». Труды Института системного программирования РАН. 2020;32(3):91-99. https://doi.org/10

УДК 621.319

Козельский Е.И. (2 курс магистратуры)
Шаляпин В.В., к.т.н., доцент

ГЕНЕРАЦИЯ ИНФОРМАТИВНЫХ ВОПРОСОВ В ИНФОРМАЦИОННО-АСИММЕТРИЧНЫХ БЕСЕДАХ

Целью работы является изучение генерации вопросов для поиска информации в открытой коммуникации, которая включает в себя сложные проблемы *NLP, а* также разработка и обучение модели, которая может генерировать прагматически актуальные и конкретные вопросы для получения новой информации о невидимом источнике текстовых знаний.

Рассматриваются диалоговые системы, в которых спрашивающая сторона на имеет доступа к контексту, из которого получены ответы, но должна прагматично рассуждать о том, как получить новую информацию, учитывая общую историю разговоров. Основными проблемами являются формальное определение информированности потенциальных вопросов и изучение чрезмерно большого пространства потенциальных вопросов, чтобы найти хороших кандидатов. Для решения этих проблем предлагаются автоматические метрики для количественной оценки информативности и специфичности вопросов, которые необходимы для эффективной итеративной разработки системы.

Таблица 1 – Оценка качества сгенерированных вопросов

	Выигрыш	Связь	Потеря	Значение p
Сравнение с базовой оценкой				
Общая	29.0%	52.5%	18.5%	0.006
INFO	19.25%	74.5%	6.25%	0.00003
SPEC	25.5%	56.5%	18.0%	0.029
Сравнение с человеческой оценкой				
Общая	17.75%	26.25%	56.00%	<0.000002
INFO	8.75%	74.50%	19.75%	0.0003
SPEC	15.25%	29.00%	55.75%	<0.000002

В таблице показана оценка качества результатов работы системы с использованием различных метрик. Результаты подтверждают, что оптимизация предлагаемых показателей ведет себя прагматично и повышает эффективность коммуникации, что представляет собой практический метод прагматических рассуждений в условиях открытого общения. Чтобы генерировать прагматические вопросы, используется обучение с подкреплением для

оптимизации, предлагаемой в работе метрики информативности в сочетании с функцией вознаграждения, разработанной для продвижения более конкретных вопросов. Демонстрируется, что полученный прагматичный вопросник существенно улучшает информативность и конкретность вопросов, созданных на основе базовой модели, как оценивается разработанными метриками, так и людьми.

В работе представлена система генерации вопросов в информационных беседах. Оптимизируя предлагаемые автоматические метрики для информативности и специфичности, модель может генерировать прагматически актуальные и конкретные вопросы для получения новой информации о невидимом источнике текстовых знаний. Предлагаемый метод представляет собой практическую, хотя и поверхностную реализацию прагматики в условиях открытого общения за пределами простых эталонных игр.

ЛИТЕРАТУРА

1. Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1173–1182.
2. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations.
3. Ashutosh Baheti, Alan Ritter, Jiwei Li, and Bill Dolan. 2018. Generating more interesting responses in neural conversation models with distributional constraints. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing.
4. Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In International Conference on Learning Representations.

УДК 621.319

Кущенко А. Е. (2 курс магистратуры)
Самочадин А. В., к.т.н., доцент

РАЗРАБОТКА ИНСТРУМЕНТА ДЛЯ ИЗВЛЕЧЕНИЯ МОДЕЛЕЙ ПРОЦЕССА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Целью работы является разработка средств построения и анализа процесса разработки программного обеспечения на основе событий, инициированных разработчиками во время работы. В первую очередь рассматриваются события, которые оставляют след в различных информационных бизнес-системах, с которыми взаимодействуют пользователи в процессе разработки программного обеспечения. Предлагается извлечь модель поведения разработчиков используя журналы событий, порождённые ими при взаимодействии с различными информационными системами. Далее, на основе полученных данных выявить слабые места, как во всём процессе работы, так и поведении отдельно взятых пользователей. Полученная модель также может быть использована для обнаружения сотрудников, которые по тем или иным причинам не следуют требованиям к организации процесса разработки в компании.

Для построения модели используется технология Process Mining. На рисунке 1 схематично приведена схема применения технологии Process Mining в процессе разработке. Исходные данные о событиях, порождённые сотрудниками в бизнес-системах используются для формирования журнала событий, который в свою очередь поступает на вход формальных методов Process Mining. Process Mining позволяет решить задачу обнаружения модели процесса, проверки соответствия этой модели и последующему улучшению модели.

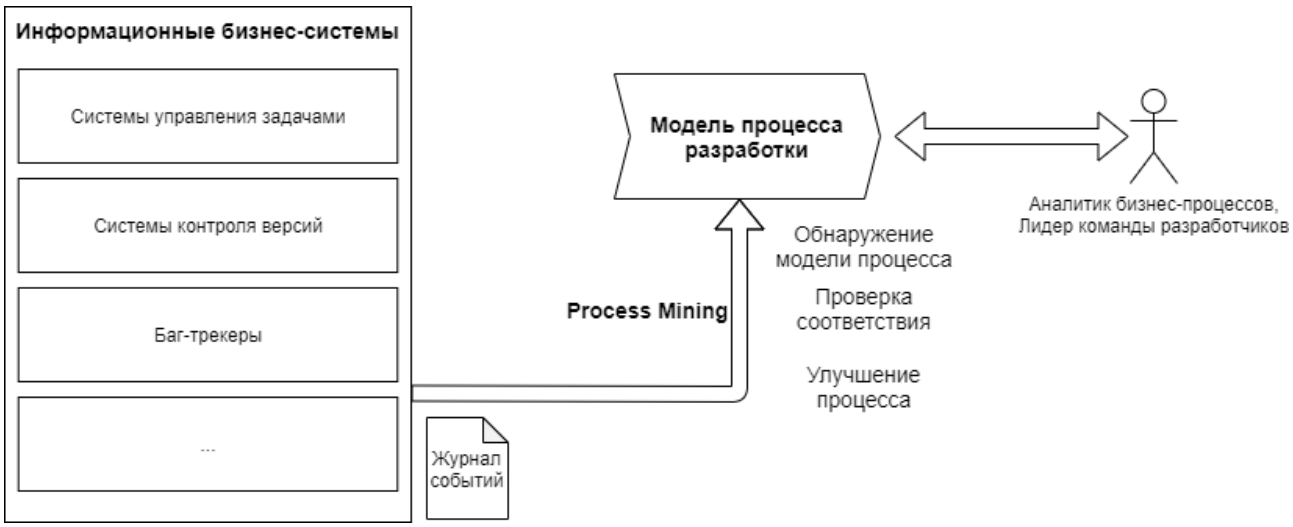


Рисунок 1 – Process Mining и процесс разработки ПО.

В разрабатываемых средствах источниками данных являются:

- Систему контроля версий GitLab;
- Системы управления задачами и баг-трекер Youtrack;
- Интернет-браузер Google Chrome.

Используемые технологии и инструменты:

В основе данной работы лежит технология Process Mining (интеллектуальный анализ процессов) – это область исследований, которая фокусируется на анализе процессов с использованием данных о событиях. В методах Process Mining журналы событий используются для решения трёх основных задач:

- Обнаружение – построение модели процесса на основе журнала событий.
- Проверка соответствия – сравнение простроенной модели с реальным поведением, описанном в журнале событий.
- Улучшение - расширение или улучшение существующей модели процесса.

Для решения задач Process Mining предлагается использовать наиболее известный инструмент с открытым исходным кодом ProM, который предоставляет широкие возможности. В ProM реализованы практически все основные инструменты Process Mining в виде плагинов, которых насчитывается более 200.

В качестве алгоритма для обнаружения модели процесса используется α-метод, реализованный в ProM как расширение “Alpha Miner”. Для интеллектуального анализа и улучшения процесса используется расширение “Mine for a Handover-of-Work Social Network” [2] для анализа производительности сотрудников и алгоритмы кластеризации, которые позволяют использовать информацию об исполнителях для обнаружения ролей, т. е. групп людей, часто выполняющих связанные действия. Кроме того, предлагается анализировать время ожидания между действиями, путём измерения разницы во времени между причинно-связанными событиями и вычисления статистик, таких как средние значения, дисперсии и доверительные интервалы. Таким образом можно определить основные узкие места [3].

Для извлечения исходных данных о событиях и построения готового журнала используется фреймворк Spring, предназначенный для разработки на языке Java.

Все вышеперечисленные бизнес-системы имеют открытый API, что позволяет запрашивать сущности, содержащие информацию об интересующих нас событиях. Для взаимодействия с API используется библиотека Webflux, включающая в себя класс WebClient, методы которого позволяют формировать HTTP запросы.

Извлечённые сущности сохраняются в СУБД Postgres. Для взаимодействия с сущностями (таблицами, столбцами) реляционной базы данных используется расширение Spring Data JPA. Для работы с базой данных описывается схема данных в виде обычных

классов на языке Java, где поля соответствуют столбцам в таблицах. Таким образом Spring Data JPA отображает объектную модель данных на реляционную.

Заключение

В этой статье рассмотрен подход для интеллектуального анализа процесса разработки ПО, основанный на информации, доступной в информационных бизнес-системах, с которыми взаимодействуют разработчики в ходе своей деятельности. Были описаны инструменты и упомянуты алгоритмы, которые позволяют решить задачи по извлечению данных о событиях, а затем построению и улучшению модели процесса разработки ПО.

ЛИТЕРАТУРА

1. XES Standart. [Электронный ресурс]. URL: http://www.xes-standard.org/
2. Mining the Control-Flow Perspective of a Process // ProM Tools URL: https://promtools.org/doku.php?id=tutorial:answers
3. AALST, W. VAN DER Process Mining: Discovery, Conformance and Enhancement of Business Processes // Springer-Verlag. - 2011.

УДК 004.415.2

Лебедев И.В. (1 курс магистратуры)
Медведев Б.М., к.т.н., доцент

РАЗРАБОТКА ДЕМОНСТРАЦИОННЫХ ПРОЕКТОВ ДЛЯ УСТРОЙСТВА ИНТЕРНЕТА ВЕЩЕЙ НА БАЗЕ ARDUINO

По данным Transforma Insights к 2030 году количество подключенных устройств интернета вещей (ИВ) составит 24,1 миллиарда [1], причем ожидаемый рост в основном будет в сегментах: смартфоны, устройства умных домов, а также устройства промышленного интернета вещей. Огромный объем рынка ИВ требует большого числа разработчиков программного обеспечения для создания встраиваемых программных средств устройств, облачных вычислений и приложений пользователя.

Одной из популярных платформ для разработки прототипов устройств ИВ является Arduino – платформа с открытым исходным кодом, основанная на простом в использовании аппаратном и программном обеспечении [2]. Основой модуля Arduino Uno является микроконтроллер ATmega328P, имеющий тактовую частоту 16 МГц, 32 КБ флэш памяти и 2 КБ ОЗУ. Модуль содержит контакты для подключения 14 цифровых входов/выходов и 6 аналоговых входов. Габариты печатной платы 6.9 x 5.3 см [3]. Для разработки встраиваемого программного обеспечения используется среда разработки Arduino IDE, имеющая версии для ОС Windows, Linux и MacOS. Для создания устройства ИВ требуется плата Arduino Uno, плата расширения Arduino MKR 1010 для подключения устройства к интернету, а также набор датчиков с цифровым или аналоговым интерфейсом, содержащий датчики движения, освещенности, температуры и влажности, датчики акустических и механических колебаний [4].

Целью работы является создание ряда демонстрационных проектов и сценариев выполнения курсовой работы с использованием Arduino в рамках практических занятий по курсу «Интернет вещей».

Программное обеспечение Arduino состоит из следующих компонентов: Arduino IDE, API Arduino IoT, библиотека для передачи данных по протоколу MQTT, библиотеки для управления датчиками и обработки сигналов датчиков, UTFT для работы с цветными дисплеями, стек TCP/IP и библиотека UIPEthernet [5, 6].

В качестве облачного сервиса в работе используется платформа IBM Watson IoT, предоставляющая обширный набор инструментов для обработки данных, полученных от

устройств [7]. IBM Cloud предоставляет 40 различных сервисов, доступных в бесплатной подписке. Демонстрационные проекты используют платформу Интернета вещей (IBM IoT Platform). При бесплатном тарифном плане присутствует ряд ограничений: до 500 зарегистрированных устройств и не более 200 МБ данных. Для того, чтобы расширить возможности использования данной платформы разрабатывается приложение на базе IBM Bluemix с использованием сервисов машинного обучения, хранилища объектов, перевода речи в текст и понимания естественного языка [8].

Разработанные сценарии выполнения курсовой работы с использованием Arduino содержат индивидуальные траектории обучения. Работа разделена на два этапа. В рамках первого этапа каждый студент выполняет демонстрационный проект, включая создание аккаунта в IBM Cloud, создание устройства на базе Arduino Uno, программирование и подключение полученного устройства к платформе IBM Cloud по инструкции.

На втором этапе разрабатывается индивидуальное задание и выполняется разработка программных средств по одной из следующих тем:

- разработка встраиваемого программного обеспечения Arduino для обработки сигналов датчиков движения, освещения, температуры;
- разработка программных средств обработки сообщений в IBM Cloud, таких, как отправка уведомления на смартфон или компьютер по триггеру;
- разработка приложения пользователя, обеспечивающее контроль состояния и управление устройством ИВ [9, 10].

Предлагаемые для реализации задания содержат оценку сложности по 10 бальной шкале.

ЛИТЕРАТУРА

1. IoT in 2030: A tri-polar world, with significant regional variations/ IoTNOW [Электронный ресурс]. – URL: https://www.iot-now.com/2020/05/21/102941-iot-in-2030-a-tri-polar-world-with-significant-regional-variations/ (дата обращения: 15.03.2021).
2. What is Arduino/ Arduino [Электронный ресурс]. – URL: https://www.arduino.cc/en/Guide/Introduction (дата обращения: 15.03.2021).
3. Arduino uno rev3/ Arduino [Электронный ресурс]. – URL: https://store.arduino.cc/usa/arduino-uno-rev3 (дата обращения: 15.03.2021).
4. Arduino MKR IoT Bundle/ Arduino [Электронный ресурс]. – URL: https://store.arduino.cc/usa/arduino-iot-mkr1000-bundle (дата обращения: 15.03.2021).
5. MQTT Client/ Arduino [Электронный ресурс]. – URL: https://www.arduino.cc/reference/en/libraries/mqtt-client/ (дата обращения: 15.03.2021).
6. Arduino IoT Cloud API/ Arduino [Электронный ресурс]. – URL: https://www.arduino.cc/reference/en/iot/api/ (дата обращения: 15.03.2021).
7. Интернет вещей в IBM Cloud/ IBM [Электронный ресурс]. – URL: https://www.ibm.com/ru-ru/cloud/internet-of-things (дата обращения: 15.03.2021).
8. IBM Cloud/ IBM [Электронный ресурс]. – URL: https://cloud.ibm.com/catalog (дата обращения: 15.03.2021).
9. 3 Sketches for Connecting Arduino ESP32 to IBM Watson IoT/ Instructable circuits [Электронный ресурс]. – URL: https://www.instructables.com/3-Sketches-for-Connecting-Arduino-ESP32-to-IBM-Wat/ (дата обращения: 15.03.2021).
10. IBM Watson Iot Platform With MQTT/ Arduino [Электронный ресурс]. – URL: https://forum.arduino.cc/index.php?topic=679770.0 (дата обращения: 15.03.2021).

УДК 004.42

Лысенко В.В., Сташевский Я. В. (4 курс бакалавриата)
Петров А.В., ст. преподаватель

ПРОТОТИП TELEGRAM-BOT POLYSCHEDULE

В наше время все учащиеся высших учебных заведений регулярно пользуются расписанием, хотя оно и выполняет базовые функции, но потребности студентов на этом не ограничиваются. На рынке бесплатного программного обеспечения существуют разные подходы к решению потребностей студентов, но такого, чтобы включало бы и ежедневные уведомления, и возможность указать домашнюю работу, и информацию о погоде на территории университета, решения найдено не было.

Целью работы является исследование возможностей Telegram Bot API[1]. Данное программное обеспечение предназначено для автоматического уведомления пользователей о предстоящих учебных мероприятиях. Бот для мессенджера Telegram[2] носит название “PolySchedule”. В его задачу входит уведомление пользователей каждый вечер о расписании занятий на следующий день. В качестве дополнительных функций, приложение уведомляет пользователя о погоде и новостях утром каждого дня, а также позволяет получить доступ к домашним заданиям, введенным другими пользователями, состоящими в одинаковой группе. Данный бот может использоваться студентами в качестве дополнительного напоминания о грядущих занятиях с целью снижения вероятности их пропуска.

Был проведен анализ существующих решений, результат которого продемонстрирован в таблице 1. Решений доступных для получения расписания не много, и все они не имеют необходимый функционал, такой как вывод погоды на территории СПбГПУ, ежедневные уведомления расписания, возможность указать домашнюю работу для всей группы. Из-за чего было принято решение о создании такого проекта, который имеет требуемый нам функционал.

Таблица 1 – сравнение существующих решений

	Просмотр расписания	Информация о погоде на территории СПбГПУ.	Возможность указать домашнюю работу для всей группы.	Ежедневные уведомления расписания.	Наличие расписания СПбГПУ	Поддерживаемые платформы
Расписание занятий| СПбГПУ @PolyRobot	Да	Нет	Нет	Нет	Да	Android, iOS, Windows Phone, Windows, macOS и GNU/Linux на базе Telegram.
Кампус Расписание занятий	Да	Нет	Да	Нет	Нет	Android, iOS.
Расписание занятий Sked	Да	Нет	Да	Да	Да	Android, iOS.

Для выполнения проекта был выбран язык программирования Python[3]. В качестве среды разработки использовался PyCharm[4]. Был использован модуль pyTelegramBotAPI. Его ключевая особенность - полная поддержка всего функционала Telegram Bot API, который является интерфейсом, основанным на HTTP протоколе. Для хранения данных была

использована документоориентированная система управления нереляционными базами данных MongoDB[5]. Для организации более эффективного решения задач проекта была использована система Trello. Архитектуру проекта было решено разделить на 4 модуля: модуль базы данных (dataBase) – отвечает за хранение данных студентов, модуль пользовательского интерфейса (TgBotUI) – отвечает за пользовательский интерфейс и является связующим звеном проекта, модуль предоставляющий расписание (ScheduleProvider), модуль предоставляющий доступ к API Яндекс.Погоды (yandexApiAdapter). На рисунке 1 изображено графическое представление связи модулей в архитектуре проекта.

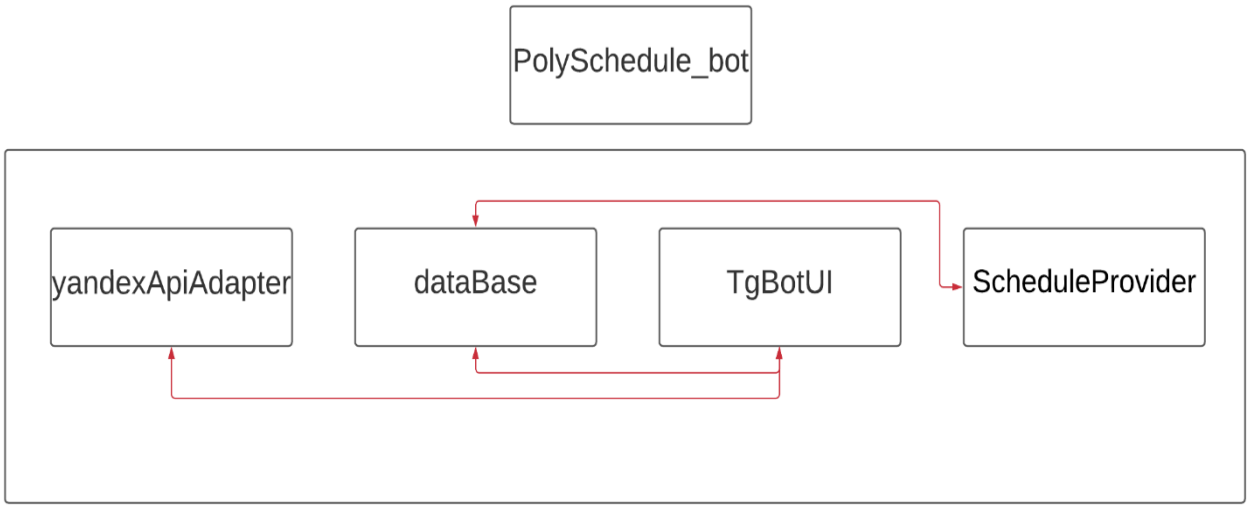


Рисунок 1 – Графическое представление связи модулей в архитектуре проекта

Рассмотрим немного подробнее тему хранения данных. При помощи пакета PyMongo осуществлялась работа с базой данных. В базе данных есть 4 коллекции документов:

1. Статус пользователя – в этой коллекции хранится информация о пользователе взаимодействующем с ботом в виде логина, шага команды, названия и аргументов команд.

2. Студенты – в этой коллекции хранятся документы о списке студентов и их группах.

3. Расписание – в этой коллекции хранятся сведения о группе, посещающей данное занятие, дате, времени, названии и типе занятия.

4. Домашняя работа – в этой коллекции хранятся домашние задания, с указанием группы, даты и названия занятия. Так как для каждой коллекции необходим свой способ управления данными, были написаны 4 подмодуля и 1 подмодуль для переноса базы данных на другую машину.

ЛИТЕРАТУРА

1. Telegram-Based Chatbot Application for Foreign People in Japan to Share Disaster-Related Information in Real-Time. [Электронный ресурс]. - Режим доступа: www. URL: https://ieeexplore.ieee.org/abstract/document/9118510/keywords#keywords - 10.03.2021.

2. Telegram F.A.Q. [Электронный ресурс]. - Режим доступа: www. URL: https://telegram.org/faq/ 10.03.2021.

3. Рамальо Л. PYTHON. К ВЕРШИНАМ МАСТЕРСТВА. М.: ДМК-Пресс, 2016 г.

4. Узнайте все о PyCharm. [Электронный ресурс]. - Режим доступа: www. URL: https://www.jetbrains.com/ru-ru/pycharm/learn/ - 10.03.2021.

5. Бэнкер Кайл MongoDB в действии. М.: ДМК-Пресс, 2017.

УДК 575.112

Марченко В.А., Хватов С.А. (4 курс бакалавриата)
Воскобойников С.П., к.ф.м.н., доцент

РАЗРАБОТКА И ОПТИМИЗАЦИЯ АЛГОРИТМА СРАВНЕНИЯ ГЕНОМОВ

Задача сравнения геномов – это одна из основных задач современной биоинформатики. На текущий момент существуют различные алгоритмы, направленные на ее решение. Среди них можно выделить такие тривиальные как использование расстояния Левенштейна или Хемминга или же более оптимизированные, к примеру, анализ повторов маркерных регионов.

Целью данной работы является разработка модульного приложения с интерфейсом командной строки, которое бы позволяло сравнивать геномы трех людей и определять, являются ли они родственниками (на уровне – мать/отец/сын). Модульность приложения позволяет собирать артефакты с различной конфигурацией (алгоритмами сравнения) без необходимости вносить существенные изменения в основной компонент приложения.

Для достижения данной цели был поставлен ряд задач:

1. Реализация алгоритма сборки геномов на основе предоставленных пользователем файлов.
2. Реализация, качественный анализ и выбор оптимального алгоритма сравнения геномов.
3. Реализация интерфейса командной строки.
4. Оптимизация алгоритма за счет параллелизации процесса сборки и сравнения геномов.

В рамках данной работы были рассмотрены 2 алгоритма сравнения геномов: алгоритм, использующий расстояние Левенштейна [1], и алгоритм, основанный на подсчете числа повторов маркерных регионов [2].

Суть первого подхода заключается в подсчете числа модификаций (удаление, замена и вставка символа), которые необходимо произвести над одной нуклеотидной последовательностью, чтобы получить другую.

Метод маркерных регионов основан на анализе половых хромосом. Каждый человек имеет 2 половые хромосомы. У мужчин это X и Y, у женщин - X и X. У мужчин X хромосома наследуется от матери, а Y от отца [3]. В данных хромосомах есть особые участки, которые имеют повторяющиеся последовательности нуклеотидов, количество повторов которых неизменно из поколения в поколение. Именно эти участки, называемые маркерными регионами, являются основным объектом исследования.

В ходе работы над поставленными задачами было разработано приложение, состоящее из основного и подключаемых модулей для оптимизации процесса тестирования. В качестве языка программирования был использован язык программирования Kotlin. Для автоматизации процесса сборки проекта и обеспечения его модульности был использован фреймворк Apache Maven. Фреймворк Spring Boot был использован в качестве DI-контейнера, что также позволило упростить процесс разделения проекта на модули.

Структура приложения выглядит следующим образом:

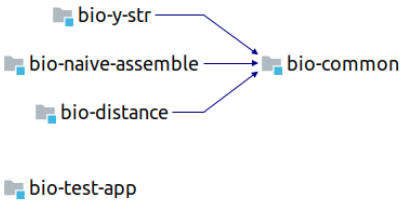


Рисунок 1 – Структура приложения

В ходе оптимизации скорости работы и использования памяти программой было принято решение о параллелизации процессов. Для этого был разработан отдельный модуль kotlin-task-processor, использующий библиотеку Kotlin Coroutines.

Результаты данной работы могут быть представлены в виде следующей таблицы:

Метод сравнения	Тип выполнения	Время работы, мс	Потребляемая память, байт
Расстояние Левенштейна	Последовательный	263163900	2 390 826 120
Маркерные регионы	Последовательный	23052557	552 705 560
Расстояние Левенштейна	Параллельный	159806738	2 482 443 232
Маркерные регионы	Параллельный	15244885	2 629 304 320

Из результатов следует, что метод маркерных регионов является более выигрышным по сравнению с методом, использующим расстояние Левенштейна. Это обусловлено необходимостью в предварительной обработке входных данных, что, в свою очередь, может быть одновременно как преимуществом, так и недостатком данного алгоритма. Также стоит отметить значительный прирост в скорости работы программы после её параллелизации.

ЛИТЕРАТУРА

1. Levenshtein Distance, Sequence Comparison and Biological Database Search [Электронный ресурс] URL:https://www.researchgate.net/publication/341566220_Levenshtein_Distance_Sequence_Comparison_and_Biological_Database_Search (дата обращения 12.03.2021)
2. Genetic Analysis of Ten Gonosomal STR Loci in an Italian Population Using the Elucigene QSTR-XY Amplification Kit [Электронный ресурс] URL: https://www.semanticscholar.org/paper/Genetic-Analysis-of-Ten-Gonosomal-STR-Loci-in-an-Ottaviani-Peconia/d9143b9bd0ec928563454140e115d7d2ae00f1e0?p2df (дата обращения 12.03.2021)
3. Forensic use of Y-chromosome DNA: a general overview [Электронный ресурс] URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5418305 (дата обращения 12.03.2021)

УДК 681.3.06

Мухин Д.А. (2 курс магистратуры)
Горавнева Т.С., к.т.н., доцент

ИССЛЕДОВАНИЕ УЯЗВИМОСТЕЙ ИНФОРМАЦИОННЫХ СИСТЕМ ПРЕДПРИЯТИЙ И МЕТОДОВ ОБЕСПЕЧЕНИЯ ИХ БЕЗОПАСНОСТИ

Хорошо известно, что абсолютно неуязвимых ресурсов не существует, как в силу технического обеспечения, так и в силу человеческого фактора. Сервера нуждаются в правильной настройке, а Web-приложения в написании с применением мер защиты.

В данной работе рассматриваются некоторые из часто встречающих видов атак на Web-приложения, а также защита сервера, на котором находится Web-приложения.

Тройка наиболее распространенных атак на веб-сайты не меняется год от года: «Внедрение SQL-кода» (SQL Injection), «Выход за пределы каталога» (Path Traversal) и «Межсайтовое выполнение сценариев» (Cross-Site Scripting, XSS). Однако, это далеко не весь список уязвимостей.

В данный момент все еще остается проблема, когда пользователи используют довольно простые пароли, которые быстро подбираются перебором (Brute Force). Или используют одинаковые пароли на различных ресурсах, которые могут быть украдены и расшифрованы.

Для предотвращения входа на ресурс злоумышленником, который получил пароль, существует двухфакторная система аутентификации (2FA). Добавление факторов

аутентификации в процесс аутентификации обычно повышает безопасность. Двухфакторная аутентификация добавляет дополнительный уровень защиты к процессу аутентификации. 2FA требует, чтобы пользователь предоставил второй фактор аутентификации в дополнение к паролю. Системы 2FA часто требуют, чтобы пользователь ввел проверочный код, полученный в текстовом сообщении на предварительно зарегистрированном мобильном телефоне, или код, сгенерированный приложением аутентификации.

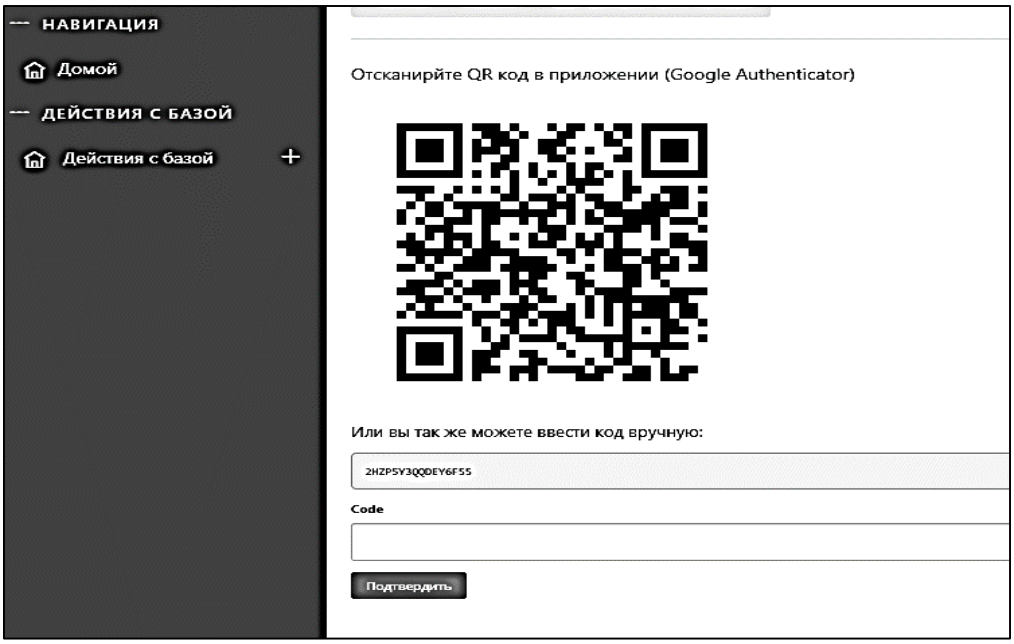


Рисунок 1 – Страница для включения двухфакторной аутентификации

На рисунке 1 представлена страница с QR кодом, где пользователю нужно отсканировать код или ввести его вручную в любое приложение для двухфакторной аутентификации (например, Google Authentification).

При входе в аккаунт, пользователю будет предложено ввести код из приложения в специальное окно (рисунок 2).

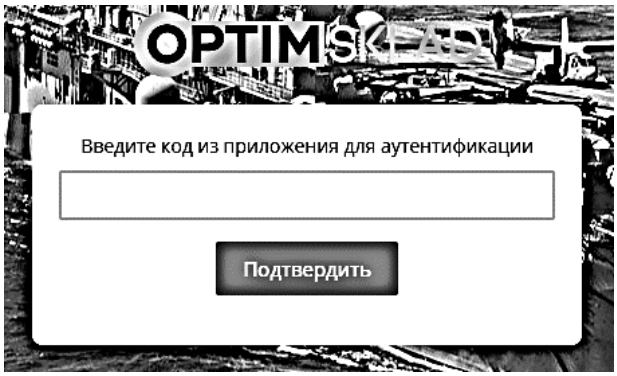


Рисунок 2 – Окно ввода кода из приложения

Помимо защиты приложения, также необходимо защитить сервер, так как при получении доступа к самому серверу злоумышленник может получить много конфиденциальной информации. Общие рекомендации по безопасности серверов:

1. Регулярное обновление безопасности.
2. Уменьшение количества пользователей и ограничение их прав.
3. Изолирование ролей сервисов
4. Отключение SSL2.0 и SSL3.0.
5. Использование HTTP версии не ниже 1.1.
6. Настройка страниц ошибок.

7. Защита от Clickjacking атак.
8. Проверка конфигурационного файла.
9. Изоляция web-контента от системных файлов.
10. Размещение заголовков «Host» на всех сайтах.
11. Отключение возможности просмотра каталогов.
12. Отключение метода «Trace».
13. Проверка режима отладки (debug).
14. Логирование и аудит.
15. Установка минимально нужного набора утилит для работы сервера.

Пример устранения уязвимости Clickjacking на сервере Apache.

Атака данного типа позволяет злоумышленнику выполнить щелчок на сайте-жертве от имени легитимного посетителя. Чтобы обезопасить свой веб-сервер, нужно использовать заголовок «X-FRAME-OPTIONS»

Отредактируем файл apache2.conf.

sudo nano /etc/apache2/apache2.conf или sudo /etc/apache2/sites-enabled/nameofsite.conf

Добавим следующую строку внутри Directory /var/www/html/:

Header always append X-Frame-Options SAMEORIGIN

Сохраним файл и перезапустим Apache.

sudo systemctl restart apache2 или sudo service apache2 restart

Теперь попробуем открыть веб-браузер для доступа к веб-серверу. Проверим заголовки ответа HTTP в firebug, должны увидеть X-Frame-Options.

В ходе работы были устранены уязвимости веб-приложения. Проделана работа по обеспечению защиты сервера. Установлена сертификация. Составлены рекомендации по обеспечению защиты для СУБД и для сервера. Также проведено сканирование веб-приложения и сервера, которое подтверждает их защищенность.

ЛИТЕРАТУРА

1. 3 Ways to Prevent XSS [URL]: https://www.checkmarx.com/2017/10/09/3-ways-prevent-xss/ – (дата обращения: 18.03.2021).
2. Search Security: Статья про аутентификацию [URL] https://searchsecurity.techtarget.com/definition/authentication – (дата обращения: 18.03.2021).
3. Рекомендации по защите: Web-серверы. Windows [URL] https://acribia.ru/recommendations/web-servers_windows – (дата обращения: 18.03.2021).
4. Веб-сервер Apache: установка, настройка и способы защиты от атак [URL] https://tproger.ru/articles/set-up-and-configure-an-apache-server/ – (дата обращения: 18.03.2021).

УДК 004.912

Новоселова Е.И. (4 курс бакалавриата)
Самочадин А.В., к.т.н., доцент

РАЗРАБОТКА СРЕДСТВ ОЦЕНКИ ТРУДОЗАТРАТ НА РЕАЛИЗАЦИЮ ПРОГРАММНЫХ ПРОДУКТОВ НА БАЗЕ РАЗЛИЧНЫХ МЕТОДИК

Целью работы является исследование методик оценки трудозатрат на реализацию программных продуктов и проектирование системы оценки трудозатрат на базе нескольких изученных методик.

Система должна вычислять трудозатраты на разработку проекта несколькими методиками, в зависимости от наличия информации о проекте.

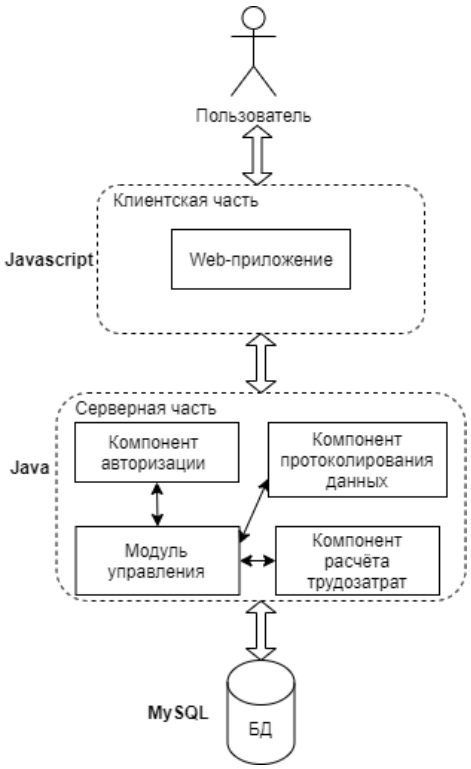


Рисунок 1 – Архитектура системы.

Особенности реализации системы:

Система комбинированная, то есть основана на базе нескольких методик. Для реализации был выбран следующий набор методик: модель COCOMO, метод функциональных точек, метод Use-case point. На основе исходных данных пользователя система выбирает методики, по которым можно провести расчёты. Если добавляются данные, то можно увеличить количество методик для расчётов.

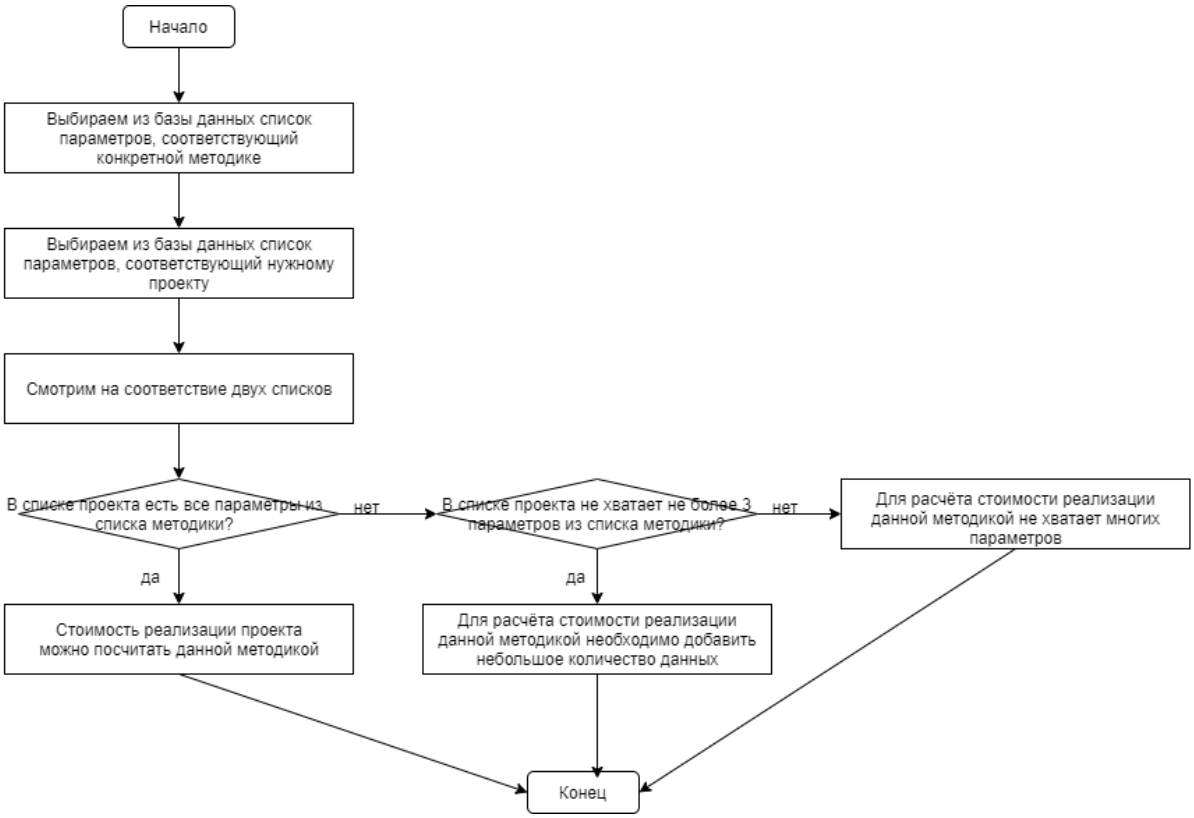


Рисунок 2 – Схема выбора методик.

Также в системе реализована конвертация параметров (позволяет вводить одинаковые параметры для нескольких методик не больше одного раза), подробные инструкции для пользователей и хранение данных о пользователях и об их проектах.

Система состоит из клиентской части, серверной части и базы данных. База данных реализована с помощью MySQL, серверная часть с помощью Java, а клиентская представляет собой Web-приложение, написанное с помощью JavaScript.

Серверная часть также делится на несколько модулей. Модуль управления регулирует работу всех остальных компонентов, в зависимости от действий пользователя; компонент протоколирования данных записывает информацию, введённую пользователем; компонент авторизации взаимодействует с базой данных; компонент расчёта трудозатрат производит необходимые вычисления, исходя из информации, предоставленной пользователем.

ЛИТЕРАТУРА

1. A methodology for software cost estimation ACM SIGSOFT Software Engineering Notes 18(2):96-105
2. An Update on Effort Estimation in Agile Software Development: A Systematic Literature Review
3. Boehm B.W. Software engineering economics. – Prentice-Hall. – 1981. – 320 p.
4. Comparative Study of Software Cost Estimation Techniques Rekha Tripathi, Dr. P. K. Rai
5. David L. Norden-Raleigh Analysis: A Useful Tool for EVM in Development projects. – The Measurable News. – 2002. – 24 p.
6. Software development effort estimation: A systematic mapping study Carlos Eduardo Carbonera1, Kleinner Farias1↚, Vinicius Bischoff
7. Е. В. Ваганова, А.А. Земцов, С. Л. Миньков Оценка стоимости разработки программного продукта: обзор
8. Н. А. Сидоров, Д. В. Баценко, Ю. Н. Василенко, Ю. В. Щебетин Модели, методы и средства оценки стоимости программного обеспечения

УДК 004.02

Пиккио П. Ф., Зайцев Д. И. (4 курс бакалавриата)
Графеева Н. Г., к. ф.-м.н., доцент

ПРОЕКТИРОВАНИЕ КОНСОЛЬНОЙ ИГРЫ ЖАНРА ROGUELIKE

Компьютерные игры жанра RogueLike характеризуются в первую очередь консольной графикой, постоянным использованием случайной генерации, невозможностью загрузить предыдущее сохранение после смерти персонажа, огромным набором игровых правил и высокой свободой действий персонажа.

Целью работы является проведение анализа, построение модели предметной области согласно принципам предметно-ориентированного проектирования, для компьютерной игры жанра RogueLike и описание того, как спроектированная система должна работать.

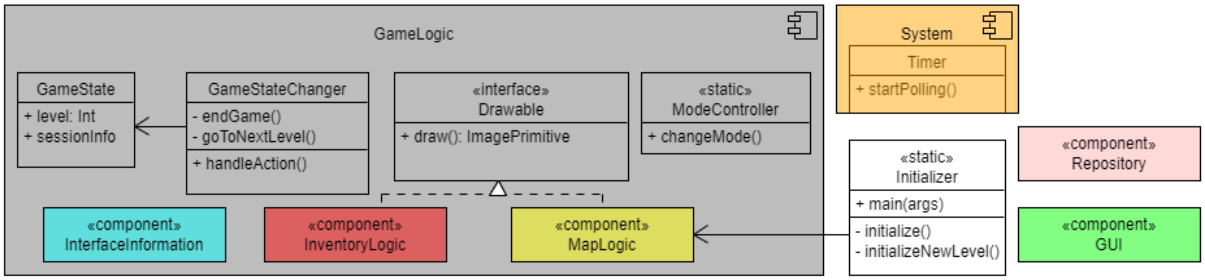


Рисунок 1 – UML - диаграмма компонентов консольной игры.

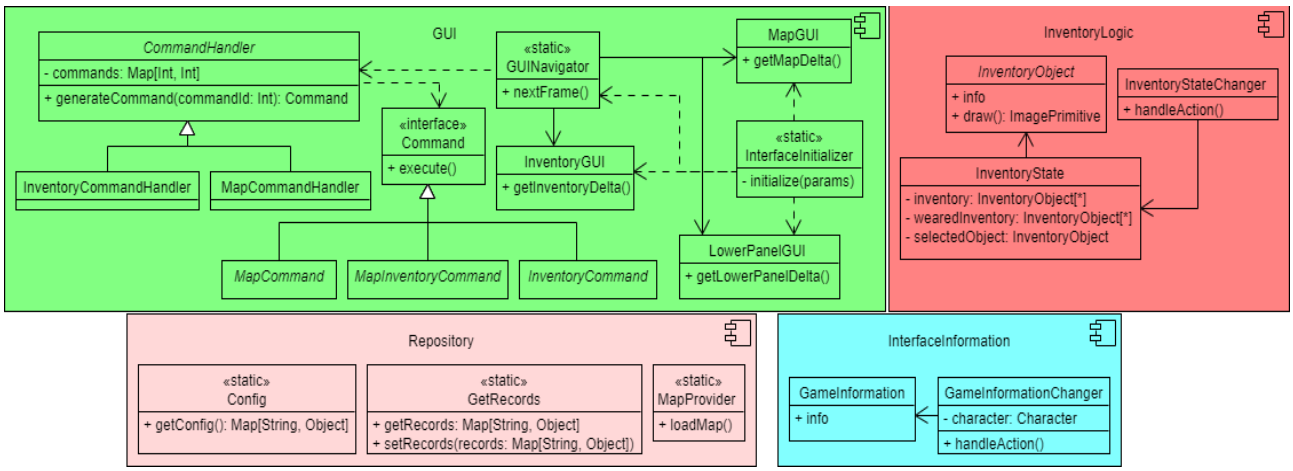


Рисунок 2 – Подробная UML - диаграмма компонентов GUI, Repository, InventoryLogic и InterfaceInformation.

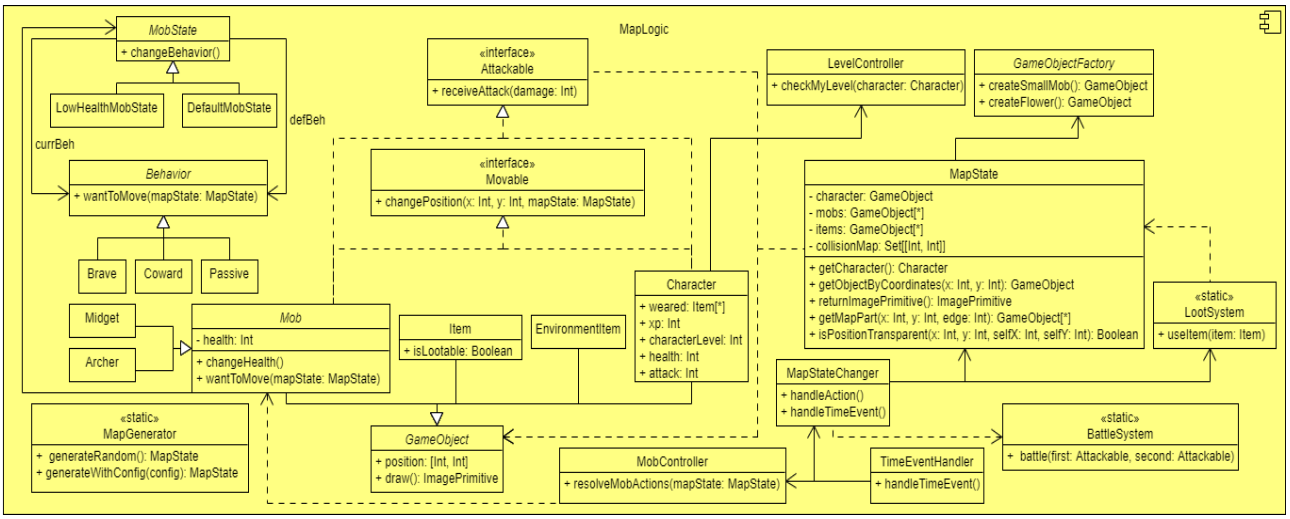


Рисунок 3 – Подробная UML - диаграмма компонента MapLogic.

Особенности спроектированной архитектуры:

В игре существует три игровых области: игровая карта, инвентарь и нижняя панель. Игра может находиться в двух состояниях: открыта карта, открыт инвентарь. Если открыта карта, то действия в игре зависят не только от пользователя. Если открыт инвентарь, то события в игре зависят только от действий пользователя.

Основной компонентой системы служит GameLogic, которая содержит всю логику игры. Она состоит из трех подкомпонент: MapLogic, InventoryLogic и InterfaceInformation. Компонента MapLogic, изображенная на Рисунке 3, содержит все, что касается игрового поля: информацию о локации, персонаже, предметах и правилах их взаимодействия. Компонента InventoryLogic содержит логику инвентаря. Компонента InterfaceInformation отвечает за информацию об игре, отображаемую пользователю. Компонента GUI отвечает за отображение на экран. Компонента Repository отвечает за информацию, сохраняемую между игровыми сессиями. Компонента System отвечает за системно-зависимые элементы. Диаграмма этих компонентов изображена на Рисунке 2. Упрощенная диаграмма всего проекта изображена на Рисунке 1.

Класс Initializer служит для начальной инициализации системы. Классы InventoryState, MapState, GameInformation отвечают за хранение соответствующих данных: объектов и их координат. Класс GameState хранит общую информацию об игре: текущий уровень игры, количество очков. Класс ModeController отвечает за переключение между режимами игры (просмотр инвентаря, игровая карта). Системный класс Timer работает с системным таймером и вызывает методы через промежутки времени. Функциональный интерфейс Drawable содержит один метод draw, который отображает объект или возвращает какой-то примитив, с

помощью которого можно отобразить объект. Класс GameObjectFactory представляет собой паттерн «Абстрактная фабрика», которая создает объекты типа GameObject. Её наследники реализуют методы для создания конкретных предметов по-разному, согласно конкретной теме (например, лес или кладбище). Behavior - общий класс поведений, наследниками которого являются все возможные типы поведений. Этот класс представляет собой паттерн «Стратегия». MobState - абстрактный класс состояний моба. Он представляет собой паттерн «Состояние», хранит в себе ссылку на текущее поведение моба, а также меняет своё состояние и поведение моба в зависимости от уровня его здоровья. Класс LevelController содержит правила перехода по уровням персонажа. При изменении количества опыта вызывается метод этого класса. Этот метод проверяет, достиг ли персонаж нового уровня, и если да, то изменяет персонажа необходимым образом. Может быть реализован по паттерну «Наблюдатель».

При запуске программы поток управления передается в Initializer, метод initialize. Этот метод инициализирует систему в таком порядке: сначала управление передается в Repository. Config, который возвращает конфигурацию системы. Затем в зависимости от темы уровня, которая узнается из конфигурации, создается экземпляр типа GameObjectFactory. Затем метод initialize создает GameState и привязывает к нему GameStateChanger, потом MapState с помощью генератора карт и MapStateChanger, потом также классы инвентаря и нижней панели. Затем управление передается в InterfaceInitializtor в метод initialize, который инициализирует классы инвентаря, связывая их друг с другом и классами логики игры. Затем управление возвращается в главный инициализатор, который связывает классы интерфейсов с классами предметной области. Инициализация GUI включает в себя инициализацию парсера нажатия клавиш для обоих режимов. Затем инициализируется таймер и управление передается в него.

Таким образом, согласно принципам предметно-ориентированного проектирования, была разработана UML - диаграмма для компьютерной игры жанра RogueLike, которая наглядно показывает, как спроектированная система должна работать.

ЛИТЕРАТУРА

1. Мартин Роберт К. Чистая архитектура. Искусство разработки программного обеспечения — 2018.
2. Эрих Гамма, Джон Влисидис, Ричард Хелм, Ральф Джонсон. Design Patterns — 1994.
3. Тепляков Сергей Владимирович. Паттерны проектирования на платформе .NET — 2015.
4. Шелл Джесси. Геймдизайн. Как создать игру, в которую будут играть все — 2020.

УДК 004.415.25

Поликарпова А. И. (2 курс магистратуры)
Самочадин А. В., к.т.н., доцент

РАЗРАБОТКА СИСТЕМЫ ПО РАСЧЕТУ КОРРЕЛЯЦИИ МЕЖДУ УРОВНЯМИ СВОЙСТВ ВНИМАНИЯ И ЧАСТОТОЙ МОРГАНИЙ

Целью данной работы является создание средств, позволяющих проводить мониторинг уровня свойств внимания в процессе интеллектуальной деятельности. Для достижения этой цели, разработан аппаратно-программный комплекс (АПК), ориентированный на решение следующих задач:

– Определение зависимостей между уровнем свойств внимания, определяемых с помощью традиционных психодиагностических методик и параметрами энцефалограмм (ЭЭГ), снимаемых с помощью бытового НКИ, способного снимать ЭЭГ.

– Настройка параметров определения уровня свойств внимания в фоновом режиме для конкретного человека.

– Мониторинг уровня свойств внимания.

Схема архитектуры АПК для режима определения зависимостей приведена на рисунке 1

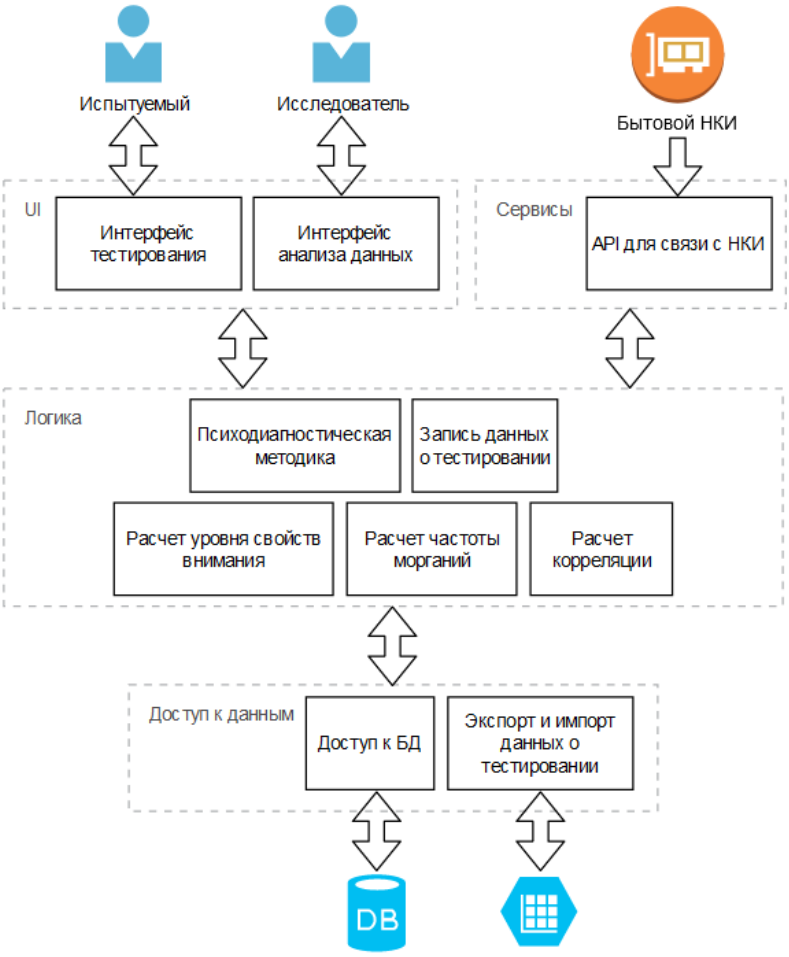


Рисунок 1 – Архитектура АПК для режима определения зависимостей.

Подсистема психодиагностических методик содержит реализации общепринятых методик расчета уровней свойств внимания (например, тест Бурдона). Подсистема записи данных о тестировании работает параллельно с самим психодиагностическим тестом. Её работа заключается в том, чтобы как фиксировать все действия испытуемого во время самого теста (например, в случае теста Бурдона сохраняются все таблицы, выбранные испытуемым символы и время их выбора), так и считывать ЭЭГ при помощи бытового НКИ. Подсистема расчета уровня свойств внимания высчитывает значения различных свойств внимания по времени, основываясь на данных, сохраненных во время тестирования. Подсистема расчета частоты морганий анализирует записанную ЭЭГ и выделяет из неё моргания, выдавая на выходе количество морганий за определенный промежуток времени. На рисунке 2 представлены результаты работы этих двух подсистем.

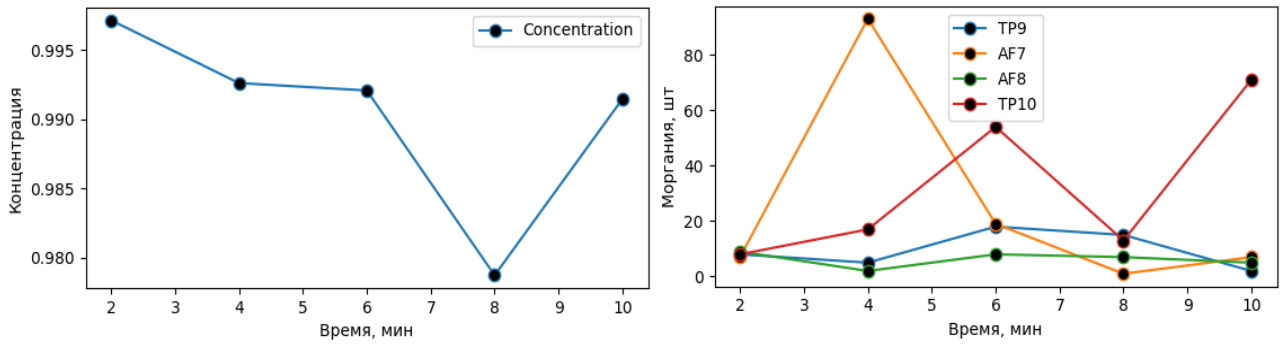


Рисунок 2 – Визуализация концентрации и частоты моргания каждые 2 минуты.

Подсистема расчета корреляции использует ранее высчитанные данные из подсистем расчета уровня свойств внимания и частоты морганий, и рассчитывает уровень зависимости между первыми и вторыми наборами данных.

Для апробации АПК в режиме определения зависимостей проводилась для проверки гипотезы о связи между частотой моргания и уровнем свойств внимания, описанной в таких работах, как [1, 2].

В результате проведенного тестирования разработанной АПК полученное значение корреляции (с учетом выбросов) между частотой моргания и уровнем концентрации внимания оказалось равным 0,32 для канала AF7 – это самое высокое значение из всех. Для остальных каналов и для их среднего, в основном корреляция составляет около -0,23. Все эти значения близки к нулю - взаимосвязь между такими признаками как концентрация и частота моргания слишком мала.

Особенности реализации системы:

Система разрабатывалась на таком языке программирования, как Python 3.7. Для считывания с испытуемого ЭЭГ используется бытовой НКИ Muse-Headband. Для связи между ним и разработанной системой используется приложение BlueMuse [3], работающее на основе технологии Lab Streaming Layer (LSL) [4].

ЛИТЕРАТУРА

1. The validity of eye blink rate by using evaluation of attention / Antanas V. [et al.] // Journal of Medical Sciences. – March, 2017.
2. Paprocki R. What does eye-blink rate variability dynamics tell us about cognitive performance? [Electronic resource] / Paprocki R., Lenskiy A. // frontiersin.org. – URL : https://doi.org/10.3389/fnhum.2017.00620. – (date of access 10.03.2021).
3. BlueMuse [Electronic resource] // github.com. – URL : https://github.com/kowalej/BlueMuse. – (date of access 09.03.2021).
4. Lab Streaming Layer [Electronic resource] // github.com. – URL : https://github.com/sccn/labstreaminglayer. – (date of access 09.03.2021).

УДК 004.4'23

Пости И. М. (2 курс магистратуры)
Танков В. Д, программист

ИНКРЕМЕНТАЛЬНЫЙ АНАЛИЗ И ПРЕДСТАВЛЕНИЕ MARKDOWN ДОКУМЕНТОВ В MARKDOWN PLUGIN ДЛЯ INTELLIJ IDEA

На данный момент Markdown стал де-факто стандартом для написания документации к программным продуктам. Многие текстовые редакторы и интегрированные среды разработки (Integrated Development Environment, IDE) поддерживают редактирование и отображение превью (отображение документа с применённым форматированием и стилями) Markdown документов.

Для реализации привычных для разработчиков функций, таких как подсветка синтаксиса и автодополнение конструкций языка, требуется производить анализ и разбор исходного кода документа. При работе пользователя с документом, в частности, при его изменениях, необходимо обновлять результаты анализа документа и его превью. Почти все редакторы поддерживают редактирование небольших Markdown документов (меньше 5000 строк), однако при работе с документами большего размера во многих редакторах (например, Visual Studio Code[1]) начинают возникать заметные задержки при изменении текста и обновлении превью.

[1] Сайт Visual Studio Code –https://code.visualstudio.com/

В IntelliJ IDEA[1] и других IDE компании JetBrains поддержка Markdown документов осуществляется через плагин Markdown, который поставляется вместе с IDE. При редактировании больших документов в этом плагине также возникают заметные задержки из-за необходимости повторного анализа структуры документа после его изменения и неэффективного обновления превью.

Чтобы передать IDE информацию о структуре языка, необходимо построить PSI дерево. PSI (Program Structure Interface) является элементом платформы IntelliJ, который отвечает за парсинг файлов и создание синтаксической и семантической моделей кода [1]. Любой файл представляется в виде PSI дерева, которое используется IDE для обеспечения работы различных функций, например, для анализа кода или автодополнения. Для построения PSI дерева необходимы лексер, разделяющий содержимое файла на токены, и парсер, который с помощью этих токенов строит абстрактное синтаксическое дерево (Abstract Syntax Tree, AST). Естественно, PSI дерево должно эффективно обновляться после каждого изменения документа [2].

Для того чтобы каждый раз полностью не перестраивать PSI, применяется технология частичного репарсинга. Для этого вводится понятие reparsable элементов, разбор которых может быть произведён локально, без переразбора всего файла, что позволяет значительно ускорить анализ документа при его последовательных изменениях [3].

Этот подход можно применить для разбора Markdown документов, изменения в которых, как правило, локализуются на уровне параграфов. Сделав параграфы reparsable элементами, можно значительно ускорить повторный разбор документа. Для эффективной работы этого механизма требуется быстрый лексер, однако текущая реализация лексера в Markdown плагине является достаточно неэффективной.

Для отображения превью документа его содержимое преобразуется в HTML, который отображается с помощью встроенного в IDE браузера. Такой подход позволяет добиться корректного отображения форматирования и стилей документа. Для обновления превью необходимо сгенерировать новый HTML и обновить отображаемую в браузере страницу. У этого подхода есть существенный недостаток – необходимость полного обновления страницы, что является дорогой операцией даже для небольших документов. Из-за перезагрузок страницы возникают проблемы с сохранением и синхронизацией позиции прокрутки между редактором и превью.

Была поставлена задача улучшить производительность Markdown плагина при редактировании больших документов. Для этого необходимо:

- заменить текущий лексер на новый, на базе JFlex[2];
- добавить поддержку частичного репарсинга для текстовых элементов, находящихся в листьях PSI дерева;
- уменьшить время обновления превью.

Используемый Markdown плагином парсер intellij-markdown[3] уже содержит подходящий лексер на основе JFlex. Токены в выходном потоке этого лексера отличались от токенов старого лексера, из-за чего построенный по нему PSI также отличался от старого. Для решения этой проблемы был реализован набор надстроек, модифицирующих выходной поток токенов базового лексера заданием дополнительных правил для разбиения и объединения некоторых токенов. Для того чтобы новый лексер можно было использовать для построения PSI, потребовалось изменить логику обхода, полученного с помощью парсера AST так, чтобы учесть новые типы токенов и их разбиения.

Очень часто изменения происходят в рамках одного текстового элемента. Поэтому чтобы уменьшить количество вызовов парсера, для текстовых элементов была добавлена поддержка частичного репарсинга. Если произошло простое изменение (не добавляющее новые токены или элементы), находящееся в границах одного листового элемента, то вместо

---

[1] Сайт IntelliJ IDEA – https://www.jetbrains.com/idea/

[2] Сайт JFlex – https://jflex.de

[3] Репозиторий intellij-markdown на GitHub – https://github.com/JetBrains/markdown

вызова парсера на всём поддереве родительского элемента можно заменить содержимое листа на новое. Условием для обновления текстовых элементов является отсутствие вхождений в новую строку токенов, порождающих новые элементы (например, символ новой строки).

Когда какой-либо элемент документа изменяется, в сгенерированном HTML меняется только часть, соответствующая этому конкретному элементу. Поэтому вместо перезагрузки страницы имеет смысл применять локальные обновления для изменившихся HTML элементов. Для этого была использована библиотека Google Incremental DOM[1], позволяющая динамически применять изменения к содержимому страницы. Теперь вместо HTML, соответствующего всему документу, превью выполняет последовательность вызовов функций Incremental DOM, повторяющих структуру сгенерированного HTML, что позволяет обновлять только изменившиеся части страницы.

Благодаря новому лексеру и добавлению поддержки частичного репарсинга текстовых элементов удалось сократить количество вызовов парсера при редактировании параграфов, что уменьшило количество задержек при редактировании документов. В таблице ниже представлены длительности задержек при редактировании документа[2] с большим количеством элементов (~30 000 строк) до и после переработки Markdown плагина.

	IntelliJ IDEA	IntelliJ IDEA после переработки	Visual Studio Code
Изменения в редакторе	~200 миллисекунд	~3 миллисекунды	>1 секунды
Обновление превью	~4 секунды	~60 миллисекунд	>5 секунд

ЛИТЕРАТУРА

1. Program Structure Interface (PSI) – IntelliJ Platform Plugin SDK – 14.01.2021. – URL: https://plugins.jetbrains.com/docs/intellij/psi.html (online; accessed: 12.03.2021).
2. Wagner, T., S. Graham., Incremental analysis of real programming languages. PLDI '97 1997.
3. Implementing Lexer – IntelliJ Platform Plugin SDK – 14.01.2021. – URL: https://plugins.jetbrains.com/docs/intellij/implementing-lexer.html (online; accessed: 12.03.2021).

УДК 004.4

Станкевичуте Е. (4 курс бакалавриата)
Воинов Н.В., к.т.н., доцент
Леонтьева Т.В., к.т.н., доцент

РАЗРАБОТКА TELEGRAM-БОТА ДЛЯ СПОРТИВНОГО КЛУБА «ПОЛИТЕХНИК»

Основной целью данной работы является централизация разнообразной информации о деятельности спортивного клуба «Политехник», центра тестирования по выполнению нормативов испытаний (тестов) Всероссийского физкультурно-спортивного комплекса «Готов к труду и обороне», об оказываемых спортивно-оздоровительных услугах спортивным комплексом «Политехник» и о получении зачёта по дисциплине «Элективная физическая культура и спорт». В ходе выполнения работы необходимо разработать систему, предоставляющую информацию об актуальных новостях, расписании, а также иную полезную информацию для студентов и сотрудников СПбПУ, клиентов спортивного комплекса «Политехник» и желающих сдать нормативы ГТО.

Разрабатываемая система должна получать и хранить данные из разных информационных ресурсов. Telegram-бот должен позволять пользователям удобно и быстро получать необходимую информацию по интересующему их вопросу. Помимо

[1] Репозиторий Google Incremental DOM на GitHub – https://github.com/google/incremental-dom
[2] https://gist.github.com/FirstTimeInForever/d8a2d98c4cb32e2d3b79e4d8de9fd6f8

непосредственной разработки бота необходимо реализовать его развертывание на сервере, обеспечивающем постоянную работу и доступ к боту.

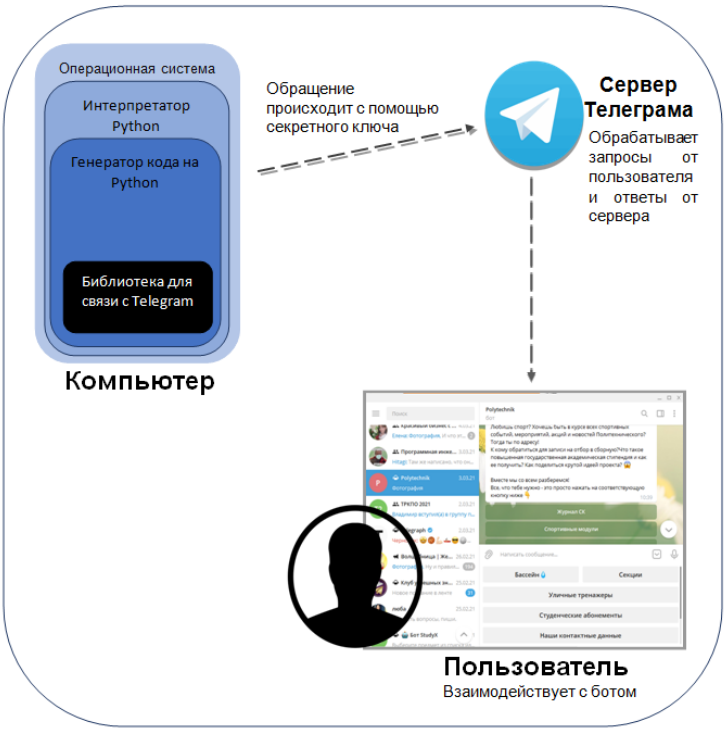


Рисунок 1 – Схема взаимодействия между элементами разрабатываемой системы.

Система будет состоять из трех элементов: компьютер с Python, сервер Телеграма и Телеграм-клиент. На компьютере работает интерпретатор Python, а внутри интерпретатора выполняется программа, написанная на языке программирования Python. Она отвечает за весь контент: в неё заложены все шаблоны текста, вся логика, всё поведение. Внутри программы на Python работает библиотека Telegram Bot API, которая отвечает за общение с сервером Телеграма. В библиотеку вводится секретный ключ - токен, чтобы сервер Телеграма понимал, что программа связана с определённым ботом.

В данный момент бот функционирует только при запущенной программе. После официального согласования с директором Спортивного клуба «Политехник» и директором ИФКСТ, разработанный бот будет загружен на удаленный сервер и постоянного доступа.

Взаимодействие между пользователем и ботом выглядит следующим образом (Рисунок 1): пользователь бота отдает ему команду, запрос приходит на сервер, а сервер отправляет его на компьютер. Запрос обрабатывается программой, далее ответ высылается на сервер Телеграма и после сервер отдаёт ответ пользователю в виде вывода ответа ботом на экран приложения пользователю. Такой цикл выполняется каждый раз, когда пользователь нажимает на одну из кнопок и взаимодействует с Телеграм-ботом.

ЛИТЕРАТУРА

1. Python Telegram Bot’s documentation: [Электронный ресурс]. (https://python-telegram-bot.readthedocs.io/en/stable/index.html). Дата обращения: 26.01.2021.
2. Building Your First Telegram Bot: A Step by Step Guide: [Электронный ресурс]. (https://www.toptal.com/python/telegram-bot-tutorial-python). Дата обращения: 15.01.2021.
3. Встроенные кнопки в Telegram Bot API — pyTelegramBotAPI: [Электронный ресурс]. (https://habr.com/ru/post/335886/). Дата обращения: 02.02.21.
4. Полное руководство Python по созданию Telegram Bot с использованием python-telegram-bot: [Электронный ресурс]. (https://dev-gang.ru/article/ja-postroil-telegrafnyi-bot-dlja-borby-s-pisczevymi-othodami-vot-kak-eto-delaetsja-inaqfmq470/). Дата обращения: 10.03.21.
5. Как создать чат-бота для Telegram с помощью Python: [Электронный ресурс]. (https://pythonru.com/primery/python-telegram-bot). Дата обращения: 10.03.21.

УДК 004.4

Стягов А. Ю. (4 курс бакалавриата)
Маслаков А. П., ст. преподаватель

ПЕРСПЕКТИВЫ ДЕКЛАРАТИВНОГО ПОДХОДА К СОЗДАНИЮ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ В ANDROID

Фреймворк Android предоставляет обширные возможности для реализации пользовательских интерфейсов с использованием большого количества встроенных визуальных компонентов системы из набора стандартных библиотек. Подобные элементы пользовательского интерфейса или виджеты представляют собой классы, расширяющие базовый класс android.view.View [1]. Помимо этого фреймворк предоставляет важный класс android.view.ViewGroup [2], также расширяющий класс View. Класс ViewGroup и его наследники представляют собой контейнеры для набора отдельных виджетов, располагаемых на экране устройства по какому-либо правилу или принципу. Благодаря иерархии классов-наследников (как встроенных, так и пользовательских) View и ViewGroup у пользователей появляется возможность создавать любые визуальные элементы и располагать их на экране друг относительно друга.

Фреймворк Android поддерживает возможность «вёрстки» пользовательских интерфейсов с помощью XML файлов разметки, в котором элементами XML схемы являются компоненты UI [3]. В зависимости от того, является ли элемент наследником ViewGroup, в него могут быть «вложены» другие компоненты UI.

Так как сам по себе файл в формате XML не является понятным для фреймворка, существует встроенный механизм «инфлейта» (layout inflating), то есть преобразования такого файла в иерархию объектов соответствующих типов. Данный подход является чисто императивным: класс View сам по себе является крайне «раздутым» по части поддерживаемых атрибутов, начиная от обязательных для всех виджетов атрибуты ширины и высоты, заканчивая атрибутами цвета фона и размер отступов от других виджетов – всё это класс использует для самостоятельной отрисовки себя на экране. При обработке файла XML разметки для каждого элемента схемы создаётся объект класса-наследника View или ViewGroup, всего атрибуты соответствующего ему тэга обрабатываются и преобразуются в значения полей класса, а вложенные внутрь виджеты добавляются в его внутреннюю иерархию. Важное следствие императивного подхода в создании UI – возможность мутации состояния виджета посредством прямых манипуляций с его полями (например, изменение его размеров или цвета во времени исполнения в ходе процессинга какого-либо пользовательского сценария). Из-за этого пользователям фреймворка необходимо при работе с состоянием приложения и отдельных его частей учитывать и/или хранить также состояния отдельных визуальных компонентов, что безусловно является перманентным источником ошибок и сбоев в работе приложений.

Альтернативой императивному является декларативный пользовательский интерфейс [4]. Если императивный подход заключается в описании того «как» мы хотим получить результат, то с декларативным подходом мы описываем только «что» мы хотим получить, без указания конкретных шагов к достижению результата. В последние годы декларативный подход набирает популярность, продемонстрировав свои преимущества сначала в области вэб-разработки (например, React), затем и в мобильной разработке (Flutter – фреймворк кроссплатформенной разработки, Swift UI в iOS).

Решением для нативного Android является библиотека Compose от Google, входящая в набор библиотек Jetpack [5]. Реализация виджетов в Compose основана не на иерархии расширений класса View, а на композиции чистых функций, отрисовывающих иммутабельные UI компоненты на экране. Это свойство можно назвать как однонаправленный поток данных или unidirectional data flow, и оно являет собой решение проблемы с хранением и обработкой

состояний, описанной выше. С точки зрения подхода к разработке, в декларативном UI отдельные виджеты более не хранят в себе избыточные поля и атрибуты (в том числе значения собственных размеров), а вся переработанная система визуальных компонентов следует принципам SOLID, в частности принципам единственной ответственности и разделения интерфейса.

Для сравнения можно рассмотреть виджет для отображения текста. Класс TextView из стандартной библиотеки Android позволяет определить содержание текста и его стиль (цвет, размер начертания, шрифт), а также размеры, фон (цвет или изображение), отступы (внутренние и внешние), обработчик нажатия, видимость, значение альфа-канала и многое другое, что безусловно важно для отрисовки на экране, но не имеет отношения к задаче – отображение текста. В свою очередь виджет Text из библиотеки Compose работает только с содержанием текста и с его стилем [6]. Для добавления различных модификаций существует специальная иммутабельная структура Modifier, предоставляющая стратегии отображения фона, отступов и прочего.

Таким образом, с точки зрения пользователя API становится прозрачнее, структура виджетов становится гораздо более плоской, убирается зависимость от версии ОС, теряется необходимость работать с несколькими форматами и подходами: весь код, как бизнес-логика, так и пользовательский интерфейс можно писать на одном языке в одном файле.

На данный момент библиотека в стадии активной разработки и в феврале 2021 года вышла её первая бета-версия, с приходом которой существовавший к тому моменту API успел претерпеть серьёзные изменения. Это говорит о том, что использовать библиотеку в рабочих проектах на данный момент не представляется возможным из-за опасности получить нерабочий код в ближайшем обновлении, по крайней мере до выхода первой стабильной версии.

На мой взгляд очевидно, что декларативный подход с использованием однонаправленных потоков данных станет приоритетным выбором в ближайшем будущем. В связи с этим считаю крайне актуальным создание инструмента для автоматической конвертации существующих XML разметок или их частей в функции библиотеки Compose, чтобы автоматизировать и ускорить перенос существующих приложений на новый подход, а также для упрощения изучения данной библиотеки пользователям.

ЛИТЕРАТУРА

1. View – Android Developers [Электронный ресурс]: https://developer.android.com/reference/android/view/View (дата обращения: 14.03.2021)

2. ViewGroup – Android Developers [Электронный ресурс]: https://developer.android.com/reference/android/view/ViewGroup (дата обращения: 14.03.2021)

3. Layout resource – Android Developers [Электронный ресурс]: https://developer.android.com/guide/topics/resources/layout-resource (дата обращения: 14.03.2021)

4. Jetpack Compose – a modern declarative UI toolkit for Android [Электронный ресурс]: https://blog.qburst.com/2020/10/jetpack-compose-a-modern-declarative-ui-toolkit-for-android (дата обращения: 14.03.2021)

5. Jetpack Compose [Электронный ресурс]: https://developer.android.com/jetpack/compose (дата обращения: 14.03.2021)

6. Text in Compose – Android Developers [Электронный ресурс]: https://developer.android.com/jetpack/compose/text (дата обращения: 14.03.2021)

УДК 004

Талалова Е.В. (2 курс магистратуры)
Тышкевич А.И., к.т.н., доцент

ПОСТОБРАБОТКА РАДИОЛОКАЦИОННЫХ ДАННЫХ ДЛЯ УЛУЧШЕНИЯ КАЧЕСТВА РАСПОЗНАВАНИЯ В ПОЛЕ ОБЗОРА РАДИОЛОКАЦИОННОГО ДЕТЕКТОРА

Основной задачей развития и совершенствования современных средств и систем в радиолокации является повышение эффективности их использования и качества работы. Решение этой задачи напрямую связано с разработкой оптимальных сигналов с заданными свойствами. Сейчас значительно увеличился интерес к использованию сверхширокополосных сигналов.

В последнее время в большом количестве работ исследуются свойства сверхширокополосных сигналов. Использование сверхширокополосных сигналов позволяет увеличить разрешающую способность, обеспечить наилучшим образом прохождения сигналов через среду, характеристики которой изменяются со временем.

Целью написания данной работы является исследование методов постобработки радиолокационных данных для улучшения качества распознавания в поле обзора радиолокационного детектора.

При написании работы были поставлены следующие задачи:

1. Исследовать актуальность улучшения качества распознавания в поле обзора радиолокационного детектора;

2. Рассмотреть пути повышения качества распознавания в поле радиолокационного детектора;

3. Представить моделирование работы радиолокационного детектора с предварительным выбором среды имитационного моделирования.

Объект – исследование методов повышения качества распознавания для радиолокационных детекторов в задачах постобработки сигналов.

Выполним постановку задачи обнаружения и рассмотрим математический аппарат задачи обнаружения. В большинстве случаев наблюдатель и цель движутся с постоянными курсом и скоростью, при этом: v – скорость наблюдателя; u – скорость цели; w – скорость цели относительно наблюдателя (относительная скорость). В нашем случае наблюдатель – детектор.

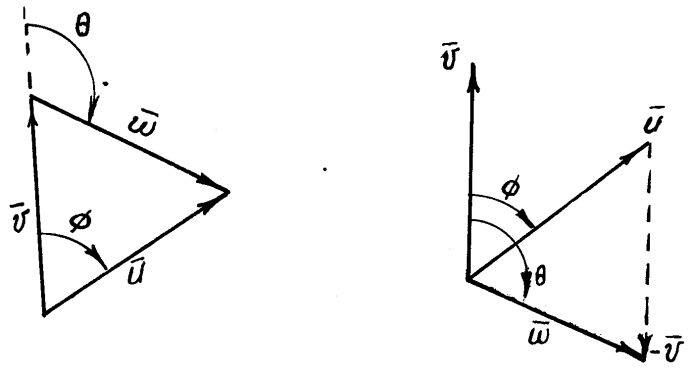


Рисунок 1 – Истинные и относительные скорости и углы.

В результате проведенного обзора сред имитационного моделирования принято решение остановиться на Matlab/Simulink.

Программный пакет MATLAB от компании разработчика MathWorks - один из самых мощных инструментов обработки данных на рынке программных продуктов.

Выполним построение усовершенствованной модели обработки сигналов в Matlab/Simulink.

Имитационная модель системы представлена на рисунке 2.

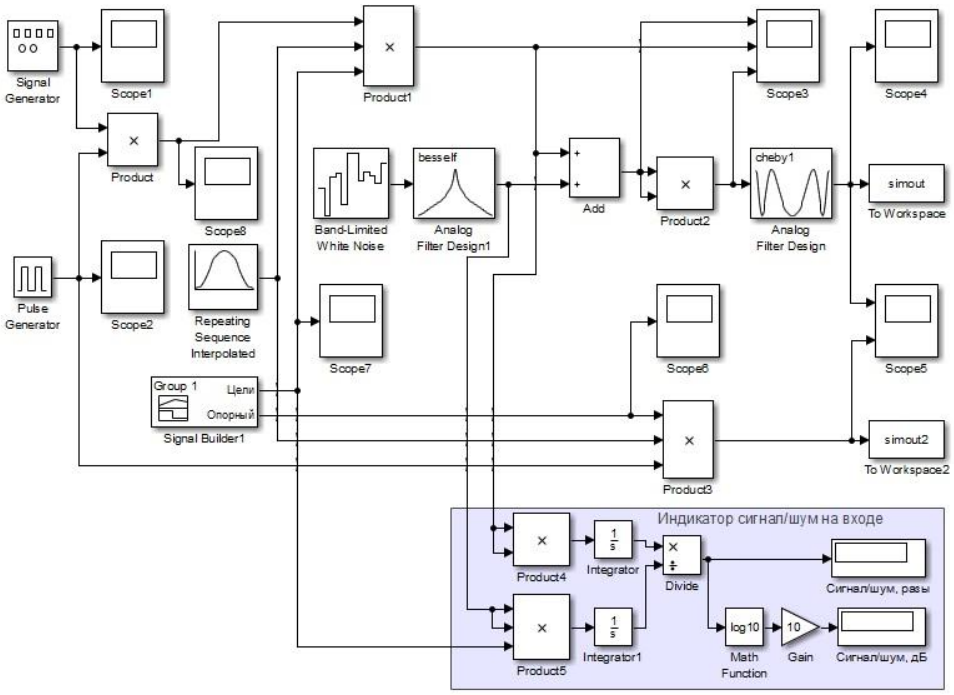


Рисунок 2 – Модель.

Результаты обнаружения сигнала представлены на рисунках ниже.

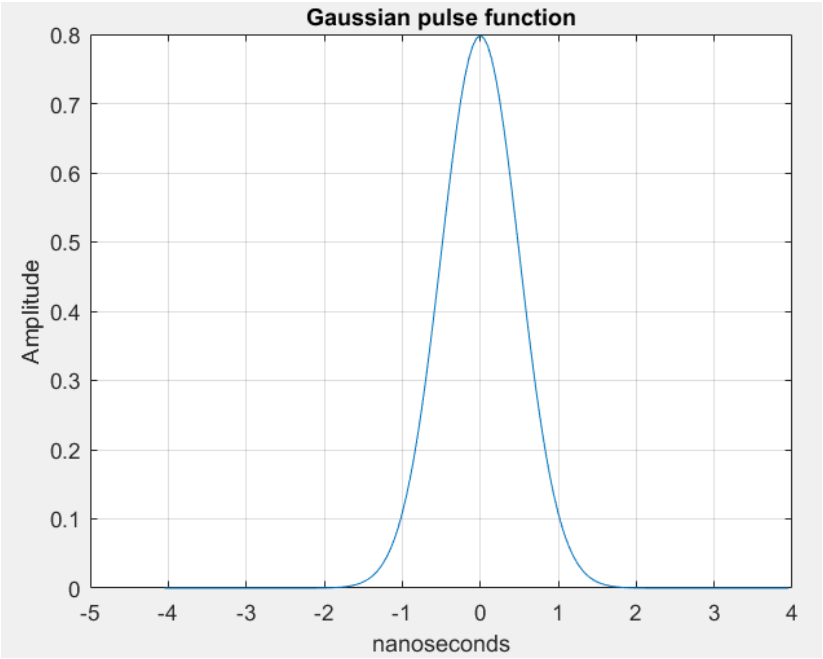


Рисунок 3 – Результаты обнаружения Гауссового импульса.

ЛИТЕРАТУРА

1. Айфичер, Э. Цифровая обработка сигналов: практический подход / Э. Айфичер. - М.: Вильямс, 2016. - 992 с.
2. Чан, Т.Т. Высокоскоростная цифровая обработка сигналов и проектирование аналоговых систем / Т.Т. Чан; Пер. с англ. К.В. Юдинцев. - М.: Техносфера, 2013. - 192 с.

3. Оппенгейм, А. Цифровая обработка сигналов / А. Оппенгейм, Р. Шафер. - М.: Техносфера, 2006. - 856 с.
4. Гадзиковский, В.И. Цифровая обработка сигналов / В.И. Гадзиковский. - М.: Солон-пресс, 2015. - 766 с.
5. Лайонс, Р. Цифровая обработка сигналов / Р. Лайонс.. - М.: Бином-Пресс, 2013. - 656 с.

УДК 004.42

Усов М.А. (4 курс бакалавриата)
Воинов Н.В., к.т.н., доцент
Зайцев И.В., ст. преподаватель

РАЗРАБОТКА ANDROID-ПРИЛОЖЕНИЙ НА ОСНОВЕ ДЕКЛАРАТИВНОГО ПОДХОДА С ПРИМЕНЕНИЕМ JETPACK COMPOSE

Целью работы является изучение и применение нового подхода к разработке мобильных приложений на операционной системе (ОС) Android. До недавнего времени приложения разрабатывались сначала в разметке xml-файлов, а затем подключались в основной код, чтобы взаимодействовать с ним. В настоящее время набирает распространение декларативный стиль разработки, позволяющий полностью отказаться от разметки и писать единый код без разделения на разметку и основной код. Данный подход реализуется с помощью Jetpack Compose – разработки компании Google для ОС Android [1,2].

Рассмотрим пример (Рисунок 1). Функция содержит аннотацию @Composable, которая дает понять компилятору, что используется декларативный подход. Представленным способом описывается карточка с картинкой и анимированным текстом. Также данная технология поддерживает функцию hot reload, которая позволяет быстро применять измененный код без компилирования всей программы.

```
@Composable
fun JetpackCompose() {
    Card {
        var expanded by remember { mutableStateOf(false) }
        Column(Modifier.clickable { expanded = !expanded }) {
            Image(painterResource(R.drawable.jetpack_compose))
            AnimatedVisibility(expanded) {
                Text(
                    text = "Jetpack Compose",
                    style = MaterialTheme.typography.h2,
                )
            }
        }
    }
}
```

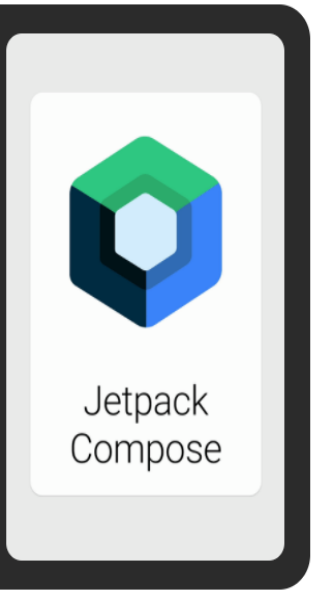


Рисунок 1 – Пример описания функции при декларативном подходе.

Для использования данного подхода достаточно иметь Android Studio последней версии и подключить необходимые библиотеки в файле build.gradle. Для подключения нужных модулей в тот же файл необходимо добавить код, представленный на Рисунке 2.

```
buildFeatures {
    // Enables Jetpack Compose for this module
    compose true
}
```

Рисунок 2 – Подключение необходимых модулей.

Ожидаемым результатом работы должен стать переход от старого подхода к написанию кода с помощью двух файлов к подходу с одним декларативным. Такой подход к реализации всего проекта позволит уменьшить кодовую базу от наследованных классов, предоставит обратную совместимость с прошлым кодом, освободит от зависимости от конкретных старых платформ, а также даст возможность автоматического сохранения state приложения.

ЛИТЕРАТУРА

1. Compose. Jetpack Compose: https://habr.com/ru/company/epam_systems/blog/522152/
2. Use Android Studio with Jetpack Compose: https://developer.android.com/jetpack/compose/setup

УДК 004.4`2

Фигловский К. С. (1 курс магистратуры)
Никифоров И. В., к.т.н., доцент,
Юсупова О.А.

ИСПОЛЬЗОВАНИЕ GRADLE BUILD CACHE ДЛЯ ОПТИМИЗАЦИИ ВРЕМЕНИ СБОРКИ

На сегодняшний день всё более-менее сложное ПО собирается с использованием систем автоматических сборок (make, Apache Maven, Gradle и т. д.). При этом одна из важнейших задач при работе с такими системами - оптимизация времени сборки. В данной работе будет рассмотрен механизм кэширования, который можно использовать для ускорения Gradle [1].

Gradle build cache - это механизм кеширования, направленный на экономию времени за счет повторного использования результатов, созданных другими сборками. Есть два уровня кэширования – кэш на локальной машине (local cache) и кэш на удаленной (remote cache) [2]. Дополнительно Gradle хранит состояние последних выполненных задач и не перезапускает их без необходимости.

Алгоритм работы Gradle с кэшами задач после запуска сборки пользователем (рисунок 1) следующий:

1. Проверяем является ли статус задачи актуальным. Если да – считаем ее завершенной.
2. Проверяем наличие данных по задаче в локальном кэше. Если нашли – считаем её завершенной.
3. Проверяем наличие данных по задаче в удаленном кэше. Если нашли – копируем данные в локальный кэш, задачу считаем завершенной.
4. Если на шагах 1-3 задача не была завершена, то выполняем ее. Результат сохраняем в кэш. Локальный или удаленный зависит от настроек, заданных разработчиком.

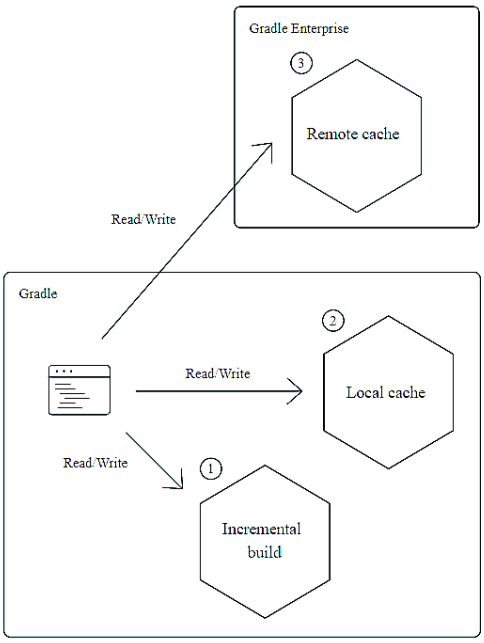


Рисунок 1 – Принцип работы кэширования Gradle.

Оценка влияния кэширования на время сборки

Для оценки влияния Gradle build cache на время сборки было использовано open-source Android приложение

«VLC for Android» [3,4]. Рассматривалось две конфигурации проекта:

1. Без кэширования (конфигурация проекта по умолчанию).
2. С использованием Remote и Local cache.

Для настройки удаленного кэширования, в рабочей среде был развернут официальный образ docker-контейнера – «Gradle Enterprise build cache node» [5]. Размер кэша был установлен в 10000 MiB, максимальный размер одного артефакта – 100 MiB.

Замеры времени проводились при помощи утилиты Gradle Profiler [6]. Алгоритм измерений, следующий: Утилита «убивает» все системные процессы, связанные с Gradle. Затем в цикле выполняется определенное количество «чистых» сборок проекта, в нашем случае 16. «Чистая» сборка означает, что перед её началом все предыдущие артефакты были удалены (не включая кэши).

Конфигурация системы:

Windows 10, Gradle 6.8.3, Java 1.8.0_251 (-XX:MaxPermSize=2048m -Xms512M -Xmx4g)

Характеристики машины:

Intel® Core™ i7-10710U @ 1.10GHz – 4.70GHz, 16 GB DDR4 RAM

Результаты измерений можно увидеть на рисунке 2 ниже.

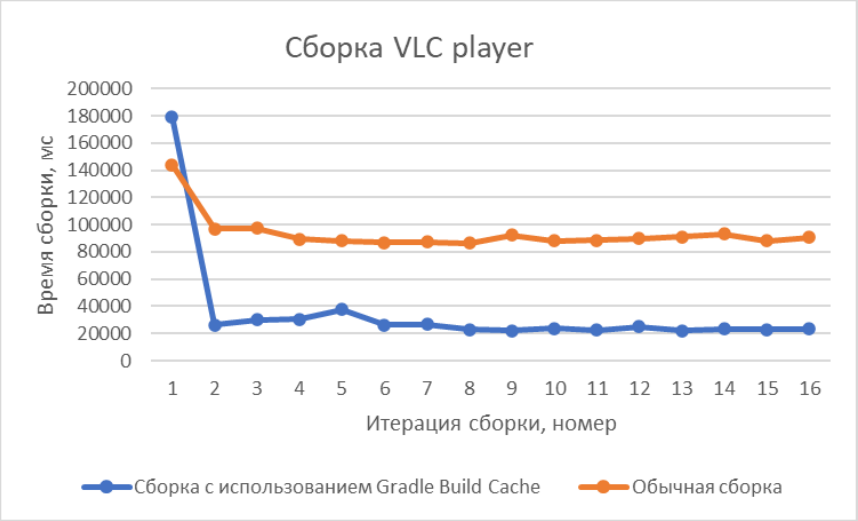


Рисунок 2 –. Время сборки VLC player.

В таблице 1 приведены расчеты по среднему, медианному, максимальному и минимальному времени сборки.

Таблица 1 – Статистические показатели по времени сборки.

	Время сборки с кэшированием (миллисекунды)	Время сборки без кэширования (миллисекунды)	Прирост скорости
Медиана	24412,5	89651,5	+73%
Среднее	35332,1875	93665,3125	+62%
Макс	179550	144139	-25%
Мин	22089	86526	+74%

Из полученных данных можно сделать вывод о том, что использование Gradle Build Cache позволило в среднем ускорить время сборки на 62%. При этом прирост скорости по медианному времени составляет 73%. Это связано с тем, что на первой итерации какая-то часть ресурсов потратилась на сохранение данных в Remote cache. Скорее всего, при большем количестве прогонов среднее время будет стремиться к медианному.

ЛИТЕРАТУРА

1. Gradle User Manual. URL: https://docs.gradle.org/

2. Сысоев И.М., Никифоров И.В., Каплан Е.В., Создание подхода автоматизации обновления конфигурационных файлов программного обеспечения, Современные технологии в теории и практике программирования. сборник материалов конференции. 2019. С. 126-127.
3. VLC for Android, GitHub: URL: https://github.com/videolan/vlc-android
4. Смольский А.Л., Воинов Н.В., Прокофьев О.В., Адаптация среды выполнения языка objective-c для платформы Android, Современные технологии в теории и практике программирования. сборник материалов конференции. 2019. С. 196-197.
5. Gradle Enterprise build cache node. URL: https://hub.docker.com/r/gradle/build-cache-node/
6. Gradle: gradle-profiler. GitHub. URL: https://github.com/gradle/gradle-profiler

УДК 004.4

Фролова А.Ю. (2 курс магистратуры)
Никифоров И.В., к.т.н., доцент

АВТОМАТИЗАЦИЯ СБОРА И РАСЧЕТА РЕКЛАМНЫХ МЕТРИК В ПРОГРАММНОМ СРЕДСТВЕ НА ОСНОВЕ ДАННЫХ ВНЕШНИХ ИСТОЧНИКОВ

Цель и задачи работы. Целью данной работы является снижение трудоемкости оценки эффективности использования рекламы за счет подхода автоматизации сбора и расчета метрик, реализованный в программном средстве, отличительной способностью которой является использование технологии веб скраппинга для доступа к данным с внешних ресурсов [1]. Разработка программного средства осуществляется в виде веб-приложения фреймворка Django [2], так как на сегодняшний день данный фреймворк является одним из самых популярных на языке Python [3].

Для достижения цели необходимо решить следующие задачи:

- Провести обзор способов оценки эффективности использования рекламы.
- Провести сравнительный анализ технологий веб скраппинга.
- Предложить подход автоматизации сбора и расчета метрик.
- Реализовать предложенный подход в программном средстве.
- Продемонстрировать эффективность работы программного средства.
- Показать снижение трудоемкости. Подвести итог.

Реализация подхода сбора метрик с использованием веб скрапинга.

Веб скраппинг - это процесс извлечения данных с внешних источников. В данной работе используется более 10 внешних источников, поэтому необходимо реализовать единый класс для получения данных. Для реализации первой задачи (определить подходящую архитектуру для сбора метрик с использованием веб скрапинга) была выбрана технология интеграции Scrapy [4] и Django. Эта система позволяет определять структуры данных, записывать данные, имеет встроенные селекторы, такие как CSS и XPath, которые мы можем использовать для извлечения данных, также имеет вывод данных в форматы: JSON, CSV и XML.

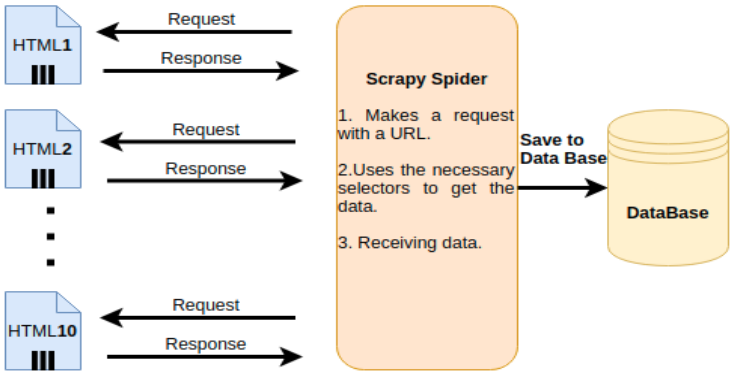


Рисунок 1 – Архитектура сбора метрик с использованием Веб Скрапинга.

На рисунке 1 представлена архитектура сбора метрик с использованием Веб Скрапинга. Главной особенностью представленной модели является блок Scrapy Spider, который использует методы для отправки запросов, парсинга запросов и записи готовых данных в базу. Для каждого внешнего источника создана отдельная модель в базе данных, которая хранит рекламные метрики (рекламные показы и количество потраченных денег) за каждый час.

Реализация подхода автоматизации сбора и расчета метрик.

В данной задаче был реализован подход, который позволяет запускать задачу веб-скраппинга каждый день, в одно и тоже время, для получения данных с внешних источников и сохранения их в базу данных, для того, чтобы использовать эти данные для расчета новых метрик. В качестве архитектуры автоматизации была выбрана библиотека Celery [5], которая использует систему управления задачами с поддержкой запуска в любое установленное время.

На рисунке 2 представлена архитектура автоматизации сбора рекламных метрик. На данном этапе уже реализовано веб приложение, в котором прописаны условия для запуска ежедневной задачи. Далее задача для запуска веб скраппиннга (описанная выше) отправляется в систему очередей RabbitMQ [6], выполняется асинхронно задача скраппинга данных, и записывается результат в базу данных.

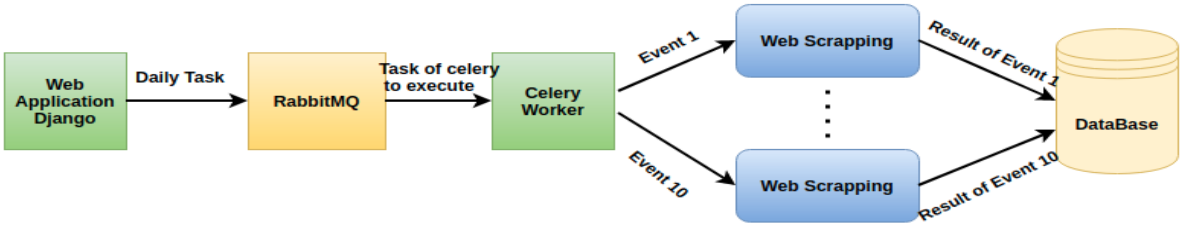


Рисунок 2 – Архитектура автоматизации сбора метрик.

Результаты.

Таблица 1 – Сравнительный анализ.

Решение/количество метрик	3 (Impressions, Spend, Clicks)	5 (Impressions, Spend, Clicks, CPM, CTR)	10 (mpressions, Spend, Clicks,...)
Создание ручного отчета	60 минут	90 минут	Более 100 минут рабочего времени
Создание автоматизированного отчета	1 минута	2 минуты	Менее 5 минут

Таким образом был реализован подход к автоматизации сбора и расчета метрик, который позволяет ускорить работу создания отчетов.

ЛИТЕРАТУРА

1. Никифоров И.В., Курсовое проектирование по учебной дисциплине "Наука о данных и аналитика больших объемов информации", Учебное пособие / Санкт-Петербург, 2017.
2. Документация Django. [Электронный ресурс]. - https://docs.djangoproject.com/en/3.1/
3. Сараджишвили С.Э., Леонтьев В.В., Воинов Н.В., Введение в обработку изображений на языке Python, санкт-петербург, 2020.
4.Документация Scrapy. [Электронный ресурс]. - https://docs.scrapy.org/en/0.24/index.html
5. Документация Celery. [Электронный ресурс]. - https://docs.celeryproject.org/en/stable/
6. Петров А.А., Селин И.А., Разработка программы по объединению облачных хранилищ данных, Современные технологии в теории и практике программирования. сборник материалов конференции. Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. Санкт-Петербург, 2020. С. 164-165.

УДК 681.3.06

Ченский Т. С. (2 курс магистратуры)
Семенова-Тян-Шанская В.А., к.т.н., доцент

СРАВНЕНИЕ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ И АЛГОРИТМОВ КЛАСТЕРНОГО АНАЛИЗА ДАННЫХ

Целью представленной работы является сравнение некоторых инструментальных средств и алгоритмов кластерного анализа данных.

Работа включает в себя хранилище данных, размещенное на сервере PostgreSQL, которое выступает в качестве источника данных для интеллектуального анализа, проект многомерных данных и интеллектуального анализа данных служб Analysis Service и несколько проектов на языке Python, использующих библиотеку scikit-learn, содержащую реализации методов иерархической и неиерархической кластеризации (алгоритм k-средних), а также собственную реализацию алгоритма кластеризации методом k-средних.

В качестве исследуемой предметной области была выбрана «Оценка экономических показателей рейса судна». Актуальность данной предметной области состоит в том, что современная мировая торговля, и в частности торговый флот, имеет тенденцию к высоким темпам развития, что создает благоприятные и перспективные предпосылки для использования аналитических средств обработки информации.

Для построения хранилища данных были выделены следующие сущности: Судно, Время, Маршрут, Порт, Рейс. На основании этих сущностей было создано хранилище с архитектурой «снежинка» (Рисунок 1).

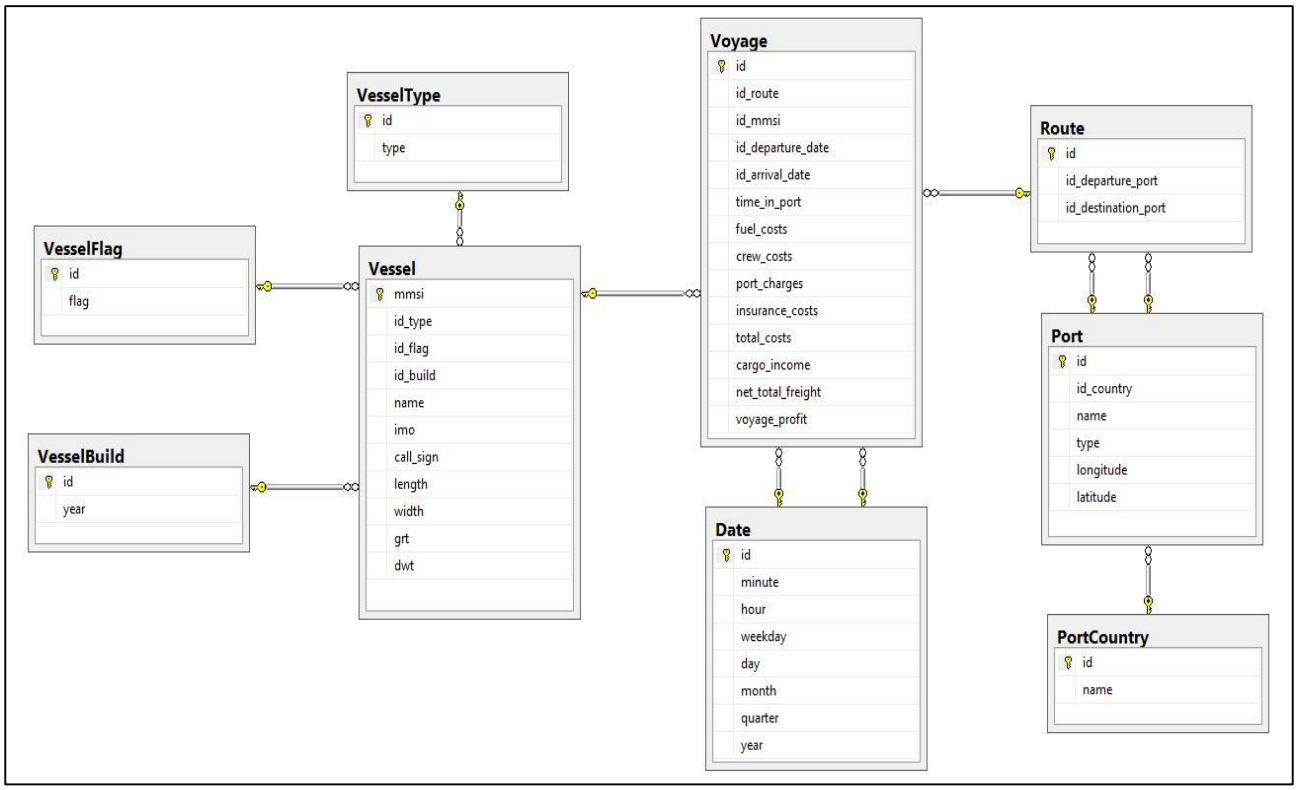


Рисунок 1 – Диаграмма хранилища данных.

Для оценки экономических показателей рейса судна были выбраны следующие характеристики: затраты на топливо, расходы на содержание экипажа, портовые расходы, расходы на страхование, общие расходы на весь рейс, доход от перевозки груза, чистый суммарный фрахт, прибыль, полученная за рейс.

При выполнении работы рассматривались два отношения: Длина судна и Прибыль рейса, Дедвейт и Затраты на топливо. Это было сделано для удобства сравнения конечных результатов работы различных алгоритмов кластеризации.

В проекте многомерных данных и интеллектуального анализа данных служб Analysis Service был создан многомерный куб, построенный на основе хранилища данных и применен встроенный алгоритм кластеризации данных.

Использование библиотеки scikit-learn для языка Python позволило рассмотреть следующие алгоритмы кластеризации:

– Иерархическая кластеризация. Были рассмотрены два алгоритма: одиночное связывание (Single Link) и полное связывание (Complete Link)

– Неиерархическая кластеризация. Рассмотрен алгоритм K-средних

При реализации алгоритма K-средних были использованы два способа выбора K - оптимального числа кластеров: метод «Локтя» (Elbow Method) и метод «Силуэта» (Silhouette Method).

Там же затронута тема предварительной обработки и масштабирования исходных данных.

Сравнивая инструментальные средства и алгоритмы кластеризации, можно составить две таблицы:

Таблица 1 – Обзор инструментальных средств.

Инструментальное средство	Обзор
Microsoft Analysis Services	Неизвестен встроенный метод кластеризации, мало инструментов для дальнейшего исследования и интерпретации результатов кластеризации
Библиотека scikit-learn для языка python	Позволяет работать как с иерархическими, так и с неиерархическими методами кластеризации, работу которых можно контролировать, а также строить различные графики с помощью сторонних библиотек

Таблица 2 – Обзор алгоритмов кластеризации.

Алгоритм кластеризации	Обзор
Иерархическая кластеризация (одиночное связывание (Single Link) и полное связывание (Complete Link))	Позволяет построить исчерпывающую иерархию возможных разбиений данных, которую можно легко исследовать с помощью дендрограмм.
Неиерархическая кластеризация (алгоритм K-средних)	Позволяет описывать кластеры с помощью их средних значений. Требует использования специальных методов для выбора K -оптимального числа кластеров.

ЛИТЕРАТУРА

1. Орешков В. И., Паклин Н. Б. Бизнес-аналитика. От данных к знаниям [Книга]. - Санкт-Петербург: «Питер», 2013.
2. Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python [Книга]. – Москва: «Вильямс», 2017.
3. Бринк Х., Ричардс Д., Феверолф М. Машинное обучение [Книга]. - Санкт-Петербург: «Питер», 2017.

УДК 004.773

Шапиев Т. К. (4 курс бакалавриата)
Дробинцев Д. Ф., ст. преподаватель

TELEGRAM BOT ДЛЯ ПОИСКА И ПОДБОРА ФИЛЬМОВ

Целью работы является создание бота-помощника, который будет искать фильмы по названиям и добавлять в библиотеку пользователя. А также имеется возможность подбора фильмов на основе выбранного. Это позволяет пользователю получать подборку фильмов основываясь на его библиотеке.

Особенности реализации:

При создании программы был сделан выбор использовать принципы SOLID [1], введенные Майклом Фэзерсом. SOLID – это 5 основных принципов объектно-ориентированного программирования и проектирования. Они помогают сделать системы легкими в поддержании и расширении в течение долгого времени.

В качестве базы данных фильмов было решено выбрать TMDb [2]. Это общедоступный API с огромным количеством различных фильмов. Также он обладает достаточным функционалом для поиска, сортировки и подбора фильмов.

В качестве клиентского приложения было выбрано приложение Telegram. Оно очень популярное среди пользователей и достаточно простое в использовании. Для взаимодействия с приложением создатели дали свободный доступ к их Telegram API [3]. Отправка сообщений пользователю и получение от него информации реализованы с помощью GET запросов.

Далее представлен пользовательский интерфейс:

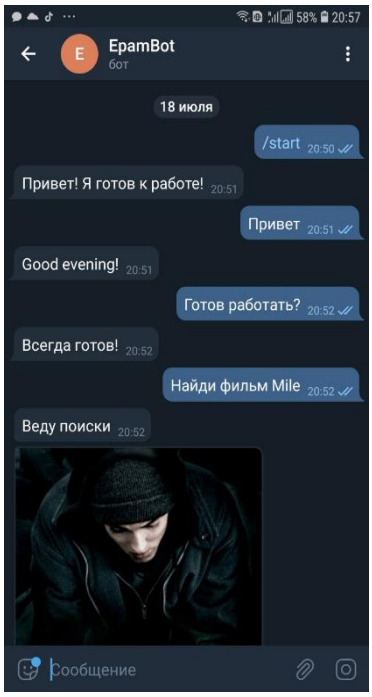


Рисунок 1– Поиск фильма.

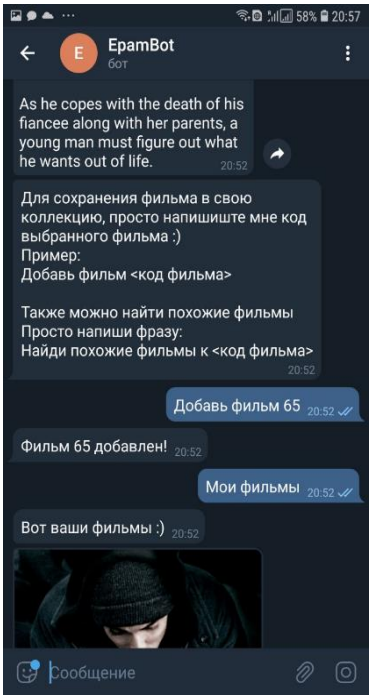


Рисунок 2– Добавление фильма в библиотеку.

Рисунок 3– Поиск похожих фильмов.

В качестве сравнения был выбран похожий бот под названием KinomanBot. Это отличный помощник, у которого множество достоинств, и он уже получил довольно большую популярность среди пользователей. Но главной идеей моего проекта был поиск фильмов по личным предпочтениям людей. И в данном пункте мой вариант имеет преимущество перед соперником, потому что KinimanBot не предоставляет такую возможность подбора фильмов.

Именно это является главным отличием моей реализации от существующих. Дальнейшим шагом для развития бота может стать добавление панели с кнопками для удобной эксплуатации.

ЛИТЕРАТУРА

1. Мартин Р., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C#. Символ-Плюс, 2011.
2. TMDb Api URL: https://developers.themoviedb.org/3/getting-started/introduction (дата обращения: 13/03/2021).
3. Telegram Api URL: https://core.telegram.org/bots/api#using-a-local-bot-api-server (дата обращения: 13/03/2021).

УДК 004.04

Алексеева Д.В. (4 курс бакалавриата)
Никифоров И.В., к.т.н., доцент

ОПТИМИЗАЦИЯ НАГРУЗКИ МУЛЬТИФРАКТАЛЬНОГО ТРАФИКА МЕЖДУ ГЕТЕРОГЕННЫМИ УЗЛАМИ РАСПРЕДЕЛЕННОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

В настоящее время популярность распределенных вычислений набирает обороты [2], что связано с возможностью решения трудоемких задач путем разбиения их на множество подзадач, не связанных между собой и применения параллельной обработки данных. Это позволяет практически полностью игнорировать задержки связи между компьютерами и сосредотачиваться только на времени сбора результатов. В связи с этим появляется проблема эффективного использования ресурсов для избежания ситуаций простоя или чрезмерной нагрузки на узлы вычислительной системы.

Целью работы является повышение эффективности использования вычислительных узлов распределенной системы за счет математического описания работы алгоритма балансировки нагрузки между узлами, отличительной особенностью которого является анализ конфигурации и состояния узлов в гетерогенном кластере и учет мультифрактальных свойств трафика.

Алгоритмы балансировки являются решением грамотного распределения нагрузки трафика в вычислительной системе [1]. Подобные алгоритмы должны определить, на какой узел назначить вычисление поступившей задачи и при необходимости перераспределить часть вычислений на сервера так, чтобы ни один из компьютеров в системе не простаивал и не был перегружен во избежание неравномерной нагрузки. Причиной неравномерной нагрузки на сервера выступают следующие проблемы:

– неоднородность вычислительной среды;

– различная степень сложности, а соответственно и запрашиваемой от узла мощности решаемых задач;

– различная пропускная способность каналов связи.

На основании текущего состояния системы алгоритмы балансировки можно разделить на статические и динамические [3]. Рассмотрим подробнее динамический алгоритм, так как статистический не учитывает текущее состояние узла, не является масштабируемым и гибким, что делает его пригодным для работы исключительно в однородной среде. В данной работе рассматривается гетерогенная вычислительная система, по этой причине использование динамического алгоритма, анализирующего расположение, текущее состояние и производительность узла на каждом этапе, является наиболее целесообразным.

Динамический подход к балансировке в свою очередь можно разделить на централизованный и децентрализованный (распределенный) [4]. Встает вопрос, каким способом осуществлять распределение ресурсов. В структуру централизированных алгоритмов входит планировщик (центральный элемент), задача которого на основе постоянной синхронизации собрать информацию о состоянии каждого узла для принятия решения. Распределенный алгоритм отличается локальностью каждого узла системы: компьютер самостоятельно планирует распределение нагрузки и построение траектории дальнейшей работы. Несмотря на уязвимость централизованного алгоритма из-за

необходимости постоянной синхронизации, зачастую он оказывается более эффективным подходом для решения некоторых прикладных задач, особенно в небольшой распределенной системе.

Трафик современных сетей все чаще вызывает слабо предсказуемые колебания нагрузки на сервер и экспоненциальный рост данных. Для обработки подобного мультифрактального трафика оказалось достаточно перспективно применять динамические механизмы балансировки в режиме, приближенному к реальному времени. Исследования доказывают [5], что достаточно часто трафик имеет самоподобные свойства, что влечет за собой потерю и задержку данных.

Самоподобность трафика проявляется в сохранении статистических характеристик при изменении масштаба времени, тем временем как мультифрактальный трафик является расширением самоподобного за счет учета масштабируемых свойств статистических характеристик второго порядка и выше. Для подобных ситуаций выполняется отношение: $M[|X(t)|^q] = c(q) \times t^{qh(q)}$ [1], где $c(q)$ – некоторая детерминированная функция, а $h(q)$ – обобщенный показатель Херста. Диапазон обобщенного показателя Херста $\Delta h = h(q_{min}) - h(q_{\max})$[2] характеризует неоднородность мультифрактального трафика. Увеличение диапазона Δh свидетельствует большом числе выбросов и неоднородности трафика.

Описание работы разрабатываемого алгоритма с учетом мультифрактальных свойств трафика выглядит следующим образом:

1. Выделяем окно фиксированной длины T в поступающем трафике;
2. Для выделенного участка подсчитываем диапазон значений обобщенного показателя Херста [2] и значение его функции;
3. Анализируем пропускную способность, состояние серверов с учетом гетерогенности системы, объем свободного ЦПУ и оперативной памяти конкретного сервера в текущий момент (обход серверов производится напрямую);
4. По результатам мультифрактальных свойств, полученных из п.2 вычисляем необходимое количество ресурсов.
5. Проводим распределение трафика по узлам на основе алгоритма балансировки;
6. Передвигаем окно длины T вперед на длину ΔT и анализируем следующий трафик.

Разработанная модель балансировки нагрузки учитывает мультифрактальные свойства трафика и обеспечивает статистически равномерное распределение нагрузки между узлами распределенной системы. Подобное решение снижает количество потерянных пакетов и время отклика и должно продемонстрировать высокие показатели производительности.

ЛИТЕРАТУРА

1. Е.И.Игнатенко. Адаптивный алгоритм мониторинга загруженности сети кластера в системе балансировки нагрузки. / Е.И.Игнатенко, В.И.Бессараб, И.В.Дегтяренко // Наукові праці ДонНТУ. Вип.21(183). 2011. С.95-102
2. Л.С.Курилов. Фрактальный принцип описания процессов динамики нагрузки в гетерогенной распределенной сетевой среде / Л. С. Курилов, А. Г. Финогеев // Информационные технологии в науке, образовании, телекоммуникации и бизнесе: тезисы докл. 33 Междунар. Конф. (Украина, Крым, Ялта – Гурзуф, 20-30 мая, 2006). – Ялта, 2006. – С. 40-42. – (Приложение к журналу «Открытое образование»)
3. Сафронов Д., Стоноженко К.М., Никифоров И.В., автоматическая балансировка нагрузки между потоковой обработкой данных и внутренними задачами кластера с использованием Kubernetes, Современные технологии в теории и практике программирования. сборник материалов конференции. Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. Санкт-Петербург, 2020. С. 165-167.
4. Шошмина И. В., Богданов А. В., Использование Grid-технологий для проведения вычислений / Вестник Санкт-Петербургского университета, ISSN: 1024-8579
5. Шелухин О. И., Причины самоподобия телетрафика и методы оценки показателя Херста – Режим доступа: https://cyberleninka.ru/article/n/prichiny-samopodobiya-teletrafika-i-metody-otsenki-pokazatelya-hersta

УДК 004

Абрамова Т.О. (2 курс магистратуры)
Тышкевич А.И., к.т.н., доцент

СОВЕРШЕНСТВОВАНИЕ МЕТОДОВ ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ ДЛЯ РАДИОНАВИГАЦИОННЫХ СИСТЕМ, ПРИМЕНЯЕМЫХ ДЛЯ ЗАДАЧ ПОЗИЦИОНИРОВАНИЯ

Морское судоходство развивается очень быстрыми темпами. Плотность движения судов растет, что приводит к необходимости в повышении точности систем, которые используются для навигации. Именно поэтому актуальным становится вопрос в исследовании систем с разделением каналов.

Целью работы является исследование алгоритмов, применяемых для позиционирования на воде в настоящее время и улучшение их для повышения точности местоположения. Было принято решение модифицировать метод цифровой обработки сигналов, основанный на смещении Гильберта, с целью повышения точности навигации в узкостях, фарватерах, каналах. Результатом данной работы является имитационная модель, которая позволяет оценить результат цифровой обработки сигналов радионавигационной системы с кодовым разделением каналов.

В качестве меры вычислительных затрат можно использовать порядок фильтра, поскольку порядок фильтра определят количество операции сложения и умножения. Искажения, вносимые усечённым фильтром Гильберта, приведут к тому, что в отрицательной области частот может наблюдаться неполное подавление спектра аналитического сигнала. Так как на практике использовать фильтр Гильберта бесконечного порядка невозможно, то при ограничении порядка фильтра Гильберта это приведёт к искажениям частотной характеристики фильтра по сравнению с идеальными характеристиками фильтра.

При увеличении порядка фильтра также увеличивается и подавление спектра аналитического сигнала в отрицательной области частот.

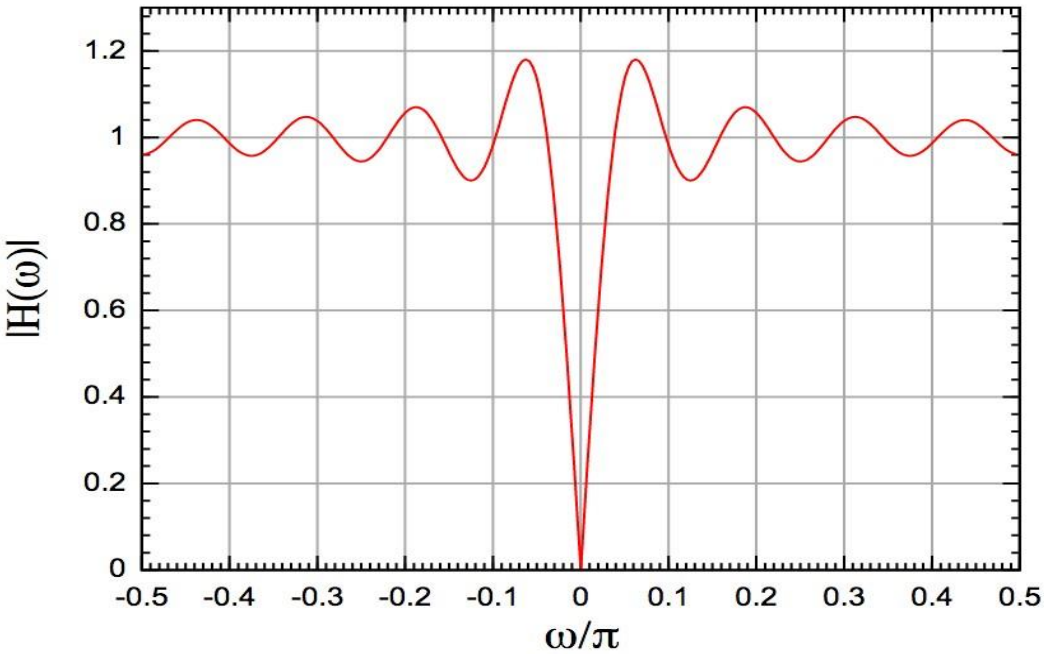


Рисунок 1 – АЧХ фильтра Гильберта 32 порядка

Выбор был остановлен на алгоритме Гильберта 80 порядка, поскольку нужно учитывать, что для комплексного сигнала вещественная и мнимая часть обрабатываются отдельно, что приводит к увеличивающимся вычислительным затратам алгоритма. Также для смещения спектра тоже нужно затратить некую вычислительную мощность.

В качестве среды имитационного моделирования остановимся на среде MATLAB. На Рисунке 2 представлена одна из частей системы.

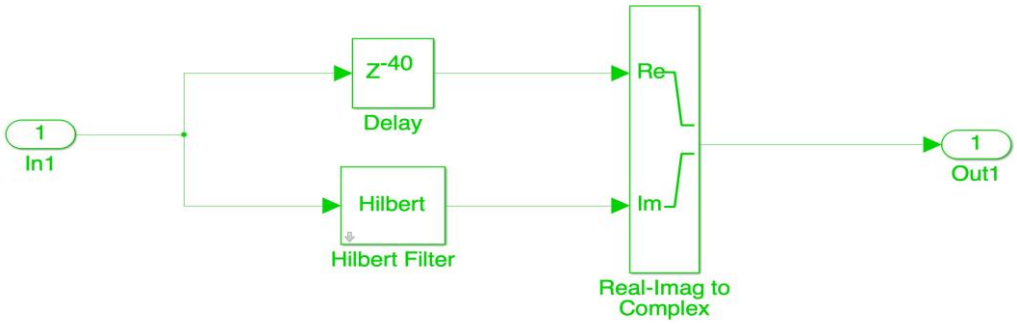


Рисунок 2 – Алгоритм формирования сигнала с одной боковой полосой на основе смещения Гильберта

В результате данной работы была выполнена разработка математической модели системы с кодовым разделением каналов, построенной на преобразовании Гильберта, а также выполнено имитационное моделирование разработанной системы с кодовым разделением каналов.

ЛИТЕРАТУРА

1. Никольский Б.А. Методы радионавигационных измерений. Часть 2: Учебн. пособие / Самар, гос. аэрокосм. ун-т. Самара, 2006,- 99 с.
2. Верещака А.И., Олянюк П.В. Радиоэлектроника, средства связи и радионавигации: Учебник для вузов,- М.: Транспорт, 1993,- 343 с.
3. Никольский Б.А. Активное радиоподавление. Учебное пособие. М.: САЙНС- ПРЕСС, 2007. - 80 с.: ил.
4. Каганов В. И. Радиотехника. Учебное пособие. Изд. центр «Академия», 2006- 352 с.

УДК 519.6

Болтунов А.А. (4 курс бакалавриата)
Черноруцкий И.Г., д.т.н., профессор

ИССЛЕДОВАНИЕ АЛГОРИТМА СТРЕКОЗЫ ДЛЯ РЕШЕНИЯ ЗАДАЧ ОПТИМИЗАЦИИ

Целью работы является исследование алгоритма стрекозы (DA, Dragonfly Algorithm) [1] для решения задач оптимизации и сравнение DA с другими популярными метаэвристическими методами оптимизации. Особенностью алгоритма DA является то, что он подражает поведению роя стрекоз в природе. Причиной их роения является либо миграция, либо охота (динамический и статический рой соответственно). В статическом рое небольшие группы стрекоз перемещаются по небольшой площади, чтобы охотиться на других насекомых. Поведение этого типа роения включает в себя локальные движения и резкие изменения. Однако при динамическом роении огромное количество стрекоз образует единую группу и движется в одном направлении на большое расстояние. Направления траекторий движения стрекоз задается при помощи пяти весов. Так же веса используются для настройки баланса разведки и эксплуатации.

Одной из основных проблем исследуемого алгоритма является преждевременная сходимость к локальным оптимумам. А преимуществом этого алгоритма является простота и гибкость использования благодаря возможности настройки весов.

Статическое и динамическое поведение роя, соответствует фазам исследования и эксплуатации алгоритма метаэвристической оптимизации. Для направления искусственных

стрекоз по различным траекториям использовались пять весов: разделительный вес (s), выравнивающий вес (a), когезионный вес (c), пищевой фактор (f), вражеский фактор (e) и инерционный вес (ω). Для исследования пространства поиска используются высокие веса выравнивания и низкой когезии, однако для использования пространства поиска можно использовать низкие веса выравнивания и высокой когезии. Для перехода между разведкой и эксплуатацией использовались радиусы окрестностей, увеличенные пропорционально числу итераций. Адаптивная настройка весов роя (s, a, c, f, e, и ω) в процессе оптимизации – это еще один способ сбалансировать разведку и эксплуатацию. Математически каждый из вышеупомянутых весовых коэффициентов показан в уравнениях (1) – (5).

Разделение рассчитывается следующим образом:

$$S_i = -\sum_{j=1}^{N} X - X_j \tag{1}$$

В уравнении (1) X обозначает положение текущей особи, X_j – положение j-й соседней стрекозы, N – число отдельных соседей стрекозиного роя, а S – движение разделения для i-й особи.

Уравнение (2) используется для расчета выравнивания:

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N} \tag{2}$$

где A_i – движение выравнивания для i-го индивидуума и V – скорость j-й соседней стрекозы.

Когезия была выражается следующим образом:

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X \tag{3}$$

где Ci – когезия для i-го индивидуума, N – размер окрестности, X_j – положение j-й соседней стрекозы, а X – текущий индивид стрекозы.

Притяжение к пище вычисляется согласно уравнению:

$$F_i = X^+ - X \tag{4}$$

где Fi – притяжение пищи для i-й стрекозы, X^+ – положение источника пищи, а X – положение текущей особи стрекозы. Здесь пища – это стрекоза, которая имеет лучшую целевую функцию.

Отвлечение хищников вычисляется согласно уравнению:

$$E_i = X^- + X \tag{5}$$

где – E_i отвлечение противника для i-й особи, X^- – позиция противника, а X – позиция текущей особи стрекозы.

Для обновления позиции в пространстве поиска искусственные стрекозы используют два вектора: шаговый вектор ΔX и позиционный вектор X. Вектор шага является аналогией вектора скорости в алгоритме PSO. Обновление позиций также основано в основном на структуре алгоритма PSO. Шаг вектора определяется в следующим образом:

$$\Delta X_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + \omega \Delta X_t \tag{6}$$

где S_i представляет собой расстояние для i-й стрекозы; A_i – это выравнивание i-й стрекозы; C_i представляет сплоченность для i-й стрекозы; F_i – представляет собой источник пищи для i- го индивидуума; E_i – представляет позицию противника для i-й стрекозы; ω – инерционный вес; а t указывает на счетчик итераций.

Когда расчет вектора шага завершен, расчет векторов положения начинается следующим образом:

$$X_{t+1} = X_t + \Delta X_{t+1} \tag{7}$$

где t указывает текущую итерацию.

Для того чтобы повысить вероятность исследования всего пространства решений оптимизационным алгоритмом, в методику поиска необходимо добавить случайное

перемещение. Когда нет соседних решений, чтобы увеличить случайность, стохастическое поведение и исследование искусственных особей стрекозы, стрекозы должны использовать случайное блуждание (полет Леви), чтобы летать по всему пространству поиска.

Результаты тестирования. Алгоритм был написан с использованием MATLAB и сравнивался с алгоритмами оптимизации светлячка (FA, Firefly Algorithm), роя частиц (PSO, Particle Swarm Optimization) и дифференциальной эволюции (DE, Differential Evolution). Сравнение проводилось с использованием тестовых функций [2], среди которых были: простая квадратичная функция, квадратичный функционал высокой степени овражности, функция Розенброка.

При тестировании количество стрекоз в популяции было равно 100, и максимальное количество итераций 300. Статистика собиралась с 20 запусков алгоритма.

Алгоритм стрекозы успешно справляется со всеми тестами кроме теста на функционале с высокой степенью овражности. Это связано с тем, что незначительные изменения во входящих данных этого функционала приводят к значительным изменениям выходящих данных. По этой причине многие методы оптимизации не могут справиться с задачами с высокой степенью овражности.

Выводы. Метод оптимизации стрекозы хорошо справляется с большей частью задач, но решение задач с высокой степенью овражности вызывает трудности. Проблема решения задач с высокой степенью овражности характерна для многих метаэвристических алгоритмов оптимизации, основанных на роении. Однако остальные тестовые задачи алгоритм решил быстро и достаточно точно относительно других метаэвристических алгоритмов, рассматриваемых при тестировании.

ЛИТЕРАТУРА

1. Chnoor M. Rahman, Tarik A. Rashid, Dragonfly Algorithm and Its Applications in Applied Science Survey // Computational Intelligence and Neuroscience. – 2019.
2. Черноруцкий И.Г. Методы принятия решений. / И.Г. Черноруцкий – Спб.: БХВ Петербург, 2005. – 417 с.

УДК 004.42

Варламов Д.А. (4 курс бакалавриата)
Шошмина И.В., к.т.н., доцент

МОДЕЛИРОВАНИЕ И ФОРМАЛЬНАЯ ВЕРИФИКАЦИЯ РАСПРЕДЕЛЕННОЙ ТРАНЗАКЦИОННОЙ ПАМЯТИ CUDA-DTM

Транзакционная память позволяет упростить параллельное программирование, выделяя целые группы инструкций в атомарные транзакции. Программные реализации транзакционной памяти не удается сделать достаточно эффективными, для разработки аппаратных реализаций требуется изменение аппаратной части. В статье [2] была предложена модель и реализация распределенной транзакционной памяти для кластеров на графических процессорах CUDA-DTM. Корректность алгоритмов модели CUDA-DTM в статье [2] не рассматривалась. Известно, что алгоритмы транзакционной памяти очень сложны и в них легко совершить ошибку. Наша цель – формально верифицировать поведение CUDA-DTM с помощью метода проверки модели относительно свойств взаимно исключающего доступа, отсутствия блокировок, сериализуемости транзакций. Для верификации мы используем SPIN[5] и TLC Model Checker [6].

Первым шагом в нашей работе стало создание модели поведения транзакционной памяти CUDA-DTM. Формально модель является структурой Крипке [4], помеченной системой переходов. Для построения формальной модели нами выявлены 3 основных составляющих CUDA-DTM: внутренние структуры данных, методы (функции) и особенности, которые вносит в ее устройство графический процессор (GPU). Внутренние структуры данных – общая

разделяемая память, транзакция, теневая запись/сущность (Shadow Entry). К методам внутри CUDA-DTM относятся: запись идентификатора транзакции id в теневую сущность, создание вложенной транзакции, валидация транзакции, прерывание транзакции, завершение транзакции, попытка завладеть участком общей разделяемой памяти. В ходе последнего алгоритма транзакция пытается завладеть участком памяти, находящемся на локальном узле. В случае отсутствия искомого участка памяти на локальном узле транзакция создает вложенную транзакцию – транзакцию, пытающуюся получить доступ к той же ячейке памяти, что и оригинальная транзакция, но исполняемую на том узле, где эта ячейка памяти находится. Не все методы описаны в [2] достаточно подробно, поэтому чтобы их восстановить, мы обратились к статье [1]. В статье [1] было описано применение транзакционной памяти для графических процессоров и указаны основные свойства, которым эта модель должна удовлетворять, однако реализации распределенной транзакционной памяти там не приводилось. В [1] также, как в [2], отсутствуют формальные доказательства свойств. Некоторые алгоритмы [2] отличаются от алгоритмов в [1]. К особенностям CUDA-DTM, связанным с графическими процессорами, относится то, что вся общая разделяемая память распределена между всеми GPU в кластере. Каждый GPU имеет доступ только к определенным ячейкам этой памяти. Если нужной ячейки в его локальной памяти не находится, он создает вложенную транзакцию.

Проверять модель памяти CUDA-DTM мы будем на следующие свойства: отсутствие блокировок (deadlock), взаимно исключающий доступ к общим ресурсам и сериализуемость транзакций. Отсутствие блокировок означает отсутствие ситуаций, при которых каждый процесс из группы параллельно работающих процессов ожидает некоторого события. Взаимно исключающий доступ означает, что в любой момент времени доступ к критической секции памяти может иметь не более чем один процесс. Транзакционная память позволяет организовать обращение разных процессов к переменным в режиме отсутствия ожидания (wait-free). Если оба процесса хотят обратиться (прочитать или записать) к одной и той же переменной, то ни один из процессов не ожидает другого. Для того чтобы организовать такую работу с памятью корректно, требуются сложные алгоритмы синхронизации транзакций. Ключевым свойством работы транзакционной памяти является сериализуемость транзакций. Сериализуемость транзакций означает, что для каждого вычисления транзакционной памяти признанные транзакции (committed transactions) могут быть упорядочены так, что транзакции выполняются последовательно, то есть одна транзакция следует за другой, без чередования. При упорядочивании транзакций принимаются во внимание значения переменных, читаемых из памяти. Сериализуемость включает атомарность транзакции, т.е. гарантирует, что эффект транзакции проявляется между ее первым вызовом и последним откликом. В частности, сериализуемость гарантирует, что если транзакция A следует за транзакцией B, то читаемое A значение переменной будет не менее свежим, чем значение этой переменной, прочитанное транзакцией B.

Мы описали модель CUDA-DTM на языке Promela. Нами были приняты следующие ограничения: Spin позволяет верифицировать только конечные модели, поэтому нам пришлось ограничить количество узлов в кластере и размер общей разделяемой памяти константами N=2 и size=4. В модели была ограничена вложенность транзакций: на каждом узле одновременно может быть создана только одна собственная транзакция, которая вызывает одну вложенную транзакцию на другом узле, который содержит необходимую ячейку памяти. В модели реализован недетерминированный выбор раздела памяти, к которому будет обращаться узел. Коммуникация между узлами реализована с помощью каналов в Promela, которые имитируют реальную коммуникацию между узлами кластера. Наша модель была проверена системой верификации Spin на свойства свободы от блокировок и свойство взаимно исключающего доступа mutualex, заданное нами на языке линейной темпоральной логики LTL. Синтаксис и семантику LTL можно найти, например, в [4]. В результате верификации наша модель удовлетворила всем заданным свойствам.

Для задания свойства сериализуемости логики LTL недостаточно. Нам необходима логика QLTL, так как она позволяет нам использовать кванторы, без которых мы не сможем сформулировать свойство сериализуемости. Язык формальной спецификации TLA+, разработанный Лэсли Лэмпортом и описанный им в его книге [3], позволяет нам задать свойство сериализуемости и проверить его при помощи инструмента проверки моделей TLC. Процесс моделирования происходил в интегрированной среде разработки TLC Model Checker.

На данный момент разработана модель CUDA-DTM в SPIN и формально доказано, что она удовлетворяет свойствам отсутствия блокировок и взаимно исключающего доступа. Также создана модель в интегрированной среде разработки TLA Toolbox[6] и сформулировано свойство сериализуемости. Результаты проверки модели CUDA-DTM на свойство сериализуемости будут представлены на конференции.

ЛИТЕРАТУРА

1. A. Holey and A. Zhai. Lightweight software transactions on gpus. In 2014 43rd International Conference on Parallel Processing, pages 461–470, 2014.
2. Irving, Samuel and Chen, Sui and Peng, Lu and Busch, Costas and Herlihy, Maurice and Michael, Christopher CUDA-DTM: Distributed Transactional Memory for GPU Clusters // Networked Systems. NETYS 2019, Lecture Notes in Computer Science. Springer. 2019.
3. Lamport, L. Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers. — Addison-Wesley, 2003.
4. Карпов Ю. Г. Model Checking. Верификация параллельных и распределенных программных систем. — СПб.: БХВ-Петербург, 2009.
5. SPIN website [Электронный ресурс]. – URL: http://spinroot.com/
6. The TLA+ Home Page [Электронный ресурс]. – URL: https://lamport.azurewebsites.net/

УДК 004.891.2

Виноградов П.А. (2 курс магистратуры)
Дробинцев Д.Ф., ст. преподаватель

ПРОГНОЗИРОВАНИЕ ИЗМЕНЕНИЙ ЦЕН АКЦИЙ ФОНДОВОГО РЫНКА С ПОМОЩЬЮ РЕКУРРЕНТНОЙ НЕЙРОННОЙ СЕТИ И АНАЛИЗА ТОНАЛЬНОСТИ ТЕКСТА

Целью работы является реализация системы для прогнозирования изменения цен акций фондового рынка при помощи различных подходов в реализации нейронных сетей и анализа тональности текста, а, в частности, новостей, которые могут влиять на изменение стоимости ценной бумаги. В данной системе была выбрана рекуррентая нейронная сеть[1], поскольку она наиболее часто используется для анализа настроения текста. Используемый в работе подход объединяет в себе как технический, так и фундаментальный анализы. Поскольку универсальную систему точного прогнозирования построить невозможно, для анализа был выбран только технологический сектор. Система была реализована на языке Python с использованием различных библиотек.

Для создания тренировочной выборки требовались новости о технологическом секторе в целом и об отдельно взятой компании. Чтобы собрать эти данные был использован API (application programming interface) NewsAPI[2], который может предоставлять статьи по ключевым словам из 128 различных новостных источников, вместе с заголовками, самой статьей и датой публикации, которая пригодится для формирования выборки.

Помимо новостей требовалось также собрать данные о ценах акции на момент закрытия рынка за определенный период времени. Для этого была использована библиотека pandas-datareader[3], с помощью которой можно получать данные из таких источников, как Google, Yahoo! Finance, Всемирный Банк и многие другие. В реализованной системе были взяты

данные из Yahoo! Finance. Полученные данные представляют собой объект DataFrame, строки которого – это даты, а столбцы – это цена открытия, цена закрытия, максимальная цена и минимальная цена для каждой конкретной даты.

Поскольку не было найдено сервисов, которые предоставляют данные по ценным бумагам на момент отдельно взятого времени публикации какой-либо новости, мы можем лишь оценивать разницу между ценой открытия и ценой закрытия и влияние всех новостей о компании и секторе вместе на итоговую цену.

Для очистки данных из текста статей сначала были убраны все знаки препинания и переносов строк, далее все знаки были переведены в нижний регистр. Помимо этого с помощью библиотеки nltk[4] и ее классов stopwords и PorterStemmer удалось избавиться от артиклей, местоимений и окончаний в словах, т.е. классификация слов приняла более универсальный вид.

В итоге после сбора и очистки данных был собран единый DataFrame, который представлял собой общие данные по 12 компаниям. По разнице цены закрытия и цены открытия можно судить о положительном или отрицательном влиянии новостей в данный день на изменение цены. На этом первый этап работы был завершен.

Следующий этап заключается в разработке рекуррентной нейронной сети[6]. Для нее была выбрана классификация «многие ко многим». Инициализация нейронной сети происходит с тремя параметрами веса и двумя смещениями.

В фазе прямого распространения нейронной сети были использованы и реализованы следующие формулы:

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

$$y_t = W_{hy}h_t + b_y$$

Где W_{ii} – наборы параметров весов, b_i – смещение, h_i – скрытые состояния сети.

В фазе обратного распространения нейронной сети была использована функция потери перекрестной энтропии:

$$L = -\ln(p_c)$$

Где p_c – предсказуемая вероятность рекуррентной нейронной сети. Таким образом, нейронная сеть может использовать градиентный спуск для минимизации потерь[5]. После того, как подсчитаны все градиенты на каждом шагу, происходит обновление параметров веса и смещения.

В результате была смоделирована рекуррентная нейронная сеть, которая принимает на вход список из текстов, на основе которых выстраивается предположение о том, как они повлияют на изменение цены акции.

Были получены следующие результаты по прогнозированию изменения цен на акцию компании Apple при соотношении обучающей и тестовой выборок 80/20:

Количество проходов	Функция потери		Точность	
100	Обучающая	0.650	Обучающая	0.652
	Тестовая	0.718	Тестовая	0.541
500	Обучающая	0.304	Обучающая	0.811
	Тестовая	0.372	Тестовая	0.698
1000	Обучающая	0.122	Обучающая	0.901
	Тестовая	0.209	Тестовая	0.820

В итоге полученные результаты показали максимальную точность в 82%, что означает, что для 82% выбранных для анализа статей было успешно предсказано – упадет акция или будет расти. Это не является высоким показателем, но из которого можно сделать следующие

выводы: а) системе требуется доработка: выборка новостей производилась за малый интервал времени, что могло сказаться на точности; требуется более тщательная очистка входных данных, чтобы нейронная сеть более точно классифицировала новости; нужны более подробные данные по ценам акций в течение дня. б) выборка новостей могла быть либо неудачной, либо выбранные новости не коррелируют с ценой на акцию.

Полученные результаты не могли и никогда не смогут гарантировать 100%-ную точность, поскольку изменение цены зависит от множества других факторов, таких как политика, резкое повышение объема спекулятивных операций, психологическое поведение инвесторов и многие другие.

ЛИТЕРАТУРА

1. Predict Stock Prices Using RNN. [Электронный ресурс]. URL: https://lilianweng.github.io/lil-log/2017/07/08/predict-stock-prices-using-RNN-part-1.html (дата обращения 16.01.21)
2. NewsAPI. Python client library. [Электронный ресурс]. URL: https://newsapi.org/docs/client-libraries/python (дата обращения 15.01.21)
3. Pandas-datareader. Python library. [Электронный ресурс]. URL: https://pandas-datareader.readthedocs.io/en/latest/ (дата обращения 15.01.21)
4. NLTK documentation. [Электронный ресурс]. URL: https://www.nltk.org/ (дата обращения 16.01.21)
5. Попова Ю.Б., Яцынович С.В. «Моделирование и программная реализация искусственной нейронной сети методом обратного распространения ошибки», 2017, Математические методы в технике и технологиях – ММТТ, т.12, стр. 168-171 (дата обращения 21.01.21)
6. A Technical Guide on RNN/LSTM/GRU for Stock Price Prediction/ [Электронный ресурс]. URL: https://medium.com/swlh/a-technical-guide-on-rnn-lstm-gru-for-stock-price-prediction-bce2f7f30346 (дата обращения 16.01.21)

УДК 004.85

Ворошилов М.К. (2 курс магистратуры)
Дробинцев П.Д., к.т.н., доцент

ПРОГНОЗИРОВАНИЕ ТРЕНДА КОТИРОВОК ФИНАНСОВЫХ ИНСТРУМЕНТОВ С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

Изучая фондовый рынок, всегда необходимо задаваться вопросом о его будущем состоянии. Чтобы добиться эффективности финансовых вложений, необходимо проводить многоуровневый анализ фондового рынка, реализуемый в инвестиционной стратегии. Хотя предсказать точное изменение котировок со стопроцентной вероятностью невозможно, актуальность этой проблемы стимулирует активное развитие программных и аппаратных средств, позволяющих собирать, хранить и анализировать большие объемы данных. Существует два основных метода анализа, применяющихся в зависимости от объёма и срока инвестирования: фундаментальный анализ и технический анализ.

Фундаментальный анализ применяется для исследования финансово-экономического состояния отраслей, отдельных компаний и их инвестиционной привлекательности. Компании привлекательны, если их акции устойчивы в долгосрочной перспективе, а у самих компаний есть потенциал развития. Иными словами, в результате фундаментального анализа можно понять, акции определённой компании переоценены или недооценены в данный момент времени. Он наиболее популярен при долгосрочных инвестициях[1].

Технический анализ позволяет прогнозировать изменение цен в будущем на основании хронологических данных об их изменениях в прошлом. Самый распространённый пример технического анализа – свечной анализ, основывающийся на графике «японские свечи». Благодаря различным формам и комбинациям свечей на графике, отражающих множество

параметров, таких как цена открытия и закрытия, максимум и минимум, можно анализировать тренды и достаточно эффективно предсказывать дальнейшее движение цен. Технический анализ позволяет определять лучший момент для заключения сделок и наиболее популярен при краткосрочной торговле[1].

Машинное обучение в финансовой сфере неотъемлемо связано с фундаментальным анализом, который предполагает использование исторических данных котировок для построения прогноза. Исследования проводятся как с точки зрения задачи получения конкретных значений и проверки их на точность, так и с точки зрения задачи классификации. Также учитывается цикличность фондовых рынков, то есть повторяющиеся движения различной продолжительности. Подобные колебания могут быть определены как шаблоны поведения. Выявление паттернов может увеличить эффективность прогнозирования, так как алгоритмы машинного обучения будут готовы к закономерным изменениям показателей цен[2].

Концепция данной работы предполагает решение задачи классификации (обучение с учителем) в постановке рост/падение. Существует несколько популярных алгоритмов классификации, такие как «деревья решений», «наивный Байес», «метод опорных векторов» (SVM). Из более сложных подходов можно выделить «многослойный перцептрон» (MLP) и нейронные сети с долгой краткосрочной памятью (LSTM). Выбор был сделан в пользу метода опорных векторов, так как он достойно зарекомендовал себя в данной сфере, зачастую даёт более точные результаты при краткосрочной торговле. Визуализация данных SVM проста для понимания. После выбора алгоритма необходимо обучить модель на обучающей выборке. Реализация будет использовать данные от провайдера финансовой информации «Yahoo! Finance». Простого импорта данных зачастую недостаточно, их необходимо подготовить для более эффективного обучения. Одним из самых известных приёмов является метод скользящего окна. Под окном в данном случае понимается временной интервал, содержащий набор значений, которые используются для формирования обучающего примера[3]. В процессе построения модели подбираются оптимальные значения гиперпараметров. После обучения производится проверка модели на тестовой выборке, сравнивается полученный результат с реальными изменениями котировок за тот же отрезок времени, производится анализ результатов.

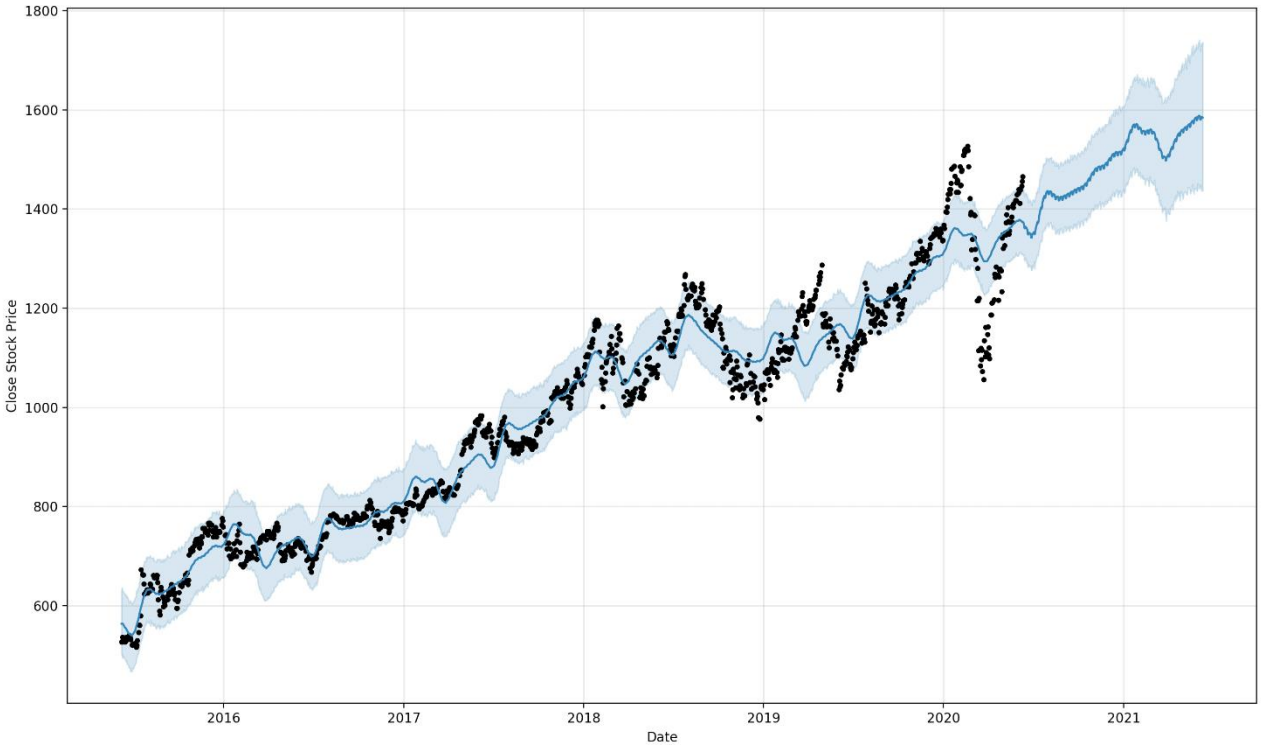


Рисунок 1 – Прогнозирование цены закрытия на основе исторических данных

Множество исследований демонстрируют успешное применение технологий машинного обучения в реализации программных проектов, а часть из них применялась как полноценный инструмент при торгах на бирже. Универсальных алгоритмов для достижения максимальных результатов на любых наборах данных не существует, каждый подход может показывать себя лучше или хуже в зависимости от реализации. Несмотря на это, крупные игроки рынка информационных технологий не остаются в стороне и предлагают свои инструменты для прогнозирования. Например, библиотека «Prophet» от Facebook специально адаптирована под работу с фондовым ценных бумаг[4]. Множество подобных продуктов используют алгоритмы машинного обучения для проведения технического анализа.

Применение метода опорных векторов к временным рядам демонстрирует высокий потенциал возможностей для точного предсказания тренда изменения котировок. При этом полагаться исключительно на алгоритмы не стоит, так они зачастую не учитывают внешние факторы среды, например, доверие инвесторов к конкретным активам. Сегодня профессиональные решения в этой области можно считать хорошим инструментом для инвестора, но до выхода на первый план полностью автоматизированной торговли без участия человека ещё далеко.

ЛИТЕРАТУРА

1. Isaac Kofi Nti, Adebayo Felix Adekoya and Benjamin Asubam Weyori “A systematic review of fundamental and technical analysis of stock market predictions”, Artificial Intelligence Review vol. 53, August 2019.
   DOI: 10.1007/s10462-019-09754-z
2. Seungwoo Jeon, Bonghee Hong, Juhyeong Kim and Hyun-jik Lee, “Stock Price Prediction based on Stock Big Data and Pattern Graph Analysis”, International Conference on Internet of Things and Big Data, January 2016.
   DOI: 10.5220/0005876102230231
3. Sliding Window Algorithm Basic Information [Электронный ресурс]. URL: https://riptutorial.com/algorithm/example/25071/sliding-window-algorithm-basic-information/ (дата обращения 14.03.2021)
4. Prophet | Forecasting at scale [Электронный ресурс]. URL: https://facebook.github.io/prophet/ (дата обращения 14.03.2021)

УДК 519.6

Григорьев А.А. (4 курс бакалавриата)
Черноруцкий И.Г., д.т.н., профессор

ПРОБЛЕМА ЖЁСТКОСТИ НА ПРИМЕРЕ АЛГОРИТМА ПЧЕЛИНОЙ КОЛОНИИ

Проблема жёсткости на примере алгоритма пчелиной колонии

Целью работы является исследование возможностей алгоритма пчелиной колонии при решении жёстких (овражных) задач.

Жёсткой или овражной считается задача, если отвечающая ей система дифференциальных уравнений – жёсткая.

Идея алгоритма основана на поведении роя пчёл. Рой содержит в своём составе три типа пчёл: рабочие, наблюдатели и разведчики. Работая вместе, пчёлы проводят поиск источников пищи. Каждая рабочая пчела проводит поиск положения нового источника пищи, коммуницируя с другими пчёлами. Если найдено положение нового источника пищи, которое лучше старого, то рабочая пчела запоминает его. Затем наблюдатели выбирают источник пищи для исследования, используя информацию, полученную от рабочих пчёл. Если найдено новое лучшее положение, пчела-наблюдатель запоминает эту позицию. Рабочие пчёлы, чьи

источники пищи были покинуты на определённое время, становятся пчёлами-разведчиками с новыми позициями источников пищи для следующего цикла поиска.

Плюсами данного алгоритма являются:

- Простота и гибкость
- Использование меньшего количества контролируемых параметров по сравнению с другими алгоритмами
- Лёгкое слияние (гибридизация) с другими оптимизационными алгоритмами
- Лёгкость в реализации с помощью базовых математических и логических операторов

Алгоритм состоит из 4 фаз:

- Инициализация алгоритма
- Фаза рабочих пчёл
- Фаза пчёл-наблюдателей
- Фаза пчёл-разведчиков

Инициализация алгоритма. Определяется общее количество пчёл, из которых 50% - рабочие пчёлы, 50% - наблюдатели. Задаётся количество попыток для улучшения решений. Генерируются случайные начальные источники пищи (решения) для рабочих пчёл, используя следующее выражение:

$$x_{x,j} = L_j + \phi_{ij}(U_j - L_j) \tag{1}$$

где $i = 1, 2, 3, \ldots, NP$ – количество рабочих пчёл, $j = 1, 2, 3, \ldots, D$ – размерность задачи, ϕ_{ij} - случайное число в диапазоне $(0,\ 1)$, U_j, L_j - верхняя и нижние границы поиска для данного измерения.

Фаза рабочих пчёл. Для каждой рабочей пчелы генерируется новый источник пищи (решение), используя следующее выражение:

$$v_{i,j} = \begin{cases} x_{i,j} + \phi_{ij}(x_{i,j} - x_{k,j});\ j = j^*, \\ x_{x,j};\ j \neq j^*, \end{cases} \tag{2}$$

где k – целое случайное число в диапазоне от 1 до NP, $k \neq i$, j^* - целое случайное выбранное число в диапазоне от 1 до D, ϕ_{ij} - случайное число в диапазоне [-1, 1]. Новое решение v_i оценивается и сравнивается со старым x_i. Если $f(v_i) < f(x_i)$, заменяем x_i на v_i. В противном случае x_i остаётся без изменений, при этом увеличиваем на 1 значение счётчика попыток улучшения решения.

Фаза пчёл-наблюдателей. Наблюдатели выбирают источники пищи в зависимости от их качества, основанного на значениях вероятностей $p(i)$, вычисляемых по следующей формуле:

$$p(i) = \frac{fit(x_i)}{\sum_{j=1}^{NP} fit(x_i)}, \tag{3}$$

где

$$fit(x_i) = \begin{cases} \dfrac{1}{1 + f(x_i)};\ f(x_i) \geq 0, \\ 1 + |f(x_i)|;\ f(x_i) < 0 \end{cases} \tag{4}$$

Если случайное число в диапазоне (0,1) окажется меньше, чем $p(i)$, тогда генерируется новое решение v_i с помощью выражения (2). Если $f(v_i) < f(x_i)$, заменяем x_i на v_i. В противном случае x_i остаётся без изменений, при этом увеличиваем на 1 значение счётчика попыток улучшения решения.

Фаза пчёл-разведчиков. Если решение x_i не может быть улучшено через заданное нами количество попыток, то генерируем новое решение x_i, используя выражение (1), сбрасывая при этом счётчик для данного решения.

После данных 4 фаз запоминаем x_{best}, f_{best} (лучшее решение). Повторяем фазы, пока не достигнем критерия остановки.

Тестирование алгоритма на жёстких задачах. Алгоритм был реализован на языке C++ (11) в среде разработки Code::Blocks 17.12. Тестирование проводилось на следующих функциях:

- Функция Розенброка

$$F_2 = 100(x_1^2 - x_2)^2 + (1 - x_1)^2;$$

- Функция Пауэлла

$$F_4 = (x_1 + 10x_2^2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

- Квадратичный функционал с высокой степенью овражности $\eta = 10^{12}$

$$F_7 = 1/2\sum_{i}^{4}\lambda_i\langle x_i, u_i\rangle - \langle b, x\rangle; b = (1;1;1;1); \lambda_1 = 10^8; \lambda_2 = \lambda_4 = 10^{(-4)}; \lambda_3 = 10^6;$$

$$u_1 = \left(\frac{1}{\sqrt{3}}\right)(1;-1;1;0); u_2 = \left(\frac{1}{\sqrt{6}}\right)(1;2;1;0); u_3 = \left(\frac{1}{\sqrt{3}}\right)(1;0;-1;1); u_4 = \left(\frac{1}{\sqrt{6}}\right)(1;0;-1;-2)$$

При сравнении результатов работы алгоритма пчелиной колонии с результатами, полученными при использовании стандартных алгоритмов (GZ1, SPACR, SPAC1, SPAC5, SIMPL), можно сделать вывод, что алгоритм пчелиной колонии не способен решать жёсткие задачи за приемлемое время и небольшие трудозатраты. Однако он может показать хорошие результаты при сужении области поиска решения или слиянии с другим алгоритмом. Также алгоритм может применяться в решении других различных проблем: проблема коммивояжёра, раскраска графа, обработка изображений и т. д.

ЛИТЕРАТУРА

1. Ankit Choubey, G. L. Prajapati // An understanding of ABC algorithm and its applications // 2015 // troinda.in: technical research organization India // URL: http://troindia.in/journal/ijcesr/vol2iss10/105-109.pdf // Дата обращения: 29.11.2020
2. И. Г. Черноруцкий, Н. В. Воинов, Л. П. Котлярова // Методы оптимизации. Учебное пособие // Санкт-Петербург 2020 // Дата обращения: 30.11.2020

УДК 004.891

Дроздов С.В. (4 курс бакалавриата)
Уткин Л.В., д.т.н., профессор

МЕТОД ИНТЕЛЛЕКТУАЛЬНОЙ КЛАССИФИКАЦИИ ГИСТОЛОГИЧЕСКИХ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ПРОСТРАНСТВЕННОГО ПОЛОЖЕНИЯ ОТДЕЛЬНЫХ ЭЛЕМЕНТОВ

Целью работы является исследование методов и алгоритмов компьютерной диагностики для анализа гистологических изображений, и разработка алгоритма обучения с частичной разметкой образцов. Метод и алгоритм должен решать задачу автоматизированной разметки и классификации гистологического изображения.

Актуальность работы: Интенсивное развитие моделей машинного обучения открывает новые возможности гистологического исследования. Однако трудоемкость разметки гистологических образцов не позволяет создавать полноценные датасеты для обучения

моделей, например, глубоких нейронных сетей. Поэтому разработка методов и алгоритмов автоматизированной разметки образцов представляется весьма актуальной.

Основная идея предлагаемого метода заключается в использовании дополнительной информации, которая определяется пространственным расположением отдельных клеток гистологии. Это позволяет реализовать параллельно несколько вариантов процедур кластеризации, что приводит к композиции алгоритмов и повышению точности разметки и классификации гистологических образцов. Пространственное расположение определяется анализом k ближайших клеток – соседей, т.е., каждая процедура кластеризации работает с k элементами, рассматриваемыми в кластеризации как единое целое.

Используемые алгоритмы:

Автокодировщик – разновидность архитектуры нейронной сети, целью которой является получение на выходе данных, идентичных данным на входе. Состоит из двух основных компонентов: кодирующего и декодирующего. Скрытые слои автокодировщика должны иметь меньше нейронов, чем входной. Таким образом кодирующий компонент уменьшает размерность исходных данных, а декодирующий компонент восстанавливает данные из представления меньшей размерности. За счёт этого нейронная сеть обучается определять закономерности во входных данных. Его главная цель – уменьшить размерность элементов гистологических образцов.

K-means – алгоритм неиерархической кластеризации, который разделят выборку на n непересекающихся групп с равной дисперсией, минимизируя суммарное квадратичное отклонение точек кластеров от центров этих кластеров (центроидов).

Описание метода:

Каждое гистологическое изображение (образец) S_i, имеющее слабую метку уровня всего изображения C_i, разбивается на непересекающиеся элементы образца равного размера (патчи) $S_{i,j}$. Элементы образца, содержащие белый фон без клеток, удаляются. С помощью автокодировщика для каждого элемента образца $S_{i,j}$ рассчитывается его сжатое векторное представление $S'_{i,j}$. Далее с помощью алгоритма K-means вычисляются значения центроидов c_k среди всех $S'_{i,j}$. Для каждого элемента образца вычисляются значения $u_{i,j,k}$, основанные на расстоянии $d_{i,j,k}$ от сжатого представления элемента образца $S'_{i,j}$ до k-го центроида:

$$u_{i,j,k} = 1, \text{если } k = argmin_k\, d_{i,j,k}, \text{в противном случае } u_{i,j,k} = 0 \quad (1)$$

Определив общие количества элементов $n_{positive}, n_{negative}$, принадлежащих образцам с раком и без рака соответственно, вычисляются значения $r_{positive,k}$ и $r_{negative,k}$ для каждого центроида:

$$r_{positive,k} = \sum_{+} u_{i,j,k} \,/\, n_{positive} \quad (2)$$

$$r_{negative,k} = \sum_{-} u_{i,j,k} \,/\, n_{negative} \quad (3)$$

где $\sum_{+}$ и $\sum_{-}$ – суммы значений $u_{i,j,k}$ всех элементов $S_{i,j}$, которые принадлежат изображениям S_i с раком и без рака соответственно.

Вероятность того, что кластер соответствует изображению с раковыми клетками:

$$I_k = r_{positive,k} \,/\, (r_{positive,k} + r_{negative,k}) \quad (4)$$

Вероятность того, что элемент образца содержит раковые клетки:

$$I_{i,j} = \sum_k I_k \times u_{i,j,k} \quad (5)$$

Если $I_{i,j} > 0.5$, то элемент образца является раковым.

После расчёта вероятностных оценок их можно уточнить с помощью оценок 4 соседних элементов образца (верхнего, нижнего, левого, правого) с помощью одной из формул:

$$I'_{i,j} = I_{i,j} + \left(\frac{\sum_{neighbors} I_{i,j}}{n_{neighbors}} - 0{,}5\right) \quad (6)$$

где $\sum_{neighbors} I_{i,j}$ – сумма вероятностных оценок соседних элементов образца, $n_{neighbors}$ – количество соседних элементов образца.

$$I'_{i,j} = tI_{i,j} + (1-t)\frac{\sum_{neighbors} I_{i,j}}{n_{neighbors}} \qquad (7)$$

где t – мера влияния оценок соседних элементов образца, $t \in [0, 1]$, настраиваемый параметр.

Используемые данные и реализация:

Набор данных состоит из гистологических снимков образцов здоровых и поражённых раком тканей предстательной железы, предоставленный Karolinska Institute и Radboud University Medical Center в рамках конкурса Prostate cANcer graDe Assessment (PANDA) Challenge.

Используются элемент образцов размером 128 на 128 пикселей.

Среднеквадратичная ошибка автокодировщика составила 0,0043 на обучающей выборке и 0,0046 на тестовой выборке.

Развитие метода:

Альтернативный способ учёта пространственного расположения элемента образца заключается в конкатенации сжатых векторных представлений $S'_{i,j}$ рассматриваемого элемента образца и его 4х соседей в общий вектор с последующей кластеризацией таких векторов для всех рассматриваемых элементов образцов и расчётом вероятностных оценок $I_{i,j}$ как описано выше.

Возможным путём развития метода может быть использование автокодировщика, принимающего на вход рассматриваемый элемент образца и его соседей, с последующей кластеризацией и расчётом вероятностных оценок как описано выше.

ЛИТЕРАТУРА

1. Kaggle.com [электронный ресурс] Prostate cANcer graDe Assessment (PANDA) Challenge https://www.kaggle.com/c/prostate-cancer-grade-assessment/data (Дата обращения 15.03.2021)
2. Yamamoto Y. et al. Automated acquisition of explainable knowledge from unannotated histopathology images //Nature Communications. – 2019. – Т. 10. – №. 1. – С. 1-9.

УДК 32.973

Друц О.В., Клещёнок Д.И. (4 курс бакалавриата)
Семенча В.Е. (2 курс магистратуры)
Муравьев Е.А., к.т.н., доцент

АВТОМАТИЗАЦИЯ СИНТЕЗА СЕМАНТИЧЕСКИХ МОДЕЛЕЙ В ТОПОЛОГИЧЕСКОЙ СЕМАНТИКЕ

В докладе “Сравнение спецификаций и программ в топологической модели семантики”, представленном в данном сборнике, обсуждалось применение семантической модели для автоматического сравнения двух фрагментов кода на языке программирования.

В данном докладе обсуждается автоматизация построения таких семантических моделей в рамках топологической модели семантики[1]. Построение топологической модели представляет собой процесс дихотомии логического определения понятия. Результатом дихотомии является дихотомическое диалектическое дерево (ДДД). Вершины дерева помечены понятиями. Вершина родитель представляет род, дочерние вершины – виды рода. Дочерние вершины разбиты на пары противоположностей. Ребра помечены предикатами – понятиями. Родитель суть субъект определения, метка ребра – определяющий предикат, дочерняя вершина – определяемое понятие.

Алгоритмы дихотомии зависят от характера исходных данных, главным образом от логической системы организации логического определения. Нами разработаны алгоритмы дихотомии для таких логических систем определений, как: словарь определения терминов; формальная грамматика языка; модель бизнес-процессов предметной области в смысле UML. Алгоритмы дихотомии включают такие топологические процедуры, как сравнение термов, сортировка векторов понятий и термов, кластерный анализ векторов понятий и термов.

Далее в докладе обсуждается пример дихотомии UML модели бизнес-процессов предметной области. Пример взят из[2]. На рис.1. приведен фрагмент бизнес-модели в виде диаграммы классов. Семантика определяемых понятий: *клиент*, *метрдотель*, *официант*, специфицирована их функциями, представленными методами. Например, семантика клиента определена списком функций: заходить в зал, быть одетым в верхнюю одежду, записываться в очередь.

Рисунок 1 – диаграмма классов фрагмента модели бизнес-процессов.

Диаграмма из UML представления акомпилируется в код на C#, приведенный ниже, показан фрагмент кода.

```
public class клиент : актор_ресторана
{

    public клиент()
    {
    }

    public void заходить_в__зал()
    {
        // TODO implement here
    }

    public void одет_в__верхняя_одежда()
    {
        // TODO implement here
    }

    public void записываться_в__очередь()
    {
        // TODO implement here
    }

}
```

C# код компилируется в систему векторов понятий. Ниже приведен пример векторов, полученных компиляцией методов классов диаграммы. Реализация компилятора с подмножества C# выполнена с применением утилиты Yacc[3].

(*зал заходить_в*)
(*верхняя_одежда одет_в*)
(*очередь записываться_в*)
(*верхняя_одежда помогать_снять*)
(*столик готовить*)

Далее применяется процедура иерархического кластерного анализа множества векторов, результатом которой является ДДД, показанное ниже. Предикаты – метки ребер для наглядности опущены.

актор_ресторана
клиент_официант
клиент
официант
метрдотель

Алгоритмы дихотомии реализованы на языке C#, проходят апробацию в учебном процессе и в промышленности.

ЛИТЕРАТУРА

1. Муравьев Е. А. Топологическая модель знаний для работы с предметной семантикой данных и алгоритмов управления в интеллектуальных системах // Труды международных конференций по морской робототехнике для освоения океана. – Санкт-Петербург: СПбГМТУ, 2019, с. 325 - 337.
2. Джозев Шмуллер. Освой самостоятельно UML за 24 часа, 2-е издание. : Пер. с англ. – М. : Издательский дом “Вильямс”, 2002. – 352 с.
3. Ахо Альфред В. Компиляторы: принципы, технологии и инструментарий, 2-е изд.: Пер. с англ. – М.: ООО “И.Д. Вильямс”, 2017. – 1184 с. : ил. – Парал. тит. англ.

УДК 004.932

Дудина А. Н. (4 курс бакалавриата)
Черноруцкий И. Г., д.т.н., профессор

СЖАТИЕ ИЗОБРАЖЕНИЙ С ПОМОЩЬЮ ГЕНЕРАТИВНО-СОСТЯЗАТЕЛЬНОЙ СЕТИ

В последнее время активно исследуются системы сжатия изображений на основе глубоких нейронных сетей. Но несмотря на то, что достижения в машинном обучении помогли улучшить производительность в ряде областей, нейронные сети все еще не превзошли существующие кодеки в сжатии изображений с потерями.

Генеративно-состязательные сети (GANs) появились как популярный метод обучения генеративных моделей для трудноразрешимых распределений неконтролируемым образом[2]. Было показано, что они способны генерировать более реалистичные и более четкие изображения, чем другие подходы, и масштабироваться до разрешения 1024 на 1024 пикселей для некоторых наборов данных[3].

Для реализации была выбрана следующая архитектура:

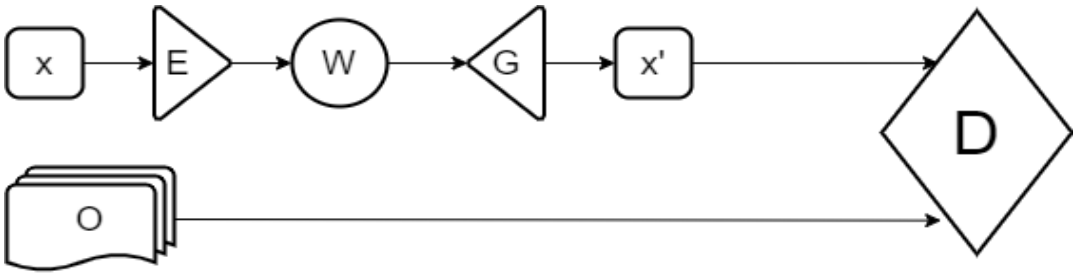


Рисунок 1 – Архитектура

где x ─ исходное изображение, E – энкодер, w – сжатое представление, G – генератор, x’ – реконструированное изображение, o – оригинальное изображение и D – дискриминатор.

Для тренировки, тестирования и валидации данной нейронной сети был использован Flickr Image dataset, содержащий около 30 000 изображений[1].

В ходе работы была получена генеративная модель, которая демонстрирует хорошие результаты пользовательской оценки визуального качества, а также позволяет обеспечить экономию битрейта при низких скоростях передачи данных.

Полученные результаты представлены на следующем изображении:

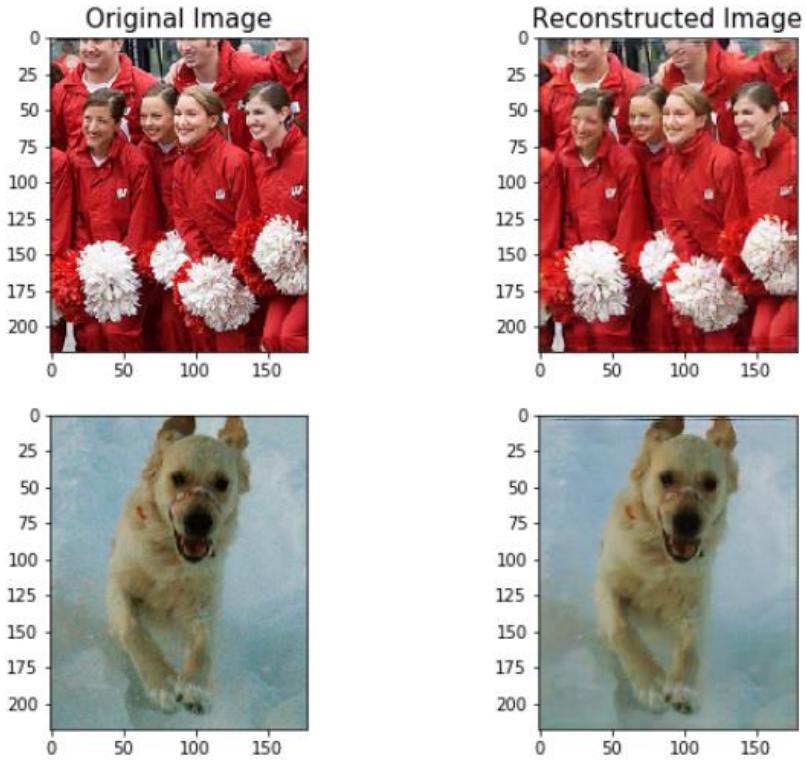


Рисунок 2 – Результат сжатия

Изображение, представленное справа, является результатом реконструкции с 43% сжатием.

ЛИТЕРАТУРА

1. Flickr Image dataset. URL: https://www.kaggle.com/hsankesara/flickr-image-dataset
2. Якуб Лангр, Владимир Бок. Генеративно-состязательные сети. Нейронные сети в глубоком обучении/ Издательство Диалектика, 2021
3. Ахирвар Кайлаш.Состязательные сети. Проекты/ Издательство, ДМК Пресс, 2020

УДК 004.021

Жукова К.А. (4 курс бакалавриата)
Черноруцкий И.Г., д.т.н., профессор

ИССЛЕДОВАНИЕ РАБОТОСПОСОБНОСТИ АДАПТИВНОГО ГРАДИЕНТНОГО МЕТОДА AMSGRAD В УСЛОВИЯХ ЖЕСТКОСТИ ЦЕЛЕВЫХ ФУНКЦИОНАЛОВ

За основу работы взята статья: «On The Convergence Of Adam And Beyond» авторов Sashank J. Reddi, Satyen Kale и Sanjiv Kumar, опубликованная как доклад на конференции ICLR (International Conference on Learning Representations) 2018. Целью работы является исследование работоспособности предложенного авторами метода AMSGrad при поиске оптимума жестких целевых функционалов.

AMSGrad принадлежит к группе адаптивных градиентных методов, основанных на применении экспоненциальных скользящих средних для масштабирования обновлений градиента. Такие методы используются, в частности, при обучении глубоких нейронных сетей для нахождения оптимальных весовых коэффициентов обучаемой модели. При решении практических задач часто приходится сталкиваться с проблемой плохой обусловленности, оптимизируемые функционалы на практике часто оказываются жесткими. Поэтому важной характеристикой метода является успешность его применения для поиска точки оптимума в условиях жесткости целевых функционалов. В работе проводилась оценка именно этой характеристики метода AMSGrad.

Метод был реализован на языке Python и протестирован на ряде тестовых функций – функции Розенброка («банановой долине»):

$$F_1 = 100({x_1}^2 - x_2)^2 + (1 - x_1)^2,$$

в которой коэффициент 100 заменялся также на 10^3-10^5, и функции Пауэлла:

$$F_2 = (x_1 + 10{x_2}^2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4.$$

Тестирование показало, что на классической «банановой долине» (с коэффициентом 100) и функции Пауэлла AMSGrad достаточно успешно попадает в окрестность точки минимума, но при увеличении коэффициента 100 в «банановой долине» до 10^3 и больше возникают ситуации джемминга (заклинивания) тем дальше от точки минимума, чем больше коэффициент, то есть чем более «сплющенной» оказывается «банановая долина», и в этих случаях алгоритм оказывается неэффективным.

Для каждого из тестовых случаев была проведена оценка полученной точности. Поскольку в зависимости от смысла решаемой задачи существуют разные подходы, оценка производилась с двух точек зрения: сходимости по аргументу (необходимой, чтобы судить об успешности решения задач идентификации) и сходимости по функционалу (достаточной в задачах аппроксимации, когда важно получить определенное значение функционала, в то время как аргумент может лежать в некоторой допустимой области).

Оценивая сходимость по аргументу, будем считать, что задача оптимизации решена с желаемой точностью, если точка, найденная алгоритмом, попадает в некоторую окрестность точки оптимума. Радиус этой окрестности зададим равным 5% нормы оптимума. Тогда, обозначив за $\|d\|$ расстояние между полученной точкой ($\tilde{x}$) и точкой оптимума (x^*), сформулируем необходимое условие:

$$\|d\| \leq 0.05\,\|opt\| \quad \text{или} \quad \frac{\|d\|}{\|opt\|} * 100\% \leq 5\%,$$

где $\|opt\|$ – норма вектора из начала координат в точку оптимума x^*.

Поскольку норма оптимума функции Пауэлла равна 0 и с ее помощью невозможно задать допустимую окрестность по указанной формуле, в этом случае радиус окрестности был задан абсолютным значением 10^{-3}.

Выполнение этих условий было проверено для каждой полученной алгоритмом точки, и было установлено, что они выполняются только для функции Розенброка с коэффициентом 100, то есть для классической «банановой долины». Однако, рассматривая случай с функцией Пауэлла, стоит отметить, что выбор абсолютного значения в качестве радиуса допустимой окрестности может быть разным, и он во многом зависит от физического смысла решаемой задачи. Поэтому для каких-то задач алгоритм окажется приемлемым по точности, а для каких-то нет.

Оценивая сходимость по функционалу, будем считать, что задача оптимизации решена с желаемой точностью, если значение целевого функционала в найденной алгоритмом точке совпадает с его значением в точке оптимума до 3 знаков после запятой или больше. Это условие выполняется как для функции Розенброка с коэффициентом 100, так и для функции Пауэлла.

Таким образом, было установлено, что в условиях сильной жесткости целевого функционала реализованный метод оптимизации AMSGrad не проявляет большой устойчивости. Более строгая сходимость по аргументу выполняется из всех рассмотренных

случаев только для классической «банановой долины», а сходимость по функционалу, кроме нее, также для функции Пауэлла. В условиях большей жесткости (когда коэффициент в «банановой долине» заменялся на 10^3-10^5) возникали ситуации заклинивания тем дальше от точки минимума, чем более «сплющенной» оказывалась «банановая долина».

Иными словами, при некотором уровне жесткости AMSGrad способен решать оптимизационные задачи, однако он оказался неработоспособен в условиях высокого уровня жесткости целевых функционалов.

ЛИТЕРАТУРА

1. Sashank J. Reddi, Satyen Kale, Sanjiv Kumar. On The Convergence Of Adam And Beyond // International Conference on Learning Representations, 2018.
2. Черноруцкий И.Г. Методы оптимизации: – учеб. пособие / И.Г. Черноруцкий – Спб.: Изд-во Политех. ун-та. 2012. – 152с.
3. Peter Auer and Claudio Gentile. Adaptive and self-confident on-line learning algorithms // In Proceedings of the 13th Annual Conference on Learning Theory, 2000. – P. 107–117.
4. Diederik P. Kingma and Jimmy Ba. Adam. A method for stochastic optimization // In Proceedings of 3rd International Conference on Learning Representations, 2015.
5. T. Tieleman and G. Hinton. RmsProp: Divide the gradient by a running average of its recent magnitude // COURSERA: Neural Networks for Machine Learning, 2012.

УДК 004.932

Забродина Е.С. (4 курс бакалавриата)
Черноруцкий И.Г., д.т.н., профессор

ПРЕОБРАЗОВАНИЕ ИЗОБРАЖЕНИЙ В СВЕРХРАЗРЕШЕНИЕ ПРИ ПОМОЩИ НЕЙРОННОЙ СЕТИ

Преобразование изображения в сверхразрешение (SISR), как фундаментальная проблема компьютерного зрения, привлекает все большее внимание научного сообщества и компаний, занимающихся ИИ. SISR направлено на восстановление изображения высокого разрешения (HR) из одного изображения низкого разрешения (LR). Первой важной работой на эту тему была SRCNN, основанная на сверточных нейронных сетях (CNN) которая сделала толчок в этом направлении. Различные конструкции сетевой архитектуры и стратегии обучения постоянно улучшали характеристики сверхразрешения (SR). Однако эти подходы имели тенденцию выдавать чрезмерно сглаженные результаты без достаточных высокочастотных деталей.

В данной работе для обучения использовался набор данных DIV2K, который представляет собой набор данных высокого качества (разрешение 2K) для задач восстановления изображений. Помимо обучающего набора DIV2K, который содержит 800 изображений, также были использованы другие наборы данных с богатыми и разнообразными текстурами для нашего обучения. С этой целью использовался набор данных Flickr2K, состоящий из 2650 изображений 2K с высоким разрешением, собранных на веб-сайте Flickr, и набор данных OutdoorSceneTraining (OST), чтобы сделать обучающие данные более разнообразными. Эмпирически было обнаружено, что использование этого большого набора данных с более богатыми текстурами помогает генератору получать более естественные результаты.

Работа заключалась в постепенной модификации базовой модели генеративно-состязательной нейронной сети для сверхразрешения (SRGAN).

Сначала были удалены слои batch-normalization (BN) для стабильной работы. Это не снижает производительность, но экономит вычислительные ресурсы и использование памяти.

Кроме того, мы заметили, что, когда сеть более глубокая и сложная, модель со слоями BN с большей вероятностью получит нежелательный результат.

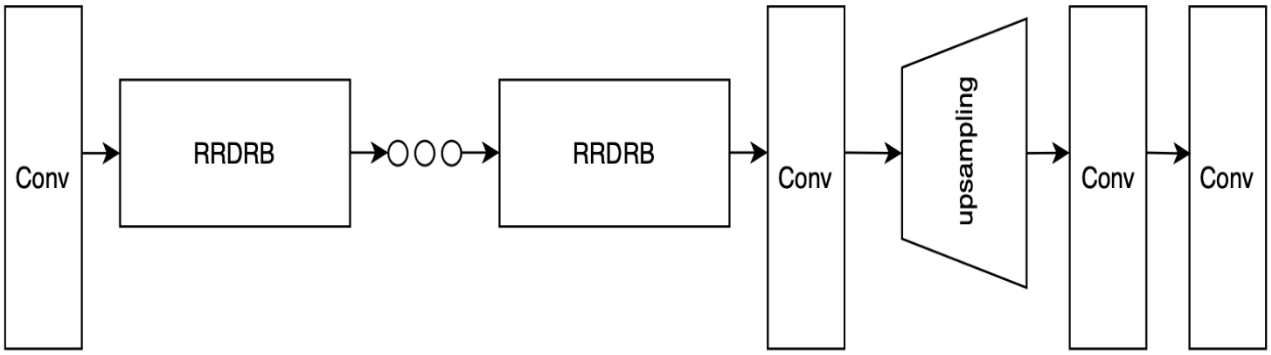


Рисунок 1 – Архитектура нейронной сети

Затем исходный базовый блок SRGAN были заменены блоком, который сочетает в себе рекуррентную нейронную сеть и полносвязных слоев. В итоге получаем архитектуру, представленную на рисунке 1.

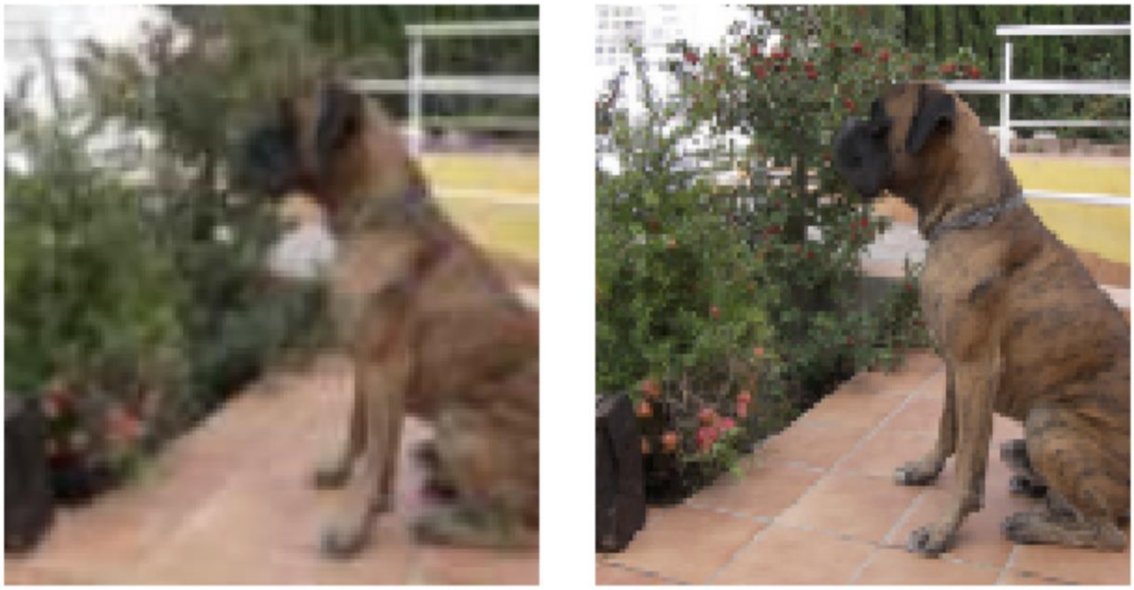


Рисунок 2 – Полученные результаты.

Поскольку не существует эффективных и стандартных показателей качества восприятия, на рисунке 2 представлен репрезентативный результат, по которому можно судить о качестве.

В ходе работы была получена модель нейронной сети, обеспечивающая высокое качество восприятия.

ЛИТЕРАТУРА

1. J. Redmon, A. Farhadi You Only Look Once: Unified, Real-Time Object Detection. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. DOI: 10.1109/CVPR.2016.91
2. A. Wheeler Standard and Super-Resolution Bioimaging Data Analysis. A Primer/ RMS. – 2018. – с. 27-89.
3. Sahar Ismail,Hala Mansour and Mahmoud Allam Super Resolution/ Lambert. – 2014. – с. 120.

УДК 004.032.26

Загородний Г.Р., Стахиев А.В. (4 курс бакалавриата)
Абрамов Н.А., ассистент

УДАЛЕНИЕ АРТЕФАКТОВ ВИДЕОПЛЕНКИ МЕТОДОМ ДВОЙНОЙ ИНТЕРПОЛЯЦИИ КАДРОВ

Со временем на кинопленке появляются дефекты, такие как грязь, потертости и царапины. Наш метод позволяет удалять данные артефакты, основываясь на информации с соседних кадров.

Ядром данного метода является нейросеть для интерполяции кадров, позволяющая создавать из двух соседних кадров промежуточный. Этот метод делает предположение о том, как объекты движутся в кадре (optical flow), и использует эту информацию, чтобы сгенерировать промежуточный кадр.

Поскольку дефекты пленки появляются только на одном кадре, алгоритм не может предположить, куда они сдвинулись, и убирает их в сгенерированном кадре. Так как после первой итерации мы получаем смещенные кадры, мы делаем вторую итерацию и получаем очищенные начальные кадры.

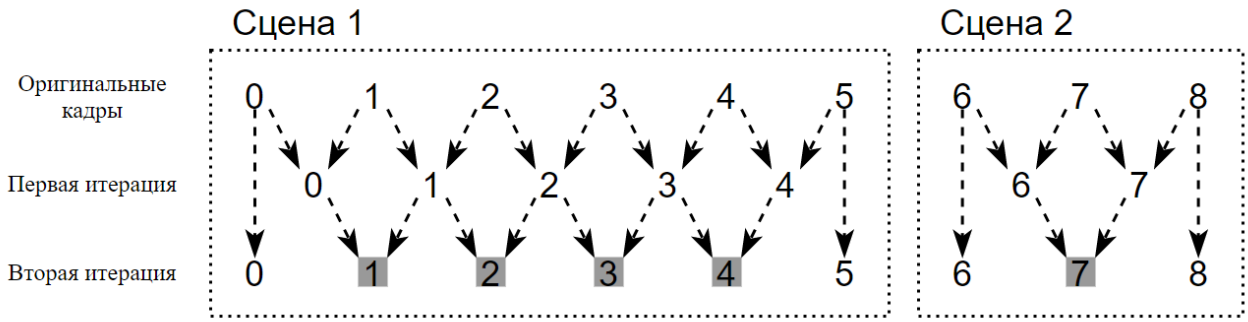


Рисунок 1 – Диаграмма работы метода. Выделенные серым кадры – без дефектов, не выделенные – с дефектами.

Проблемными являются кадры, стоящие на стыке сцен, а также первый и последний. Наш алгоритм невозможно использовать для них, так что мы просто копируем их исходный вариант. Границы сцен находятся автоматически при помощи алгоритмического анализа изменения цветовых гистограмм в соседних кадрах.

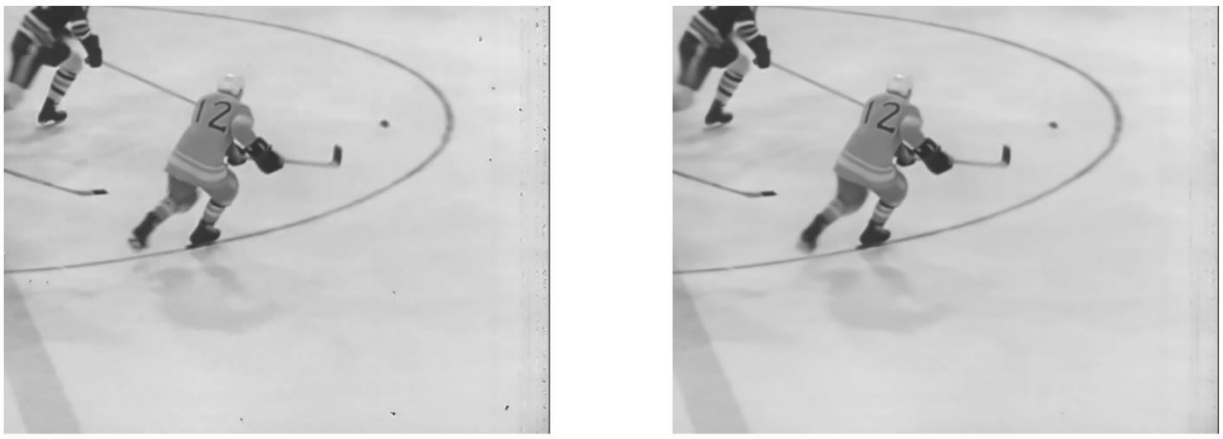


Рисунок 2 – Результат

Данный метод показывает хорошие результаты в большинстве случаев. Но он также обладает теми же ограничениями, что и нейросети для интерполяции кадров. В частности, при быстром движении может потеряться четкость объекта (например, левый конек хоккеиста на

рисунке 2). Также, время для одной итерации может доходить до 1 секунды на FullHD кадр при использовании state-of-art нейросети RIFE [1].

Над проблемой интерполяции кадров работает множество ученых, и каждый год появляются новые решения [2]. Таким образом можно с уверенностью сказать, что со временем наш метод будет становиться точнее и быстрее.

ЛИТЕРАТУРА

1. RIFE: Real-Time Intermediate Flow Estimation for Video Frame Interpolation (2021) https://arxiv.org/abs/2011.06294
2. https://github.com/lyh-18/Video-Frame-Interpolation-Collections

УДК 004.75

Каргалов Л.А. (4 курс бакалавриата)
Черноруцкий И.Г., д.т.н., профессор

ОПТИМИЗАЦИЯ МАРШРУТИЗАЦИИ В СЕРВЕРНО-ОРИЕНТРИРОВАННЫХ СЕТЯХ

Современные протоколы маршрутизации такие как RIP и OSPF достаточно часто используют для маршрутизации трафика в серверно-ориентированных сетях, в их основе лежат алгоритмы Беллмана-Форда и Дейкстры соответственно. Их основной недостаток заключается в оценке стоимости перехода между узлами (далее хоп, от англ. hop - переход) лишь одним критерием и отсутствием загруженности канала связи. Отсюда вытекает цель данной работы, разработать и реализовать алгоритм способный учитывать стоимость маршрут как векторную величину (совокупность различных характеристик канала связи) и загруженность канала, иными словами, проводить балансировку трафика.

В связи с различными свойствами характеристик ограничений они могу быть разделены на три основные группы: аддитивные, мультипликативные и вогнутые.

Пускай путь P содержит j хопов, тогда $\omega_i(e)$ означает вес $i^{\text{ой}}$ компоненты вектора весов ребра e. Исходя из природы ограничений можем построить следующие определения:

1. Аддитивные ограничения – задержка и джиттер. Такое ограничение рассчитывается по формуле:

$$\omega_i(P) = \sum_{l=1}^{j} \omega_i(e_l)$$

где i – порядковый номер ограничения, e – ребро графа (среда распространения сигнала).

2. Мультипликативные ограничения – надежность соединения, вероятность потери пакетов. Такое ограничение рассчитывается как произведение вектора весов переходов

$$\omega_i(P) = \prod_{l=1}^{j} \omega_i(e_l) = e^{(\sum_{l=1}^{n}[\omega_i(e_l)])}$$

3. Вогнуто ограничение – пропускная способность, для ясности можно сравнить с законом Амдала – «Скорость исполнения параллельной программы равна скорости самого медленного ее потока». Такое ограничение рассчитывается как функция максимума от вектора весов переходов

$$\omega_i(P) = \min\{\omega_i(e_1), \omega_i(e_2), \omega_i(e_3), \dots, \omega_i(e_j)\}$$

Далее необходимо ввести механизм, позволяющий производить балансировку трафика. Для этого добавим еще одно вспомогательное ограничение (штрафной коэффициент) L, определяющее загруженность маршрута. После обнаружения очередного маршрута данный коэффициент инкрементируется на каждом узле следования. Это позволит увеличить стоимость маршрута при высоком спросе и принудит использовать альтернативные маршруты. Ограничение будет иметь аддитивную природу.

Так же в целях повысить устойчивость системы используем функцию запаса работоспособности

$$Z_i\big(\omega_i(P)\big) = \frac{-c_i + \omega_i(P)}{|c_i| * (-\delta_i)} + 1 \leq 0$$

где c_i – максимально допустимое значение характеристики, а δ_i – значение запаса характеристики.

Перейдем к расчету стоимости пути. Составляется вектор весов $W(P) = \{Z_1(\omega_1(P)), Z_2(\omega_2(P)), \dots, Z_2(\omega_j(P)), Z(L(P))\}$ и проверяется соответствие неравенству $Z_i \leq 0$, для каждого компонента. Имея на столько обширный список ограничений, функция расчета пути попросту не может быть линейной.

$$COST(P) = \left[\sum_{i=1}^{k} Z_i(\omega_i(P))^q + Z(L(P))^q\right]^{\frac{1}{q}}$$

Здесь q – это норма Гельдера, это варьируемый параметр, позволяющий при его увеличении находить больше возможных путей. Так для q стремящемся к бесконечности имеем:

$$COST_\infty(P) = \max_{1 \leq i \leq k}\{\, Z_i\big(\omega_i(P)\big), Z\big(L(P)\big)\,\}$$

Перейдем к реализации алгоритма.

1. На входе имеем матрицу смежности, которая отражает связи между соседними попарно соединенными узлами. В каждой ячейке матрицы содержится дополнительная информация об узле в виде вектора $\{\omega_1, \omega_2, \dots, \omega_j, L\}$ его характеристик и требования к линии связи для исходящих маршрутов $\{c_1, c_2, \dots, c_j, c_{j+1}, \delta_1, \delta_2, \dots, \delta_j, \delta_{j+1}\}$, c_{j+1} и δ_{j+1} необходимы для ограничения L.

2. Далее обрабатываем исходную матрицу смежности и строим новую соответствующую соединению узлов через один побочный, таким образом получаем матрицу содержащую информацию о путях с длинной 2. Производим расчёт стоимости маршрутов по формуле $COST(P)$, при этом если вектор весов $W(P)$ не соответствует ограничениям, то не имеет смысл строить новые маршруты длиннее текущего из этого узла.

3. Построение новых матриц буде продолжаться до тех пор, пока не появятся циклы в маршрутах или не будет достигнут необходимый предел количества хопов.

4. Итогом работы алгоритма будет сортировка всех маршрутов по их стоимости.

Произведем ряд тестов, частично подтверждающих работоспособность алгоритма. Для этого построим сеть с топологией DCell состоящей из 6 узлов. Первый тест проводился в условиях равновесия системы, его результаты используем как контрольные. Второй тест проводился в условиях дисбаланса системы, характеристики узлов устанавливались случайным образом. Третий тест демонстрирует ситуацию, при которой один из узлов вышел из строя.

Название теста	Средняя стоимость маршрута	Среднее количество хопов	Количество маршрутов, проходящих через узел
Равновесие	0.349	2.556	N1=10, N4=12, N0=14, N5=18, N2=18, N3=20
Дисбаланс	0.423	2.222	N0=12, N1=12, N2=14, N3=16, N5=16, N3=16
Отказ узла	0.471	1.933	N2=4, N0=10, N3=10, N1=14, N5=14, N4=16

Из результатов теста видно, что несмотря на дисбаланс системы или отказ одного из узлов средняя стоимость маршрута увеличивается в пределах 21-34%, среднее количество

хопов уменьшается. Это доказывает эффективность алгоритма для серверно-ориентированных сетей.

ЛИТЕРАТУРА

1. И.Г. Черноруцкий, Методы принятия решений. - СПБ.: БВХ-Петербург, 2005. – 416 с
2. Routing optimization for server-centric data center networks // URL: https://www.researchgate.net/publication/305417517_Routing_optimization_for_server-centric_data_center_networks

УДК 004.42

Крамаров Е. В. (4 курс бакалавриата)
Черноруцкий И. Г., д.т.н., профессор

ПРИМЕНЕНИЕ ГИБРИДНОГО АЛГОРИТМА WOAGWO ДЛЯ ЗАДАЧИ КЛАСТЕРИЗАЦИИ

Задача кластеризации – это задача разделения группы объектов по некоторому заранее неизвестному признаку на группы – кластеры. Эта задача имеет высокую актуальность в сфере машинного обучения, поскольку предварительная кластеризация упрощает последующее обучение. Кроме того, решение задачи кластеризации позволяет строить гипотезы на основе статического анализа данных без дополнительных сведений. Также кластеризация может быть полезной при сжатии данных, поскольку у схожих объектов внутри одного кластера можно в некоторых случах отбросить какие-то признаки. Или же схожие объекты внутри одного кластера можно описать их общим способом и для каждого только задавать отклонение от предложенной общей записи.

Целью данной работы является разработка алгоритма на базе метаэвристического оптимизационного алгоритма WOAGWO для решения задачи кластеризации.

В общем случае задача кластеризации довольно просто сводится к задаче оптимизации, в которой минимизируется функция потерь. В зависимости от постановки конкретной задачи эта функция может задаваться по-разному, но чаще всего она задаётся, как сумма расстояний от всех объектов, отнесённых к данному кластеру до точки, принятой за центр кластера. Исходя из такого определения задачи, очевидно, что подобная задача может быть решена с помощью существующих методов оптимизации.

На данный момент были предложены решения этой задачи с помощью метаэвристических алгоритмов оптимизации PSO (particle swarm optimizer – оптимизатор роем частиц) и WOA (whale optimization algorithm – китовый алгоритм оптимизации). При этом, алгоритм WOA показал лучшие результаты[2]. Позже был предложен гибридный алгоритм WOAGWO (whale optimization algorithm + grey wolf optimizer), улучшающий результаты WOA в целом ряде задач за счёт добавления в него элементов из GWO (grey wolf optimizer – оптимизатор серого волка)[1]. Его и предлагается использовать в данной работе для решения задачи кластеризации. Предполагается, что данный алгоритм продемонстрирует лучшие результаты.

В ходе работы была написана программа на языке python версии 3.8, реализующая алгоритм оптимизации WOAGWO и протестирован на различных наборах тестовых данных, приведённых в [2].

В работе приводятся результаты сравнения реализованного алгоритма с результатами других алгоритмов, решавших ту же задачу.

ЛИТЕРАТУРА

1. Mohammed, H., Rashid, T. A novel hybrid GWO with WOA for global numerical optimization and solving pressure vessel design. Neural Comput & Applic 32, 14701–14718 (2020). https://doi.org/10.1007/s00521-020-04823-9

2. Jhila Nasiri & Farzin Modarres Khiyabani | Akiko Yoshise (Reviewing editor) (2018) A whale optimization algorithm (WOA) approach for clustering, Cogent Mathematics & Statistics, 5:1, DOI: 10.1080/25742558.2018.1483565

УДК 004.773

Лаппо С.С. (2 курс магистратуры)
Петров А.В., ст. преподаватель

АДАПТИВНАЯ СИСТЕМА ПЛАНИРОВАНИЯ ВРЕМЕНИ

Целью работы является разработка программного обеспечения, позволяющего упростить планирование личного, а также рабочего времени. Программное обеспечение должно быть доступным на большинстве популярных операционных систем и предоставлять возможности по автоматизации планирования как рутинных задач, так и задач требующих определенных ресурсов. Кроме того, система должна позволять достигать договоренности о встрече нескольких участников с минимальным участием пользователей.

С появлением компьютера человечество смогло автоматизировать огромное количество различных задач, ускорить выполнение огромного множества процессов, однако, планирование задач во многом не поменялось. Появились todo приложения, автоматизированные доски, для упрощения планирования встреч существуют различные утилиты, однако же, уровень автоматизации в них крайне низок.

Для решения данной проблемы было решено разработать программное обеспечения, которое бы работало на максимально возможном количестве платформ, таким образом, предоставляя пользователям терминал для взаимодействия с системой в любом месте.

Для решения поставленной проблемы был разработан подход, комбинирующий в себе несколько техник оптимизаций с ограничениями, процесс достижения консенсуса и необычную модель события.

Традиционные планировщики задач и календари в среднем содержат крайне ограниченный набор данных, таких как название задачи, срок выполнения, описание. Изредка к этому добавляется местоположение или некоторые подзадачи. Кроме того, в случае если речь идет не о задаче, а о встрече, то дополнительное появляется список участников. Однако, этих данных недостаточно для того, чтобы формально оценить, когда пользователю предпочтительно выполнять ту или иную задачу. Поэтому в модель задачи были добавлены такие поля как: польза (значение от 1 до 10 по мнению пользователя, с коэффициентом ослабевания со временем), тип задачи, может ли данная задача выполняться параллельно с другой, или, может она вообще является чем-то, намеренно занимающим место в календаре, и обозначающем, что в данный промежуток времени могут выполняться только определенные задачи, в связи с чем были введены бинарные, импликационные и ограничения близости/порядка. Кроме того, была добавлена возможность ограничивать продолжительность не только всей задачи, но и отдельных частей её выполнения, а также временные интервалы между данными частями. Ещё одним важным упущением существующих моделей является игнорирования того, что для выполнения определенных задач пользователю может требоваться определенный набор вещей, который находится в том или ином месте. Кроме того, некоторые задачи могут выполняться не в фиксированном месте, а в ряде мест (например, покупка продуктов может происходить в любом супермаркете) и предпочтительно может быть запланировать её в том месте, которое находится на пути пользователя.

В первую очередь данная модель отталкивается от выставленной пользователем полезностью задачи, где 1 – может быть не выполнена, а 10 – должна быть обязательно выполнена в срок. Такая же градация применяется и к встречам, если важность встречи

указана как 1 – она может быть отменена или рекомендована к переносу. В то время как если важность указана равной 10 она будет иметь наивысший приоритет во время оптимизации.

Оптимизация состоит из 3 различных алгоритма, каждый из которых предполагает определенную степень детализации. Первый – алгоритм планирования ближайшего времени, с учетом ограниченных ресурсов. Данный алгоритм фокусируется на ближайших 24 часах и том, что пользователь должен выполнить в процессе. Так как любая активность по сути является задачей, на которую имеется ограниченные ресурсы – алгоритм оптимизации настроен таким образом, чтобы в условиях постоянно уменьшающихся ресурсов (где главный время) достичь наибольшего количества выполненных задач. Данный алгоритм специально выбран таким образом, чтобы адаптироваться в случае, если пользователь не смог выполнить очередную задачу или решил не делать этого.

Второй – алгоритм, отвечающий за планы на ближайшие 4 недели. Он опускает многие ограничения и имеет возможности к более гибкому перестроению. Если первый алгоритм, не имеет возможности удалить определенные задачи для повышения эффективности, а лишь добавить, то второй после каждого действия пользователя проводит повторную оптимизацию, в том числе активируя процесс консенсуса.

Консенсус является третьим алгоритмом, входящим в процесс оптимизации. В рамках данного процесса календари различных пользователей посылают предложения удаленному серверу с доступными временными слотами по приоритету, который вычисляет наиболее подходящий временной промежуток, после чего он отдается на согласование пользователю.

Для реализации клиента был выбран язык Dart с пользовательским интерфейсом, реализованным на фреймворке Flutter. Это обосновано тем, что данный фреймворк позволяет писать один код для множества платформ, в том числе для WEB. Так как оптимизация – ресурсоемкая задача код сервера реализован на GO (golang), а в качестве базы используется PostgreSQL. На данный момент рассматривается возможность миграции на NoSQL базу так как хранение таких данных в структурированном виде очень затратно и во многом не имеет смысла.

Оценка эффективности представленного решения на данный момент не проведена и является следующим этапом выполнения данной работы.

ЛИТЕРАТУРА

1. Jacob Bank, Zachary Cain, Yoav Shoham, Caroline Suen, and Dan Ariely. 2012. Turning personal calendars into scheduling assistants. In Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts-CHIEA'12. ACMPress. https://doi.org/10.1145/2212776.2223854
2. Abder Rezak Benaskeur, Froduald Kabanza, and Eric Beaudry. 2010. CORALS. ACM Transactions on Intelligent Systems and Technology 1, 2 (ноябрь 2010), 1–21. https: //doi.org/10.1145/1869397.1869402
3. Pauline M. Berry, Melinda Gervasio, Bart Peintner, and Neil Yorke-Smith. 2011. PTIME. ACM Transactionson Intelligent Systems and Technology 2,4 (июль 2011), 1–22. https://doi.org/10.1145/1989734.1989744
4. Justin Cranshaw, Emad Elwany, Todd Newman, Rafal Kocielnik, Bowen Yu, Sandeep Soni, Jaime Teevan, and Andre´s Monroy-Herna´ndez. 2017. Calendar.help. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. ACM. https://doi.org/10.1145/3025453.3025780
5. Patrick Hosein and Steffan Boodhoo. 2016. Event scheduling with soft constraints and on-demand re-optimization. In 2016 IEEE International Conference on Knowledge Engineering and Applications (ICKEA). IEEE. https://doi.org/10.1109/ickea.2016. 7802993
6. Donghyeon Kim, Jinhyuk Lee, Donghee Choi, Jaehoon Choi, and Jaewoo Kang. 2018. Learning User Preferences and Understanding Calendar Contexts for Event Scheduling. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management. ACM. https://doi.org/10.1145/3269206.3271712

УДК 32.973

Иванов Г.А., Королев К.И. (4 курс бакалавриата)
Семенча В.Е. (2 курс магистратуры)
Муравьев Е.А., к.т.н., доцент

СРАВНЕНИЕ СПЕЦИФИКАЦИЙ И ПРОГРАММ В ТОПОЛОГИЧЕСКОЙ МОДЕЛИ СЕМАНТИКИ

Топологическая модель семантики разработана одним из авторов данного доклада[1], при содействии соавторов доклада. Основу формальной семантики составляет топологическое пространство на словаре понятий, связанных в дерево конструкциями диалектики Г.В.Ф.Гегеля. В данном докладе на примере сравнения двух фрагментов кода программы демонстрируется процедура автоматического сравнения структурированных спецификаций и программ. Основной результат топологической семантики состоит в положении: любые два иерархических терма, включающие только понятия из топологического пространства, формально и однозначно сравнимы.

Пример взят из [2]. Рассмотрим два сценария, представленных фрагментами кода на языке высокого уровня. Коды включают переменные, семантика которых определена топологической моделью. Клиент ресторана заходит в ресторан. Первый сценарий: *Если клиент одет в верхнюю одежду, то метрдотелоь помогает ее снять*. Второй сценарий: *Если столик зарезервирован, то клиент проходит в зал, иначе записывается в очередь.* Ниже представлены спецификации сценариев на языке программирования высокого уровня.

if(одет_в(клиент, верхняя_одежда))помогает_снять(метрдотель, верхняя_одежда);
if(зарезервирован(столик))проходить_в(клиент, зал));
else записываться_в(клиент, очередь);

Результаты сравнения данных фрагментов кода представлены в таблице, приведенной ниже.

Таблица результатов сравнения фрагментов кода.

	Общее	Различие_0 от общего	Различие_1 от общего
В области конструкций языка программирования:	*Конструкция if.*	*if без else.*	*if с else.*
В предметной области:	Условие if - *ресторан.*	Субъект условия if – *клиент.* Предикат условия if – *одет/не одет.* Объект условия if – *верхняя одежда.*	Субъект условия if – *столик.* Предикат условия if – *зарезервирован/нет.*
	Оператор if - *действие выполняется членом сценаря по теме ресторан.*	Субъект оператора if – *метрдотель.* Предикат оператора if – *помогать_снять.* Объект оператора if – *верхняя_одежда.*	Субъект оператора if – *клиент.* Предикат оператора if – *проходить_в.* Объект оператора if – *зал.*
			Субъект оператора else – *клиент.* Предикат оператора else – *записываться_в.* Объект оператрра else – *очередь.*

Процедура сравнения выявляет сходства и различия сравниваемых термов. При этом важно отметить, что ведется сравнение как конструкций языка программирования, так и

сравнение семантики предметных понятий – переменных. В части сравнения конструкций языка программирования, как следует из первой строки таблицы, общим для сравниваемых фрагментов кода является то, что это конструкция if. Различием первого фрагмента от общего является то, что это if без else, второго, то что это if с else.

Следующие строки таблицы представляют общее и различие деталей фрагментов кода, и включают сравнение семантики входящих в коды переменных.

Обратим внимание на вторую строку таблицы сравнения. Общим в части условия if является род субъектов термов, представляющих условие в первом и втором случаях. Таковым родом является понятияе *ресторан,* видами – *клиент* в первом сценарии, и *столик* во втором. Данный фрагмент сравнения демонстрирует способность системы вести сравнение понятий по родо-видовм отношениям, а не просто по совпадению входящих в термы имен.

Программное обеспечение реализовано на языке программирования C#, проходит апробацию при решении задач в учебном процессе и в промышленности.

ЛИТЕРАТУРА

1. Муравьев Е. А. Топологическая модель знаний для работы с предметной семантикой данных и алгоритмов управления в интеллектуальных системах // Труды международных конференций по морской робототехнике для освоения океана. – Санкт-Петербург: СПбГМТУ, 2019, с. 325 - 337.
2. Джозев Шмуллер. Освой самостоятельно UML за 24 часа, 2-е издание. : Пер. с англ. – М. : Издательский дом “Вильямс”, 2002. – 352 с.

УДК 004.021

Масленникова А. А. (4 курс бакалавриата)
Черноруцкий И. Г., д.т.н., профессор

ИССЛЕДОВАНИЕ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМА ADAM В МОДЕЛЯХ ГЛУБОКОГО ОБУЧЕНИЯ

Adam - это адаптивный алгоритм оптимизации скорости обучения, разработанный специально для обучения глубоких нейронных сетей. Этот алгоритм можно рассматривать как комбинацию RMSprop и стохастического градиентного спуска с импульсом. Он использует квадраты градиентов для масштабирования скорости обучения, например, RMSprop, и использует преимущество импульса, используя скользящее среднее градиента вместо самого градиента, как SGD с импульсом.

Целью этой работы является реализация алгоритма Adam и некоторых других алгоритмов оптимизации нейронных сетей, чтобы сравнить их эффективность. Так как Adam является одним из самых популярных способов оптимизации нейронных сетей, данная работа является очень актуальной.

Алгоритмы, выбранные для последующей реализации:

– Adam. Основная идея алгоритма заключается в накопления движения и идеи более слабого обновления весов для типичных признаков. Это метод адаптивной скорости обучения, что означает, что он вычисляет индивидуальную скорость обучения для разных параметров. Алгоритм использует оценки первого и второго моментов градиента, чтобы адаптировать скорость обучения для каждого веса нейронной сети. Для оценки моментов используется экспоненциально скользящие средние, вычисленные по градиенту.

$$m_n = E[X^n];\ m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t;\ v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2;\ m_t = (1-\beta_1)\sum_{i=0}^{t} \beta_1^{t-i} g_i\,;\ w_t = w_{t-1} - \eta \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \varepsilon}$$

– Nadam. Импульс Нестерова, включенный в Adam. Идея алгоритма состоит в том, чтобы использовать термин импульса Нестерова для первых скользящих средних. Правило обновления эквивалентно шагу в направлении вектора импульса, а затем шагу в направлении градиента. Однако шаг импульса не зависит от текущего градиента, поэтому мы можем получить направление шага градиента более высокого качества, обновив параметры с шагом импульса перед вычислением градиента.

$$m_t = \beta m_{t-1} + \eta g_t;\ g_t = \nabla f(w_{t-1} - \beta m_{t-1});\ w_t = w_{t-1} - m_t$$

– AdaGrad. Алгоритм выполняет большие обновления для нечастых параметров и меньшие для частых, в основном устраняет необходимость настраивать скорость обучения.

$$n_t = n_{t-1} + g_t^2;\ w_t = w_{t-1} - \eta \frac{g_t}{\sqrt{n_t} + \varepsilon}$$

– RMSProp. RMSprop является модификацией алгоритма AdaGrad, но сумма квадратов значений производной по весу в течение всех предыдущих шагов рассчитывается иным способом. По каждому весу есть квадраты значений частных производных. В методе RMSprop сумма квадратов значений равно их скользящему среднему с экспоненциальным затуханием. В AdaDelta вместо суммирования всех прошлых квадратных корней используется скользящее окно, которое позволяет уменьшить сумму.

$$n_t = v n_{t-1} + (1 - v) g_t^2;\ w_t = w_{t-1} - \eta \frac{g_t}{\sqrt{n_t} + \varepsilon}$$

Реализовав вышеперечисленные алгоритмы, было проведено два эксперимента:

– Алгоритмы были запущенны из одной точки. Все они довольно хорошо сходятся (с минимальным подбором скорости обучения).

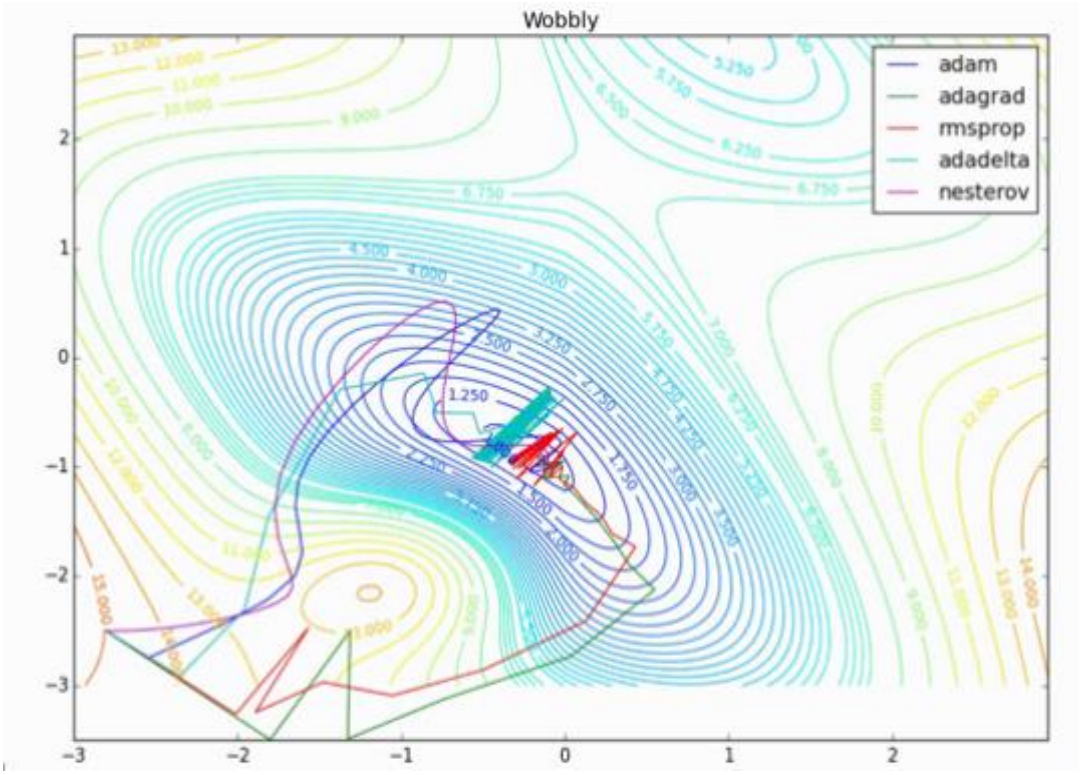


Рисунок 1 – Сравнение производительности алгоритмов. Эксперимент №1.

– Во втором эксперименте все алгоритмы, кроме Adam, оказались заперты в локальном минимуме.

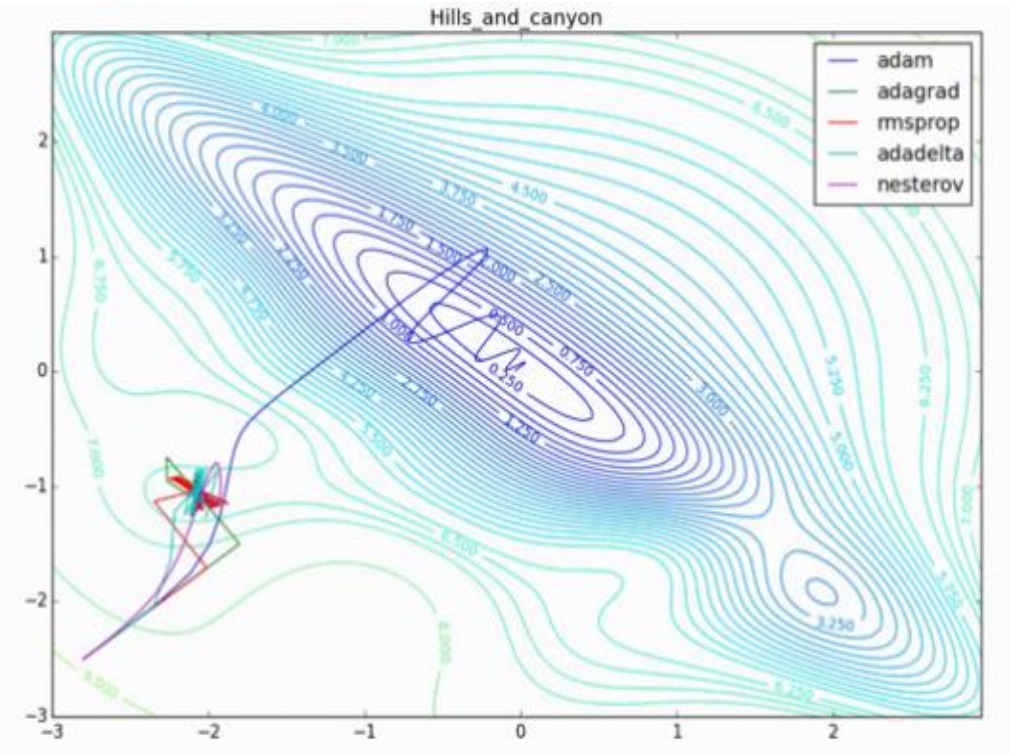


Рисунок 2 – Сравнение производительности алгоритмов. Эксперимент №2.

Алгоритм Adam использует адаптивную скорость обучения и является наиболее эффективным методом стохастической оптимизации, для которой требуются только градиенты первого порядка с небольшими требованиями к памяти. Он сочетает в себе преимущества оптимизатора AdaGrad для работы с разреженными градиентами и способность оптимизатора RMSProp справляться с нестационарными задачами.

ЛИТЕРАТУРА

1. Vitaly Bushaev. Adam — latest trends in deep learning optimization. – [Электронный ресурс] URL: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c
2. Diederik P. Kingma and Jimmy Lei Ba. Adam : A method for stochastic optimization. 2014. arXiv:1412.6980v9 – [Электронный ресурс] URL: https://arxiv.org/abs/1412.6980
3. Jason Brownlee. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning – [Электронный ресурс] URL: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

УДК 519.6

Мордвинцева А.И. (4 курс бакалавриата)
Черноруцкий И.Г., д.т.н., профессор

ПРИМЕНЕНИЕ АЛГОРТМА СОЦИАЛЬНОГО ПАУКА ПРИ РЕШЕНИИ ЗАДАЧ ОПТИМИЗАЦИИ

Целью работы является исследование свойств метаэвристического алгоритма Social Spider Optimization (SSO, Оптимизация социального паука) при решении задач оптимизации. Алгоритм SSO относится к классу роевых алгоритмов и вдохновлён коллективным поведением пауков в колонии. В SSO поисковыми агентами являются пауки, которые коллективно перемещаются в соответствии с биологическим поведением колонии.

В большинстве алгоритмов роевого интеллекта особи моделируются на основе одних и тех же свойств и поведения. Напротив, SSO определяет два разных вида поисковых агентов: мужского и женского пола с разным набором эволюционных операторов.

Задача глобальной оптимизации, для решения которой применяется алгоритм, представляется в следующем виде:

$$\begin{array}{ll} \textit{минимизировать} & f(x),\ x=(x^1,x,^2..,x^d) \\ \textit{при условии} & x\in X,\ X=\{x\in R^d \mid l_h\le x\le u_h,\ h=1,..,d\} \end{array} \quad (1)$$

где $f:R^d\to R$ – нелинейная функция, X – допустимое множество решений, ограниченное нижними l_h и верхними u_h границами.

Популяция пауков размером N делится на две группы поисковых агентов: мужского и женского пола. Для моделирования реальной колонии пауков количество самок выбирается случайным образом из расчёта 65-95% от общего размера популяции, тогда как остальные особи рассматриваются как особи мужского пола.

Каждый паук имеет вес, который вычисляется в соответствии с его параметром приспособленности (fitness-параметром), который показывает, насколько оптимально решение, которое представляет паук. Вес рассчитывается следующим образом:

$$we_i=\frac{fit_i-worst}{best-worst},\quad i\in\{1,...,N\} \quad (2)$$

где fit_i – это показатель приспособленности паука; $best$ и $worst$ - лучшее и худшее значение показателя приспособленности по всей популяции соответственно.

Обмен информацией в SSO между пауками моделируется вибрациями, возникающими в паутине. Вибрация, которую i-й паук воспринимает от j-го другого паука, вычисляется следующим образом:

$$V_{i,j}=we_j e^{d_{i,j}^2} \quad (3)$$

где d – это расстояние между пауками. Каждый i-й паук может воспринимать три типа вибраций: $v_{i,n}$, $v_{i,b}$ и $v_{i,f}$, где n – ближайший паук с большим весом ($we_n>we_i$), b – паук с лучшим значением показателя приспособленности, f – ближайшая самка (используется только в случае, если i-й паук – самец).

На каждой итерации алгоритма воспроизводятся 3 фазы: фаза перемещения особей женского пола, фаза перемещения особей мужского пола и фаза спаривания.

Фаза перемещения особей женского пола. Новое положение особи женского пола получается путем изменения текущего положения паука, которое случайным образом зависит от вероятности Pf, которая является параметром алгоритма, а движение производится в зависимости от расположения других пауков и вибраций, принимаемых самкой:

$$fs_i^{k+1}=\begin{cases} fs_i^k+\alpha\cdot V_{i,n}\cdot(s_n-fs_i^k)+\beta\cdot V_{i,b}\cdot(s_b-fs_i^k)+\delta\cdot(rand-\frac{1}{2}) & \textit{с вероятностью } Pf \\ fs_i^k-\alpha\cdot V_{i,n}\cdot(s_n-fs_i^k)-\beta\cdot V_{i,b}\cdot(s_b-fs_i^k)+\delta\cdot(rand-\frac{1}{2}) & \textit{с вероятностью } 1-Pf \end{cases} \quad (4)$$

Здесь числа α,β,δ – случайные числа от 0 до 1, k – номер итерации. С вероятностью Pf самка движется в стороны лучшего и ближайшего паука с большим весом (привлечение), иначе она движется в обратную сторону (отталкивание).

Фаза перемещения особей мужского пола. С другой стороны, пауки-самцы подразделяются на два типа: доминантный (D) и недоминантный (ND). Группа D состоит из самцов, чей показатель приспособленности лучше, по сравнению с остальной частью самцов (ND). В процессе оптимизации новое положение самцов моделируется следующим образом:

$$ms_i^{k+1} = \begin{cases} ms_i^k + \alpha \cdot V_{i,f} \cdot (s_f - ms_i^k) + \delta(rand - \frac{1}{2}), & ms_i \in D \\ ms_i^k + \alpha \cdot \left(\frac{\sum_{h \in ND} ms_h^k \cdot w_h}{\sum_{h \in ND} w_h} - ms_i^k \right), & ms_i \in ND \end{cases} \quad (5)$$

Фаза спаривания. Заключительная фаза итерации заключается в порождении новой особи в результате спаривания пауков из группы D с особями женского пола в некотором радиусе R от них. Чем больше вес паука, тем больше вероятность его влияния на новую особь. После генерации нового паука его сравнивают с худшим пауком в популяции; если он лучше по значению fitness-параметра, то новый паук заменяет худшего паука.

Оценка производительности. Алгоритм был реализован на языке программирования MATLAB и сравнивался с двумя другими популярными в литературе алгоритмами роевого интеллекта: алгоритмом пчелиной колонии и методом роя частиц.

При тестировании алгоритмов размер колонии был установлен 50, ограничение на количество итераций 1000. Уникальные параметры для каждого алгоритма были установлены в соответствии с рекомендациями [1-3]. Тестирование алгоритмов производилось на тестовых функциях [4], в числе которых были функции с квадратичными и неквадратичными зависимостями, функция с большим количеством локальных минимумов, овражные функции.

Тестирование алгоритмов на тестовых функциях показало, что по результатам 30 прогонов среднее количество итераций, необходимое алгоритмам для достижения оптимального значения с заданной точностью, меньше для алгоритма SSO.

На функциях с высокой степенью овражности все алгоритмы не дали удовлетворительных результатов, что характерно для многих алгоритмов оптимизации в целом.

Выводы. На основании изложенного можно сделать вывод, что алгоритм социального паука эффективен при решении задач оптимизации, на тестовых функциях показал результаты лучше, чем алгоритм пчелиной колонии и метод роя частиц. Однако алгоритм требует настройки параметров и неприменим к овражным задачам высокого порядка.

ЛИТЕРАТУРА

1. Cuevas E., Cienfuegos M. Social Spider Optimization Algorithm: Modifications, Applications, and Perspectives // Mathematical Problems in Engineering. – 2018.
2. Karaboga D., Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm // Journal of Global Optimization. – 2007. – №39. – С. 459–471.
3. Kennedy J., Eberhart R. Particle swarm optimization // Proceedings of the IEEE International Conference on Neural Networks. – 1995. – №4. – С. 1942-1948.
4. Черноруцкий И.Г. Методы принятия решений. / И.Г. Черноруцкий – Спб.: БХВ Петербург, 2005. – 417 с.

УДК 004.93

Русаков Е. С. (4 курс бакалавриата)
Черноруцкий И. Г., д. т. н., профессор

АНАЛИЗ СПУТНИКОВЫХ СНИМКОВ ГОРОДОВ С ПОМОЩЬЮ СВЁРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

Прикладная область городского планирования (планирование энергетики, инфраструктуры, инвестирование, и т. д.) требуют понимания уже имеющейся инфраструктуры и её окружения, например, как физических особенностей (количества зелёных насаждений, доступных для застройки площадей и геометрии и т. д.), так и более абстрактных понятий, как, например, классы землепользования (заключающие в себе экспертное понимание социально-экономической пользы).

Классификация землепользования важна для прикладных задач от городского планирования, зонирования пространств и выдачи предпринимательских разрешений до строительства и развития инфраструктуры. Классификация использования земли обычно основывается на экспертной оценке. Таким образом, это задача трудоёмка, выполняется нечасто, медленна и затратна. Как результат, такие данные доступны в основном в развитых странах и больших городах, имеющих потребность в них и ресурсы, необходимые для сбора и обработки таких данных. Во многих же менее богатых регионах, включая многие активно развивающиеся страны, где эта информация больше всего нужна, она недоступна.

Алгоритмы глубокого обучения лишь недавно стали применяться для анализа спутниковых снимков. Таким образом, область литературы по использованию этих инструментов в целях определения класса землепользования ещё только зарождается.

В данной работе рассматривается применимость алгоритма свёрточных нейронных сетей к задаче классификации городской застройки. Используются предобученные для распознавания простых форм нейронные сети архитектур VGG-16 и ResNet, как показавшие себя наиболее эффективно в решении других задач классификации изображений. Набор данных создаётся на основе данных из европейского исследование Urban Atlas [1] и снимков, полученных с помощью Google Maps API. Каждый снимок покрывает площадь 250×250 м^2, и ему присваивается класс из Urban Atlas, наиболее представленный по площади.

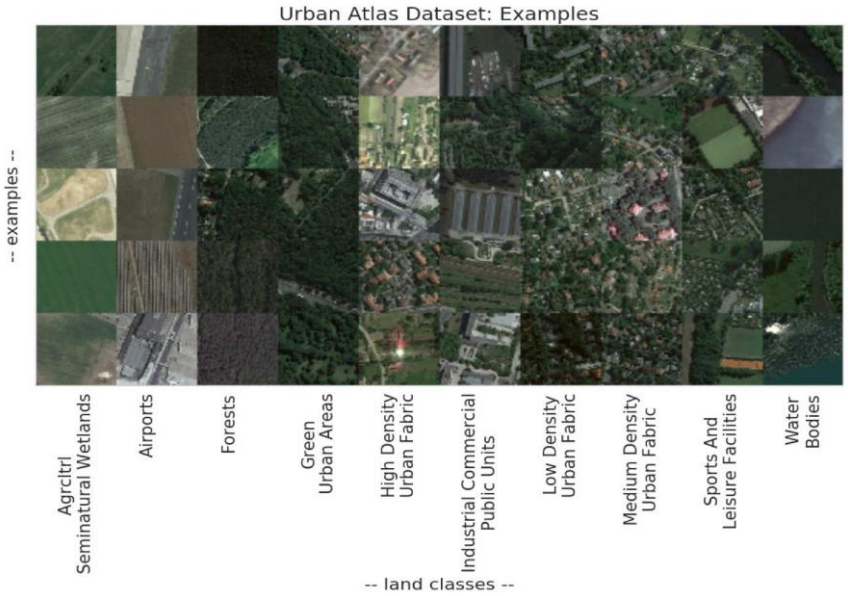


Рисунок 1 – Примеры изображений построенного по Urban Atlas

Препятствием является несбалансированность получаемого набора данных. Так, например, класс аэропортов представлен очень мало, по сравнению с, водоёмами, зонами озеленения или жилой застройкой.

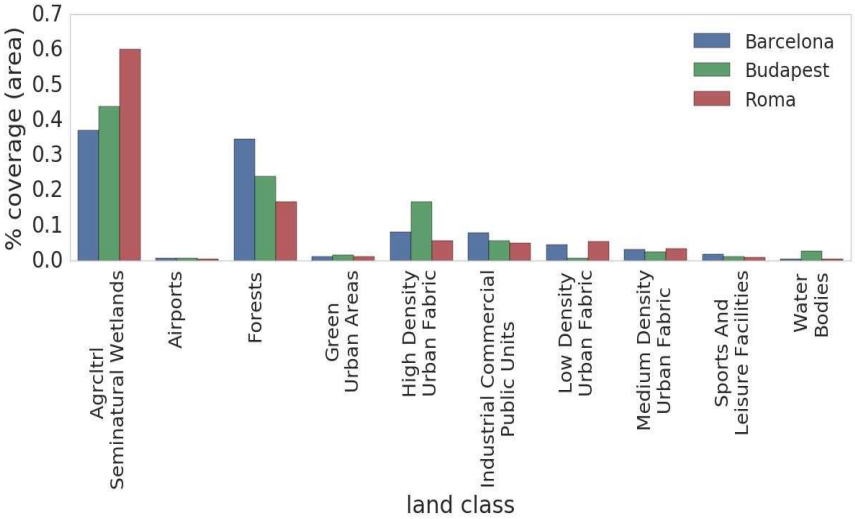


Рисунок 2 – Разница в представленности земель по классам на примере трёх городов

Ожидаемые результаты:

Поскольку назначение зданий и их внешний вид на снимках сильно различаются, от метода не ожидается очень высокая точность. Кроме того, стиль городской архитектуры также сильно меняется от города к городу, отчего сеть, обученная на одном городе, будет показывать худшие результаты на городах, на данных из которых она не обучалась.

Таким образом, для создания большой экспертной системы оценки классов землепользования потребуется очень большой набор данных о городах с разными архитектурными стилями.

ЛИТЕРАТУРА

1. Urban Atlas — Copernicus Land Monitoring Service // URL: land.copernicus.eu/local/urban-atlas
2. Google Maps API // URL: developers.google.com/maps
3. Гудфеллоу Я., Бенджио И., Курвилль А. − Глубокое обучение, 2017 г.

УДК 004.9

Сабуткевич А.М. (4 курс бакалавриата)
Самочадин А.В., к.т.н., доцент

ПРОГРАММНЫЕ СРЕДСТВА ОРГАНИЗАЦИИ КОНТРОЛЬНЫХ ТЕСТИРОВАНИЙ НА ОСНОВЕ МЕТОДИКИ АДАПТИВНОГО ОБУЧЕНИЯ

Применение методики адаптивного обучения является одним из способов повышения эффективности всех составляющих образовательного процесса. Информационные технологии являются основным инструментом для реализации данной методики [1].

Целью работы является разработка системы, нацеленной на проведение контрольных тестирований обучающихся. В процессе тестирования обучающемуся предлагаются вопросы с вариантами ответов и рекомендательные учебные материалы в рамках электронной системы обучения. Применение адаптивной методики подразумевает рассмотрение архитектуры системы на основе следующих моделей: предметной области, адаптации и обучающегося [2]. В связи с тем, что данная система рассчитана на использование в течение короткого интервала времени, отведенного на проведение тестирования, то накапливание статистических данных в

рамках модели обучающегося не предусматривается. Архитектура системы, основанная на моделях предметной области и адаптации, приведена на рисунке 1.

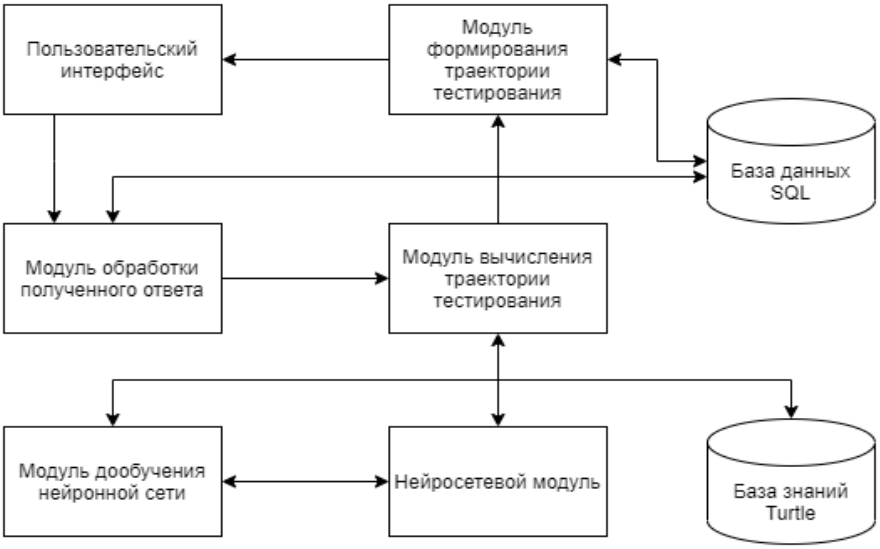


Рисунок 1 – Архитектура системы по осуществлению контрольного тестирования на основе методики адаптивного обучения.

В качестве языка реализации системы был выбран C#, пользовательский интерфейс был разработан с использованием Windows Forms. Модель предметной области реализована с использованием следующих средств:

– базы знаний, определяющей правила объединения элементов в персонализированную траекторию тестирования, с использованием нотации N3. Все знания, хранящиеся в базе знаний, представлены в нотации Turtle. Для определения правил взаимосвязи между элементами траектории используются динамические запросы на языке SPARQL. Для интеграции базы знаний в систему была использована библиотека dotNetRDF.

– базы данных, хранящей символьное и семантическое представление элементов траектории и реализованной с использованием языка SQL и системы управления MySQL.

В рамках системы траектория тестирования представлена в виде матрицы, размерность которой соответствует количеству допустимых вариантов ответа и количеству вопросов. Модель адаптации основывается на модуле вычисления траектории тестирования и нейросетевом модуле. В условиях выбранной структуры данных в качестве нейронной сети была выбрана архитектура Hopfield network, реализованная с использованием языка C++. Задачей модуля нейронной сети является сопоставление полученной матрицы с имеющимися образцами, и определение эталонного образца, которому соответствует полученный в условиях устойчивости к изменениям. Инструкции по запуску процесса, соответствующего нейронной сети из основной системы, хранятся в базе знаний. В том случае, если модулю нейронной сети не удается однозначно определить рекомендованную траекторию тестирования, вызывается дополнительный модуль дообучения, который при помощи динамических запросов к базе знаний на языке SPARQL определяет правила построения траектории в условиях анализируемых действий обучающегося.

ЛИТЕРАТУРА

1. Vagale V., Niedrite L., Ignatjeva S.: The Architecture of the Personalized Adaptive E-Learning System. In Lupeikiene A., Matulevičius R., Vasilecas O. (eds.): Baltic DB&IS 2018 Joint Proceedings of the Conference Forum and Doctoral Consortium (2018)
2. Ahmed M. U., Sangi N. A., Mahmood A. A.: Model of adaptive e-learning in an ODL environment. Mehran University Research Journal of Engineering and Technology 37(2), 367-382 pp. (2018)

УДК 004.8

Сатюков Е. Д. (4 курс бакалавриата)
Уткин Л. В., д.т.н., профессор

РАЗРАБОТКА И ИССЛЕДОВАНИЕ МЕТОДА ОБЪЯСНЕНИЯ ПРЕДСКАЗАНИЙ МОДЕЛИ МАШИННОГО ОБУЧЕНИЯ С ЦЕНЗУРИРОВАННЫМИ ДАННЫМИ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ

Быстрое развитие методов машинного обучения и искусственного интеллекта привело к созданию большого числа достаточно эффективных моделей, решающих задачи в самых различных областях. Одним из классов таких моделей машинного обучения являются модели выживаемости, которые нашли широкое применение в медицине, анализе надежности сложных систем, безопасности и т. д. В то же время, большинство таких моделей, например, нейронные сети или композиции, являются «черными ящиками» в том смысле, что для них известны только вход (примеры) и соответствующий выход (предсказания), а вопрос, как получено то или иное предсказание, что повлияло на него, остается открытым. Это ограничивает применение соответствующих моделей машинного обучения, снижает доверие к результатам их функционирования. Одним из путей преодоления этого недостатка является разработка методов и моделей объяснения и интерпретации предсказаний, которые объединяются под общим названием «объяснительный интеллект» и отвечают на вопрос, какие признаки входных данных отвечают за полученные предсказания. При этом различают локальные и глобальные объяснения, когда соответственно объясняется предсказание конкретного примера, например, диагноз одного пациента, и в среднем предсказания по всей обучающей выборке.

В настоящее время существует множество методов объяснения, но практически нет методов, которые бы объясняли предсказания моделей выживаемости, а существующие методы (SurvLIME, SurvLIME-KS [1]) являются достаточно сложными с вычислительной точки зрения и решают только задачу локального объяснения. Поэтому в работе предлагается новый метод, названный SurvNAM, который основан на комбинации идеи нейронной аддитивной модели (нейронной сети специального вида – neural additive model – NAM [2]) и идеи метода SurvLIME. Нейронная сеть, реализующая NAM, обучается при помощи специальной функции потерь, которая в свою очередь реализует расширение известной модели Кокса на случай обобщенной аддитивной модели. Такое обучение позволило за достаточно короткое время получить функции влияния каждого признака входных данных, которые как раз и показывают, как влияет каждый признак на предсказание модели выживаемости в виде кумулятивной функции риска.

Для реализации и исследования (тестирования) предлагаемого метода были разработаны две программы на языке Python 3.7, для глобального и локального объяснения. Реализация первой состоит из следующих этапов:

- обработка данных с помощью библиотек numpy, pandas, sklearn.preprocessing и skusrv.preprocessing: загрузка необходимого датасета, разрешение проблемы NaN (Not a Number) значений, кодирование категориальных признаков;
- создание и обучение «черного ящика», в качестве которого выступает RSF (Random Survival Forest) из библиотеки sksurv.ensemble;
- получение базовой кумулятивной функции риска с помощью модели Кокса из библиотеки sksurv.linear_model, обученной на тех же данных, что и RSF;
- вычисление с помощью обученной модели RSF кумулятивных функций риска для примеров из исходного датасета, которым будет обучаться нейронная сеть, реализующая NAM;
- создание и обучение нейронной сети на основе NAM с помощью библиотеки TensorFlow;

– сравнение полученной модели SurvNAM с «черным ящиком» RSF на основе метрики C-index (concordance index) из библиотеки sksurv.metrics;
– визуализация для каждого признака полученной функции влияния с помощью библиотеки matplotlib.pyplot, а также представление кумулятивных функций риска и функций выживаемости для нескольких примеров из исходных данных.

Реализация второй программы отличается наличием дополнительного этапа после обучения RSF и модели Кокса, суть которого состоит в генерации новых данных вокруг выбранного из исходного датасета примера, используя нормальное распределение из библиотеки numpy.random для каждого некатегориального признака со средним, равным значению этого же признака для интерпретируемого примера, и с заданным среднеквадратическим отклонением. С помощью RSF для сгенерированных данных вычисляются кумулятивные функции риска, на основе которых далее обучается нейронная сеть, реализующая NAM.

Разработанный метод был применен на нескольких реальных наборах с цензурированными данными:
– German Breast Cancer Study Group 2 (выживаемость при раке молочной железы);
– Stanford Heart Transplant data (выживаемость участвующих в Стэнфордской программе трансплантации сердца);
– Monoclonal Gammopathy Data (выживаемость при моноклональной гаммапатии неопределенного значения);
– Survival from Malignant Melanoma (выживаемость при злокачественной меланоме);
– The Veterans' Administration Lung Cancer Trial (выживаемость при запущенном неоперабельном раке легких).

Для каждого датасета были получены соответствующие функции влияния входных признаков, а также кумулятивные функции риска и функции выживаемости.

Рассмотрим на примере датасета Stanford Heart Transplant data (в нем всего два входных признака) полученные для глобального объяснения функции, показывающие, насколько каждый признак влияет на предсказание «черного ящика».

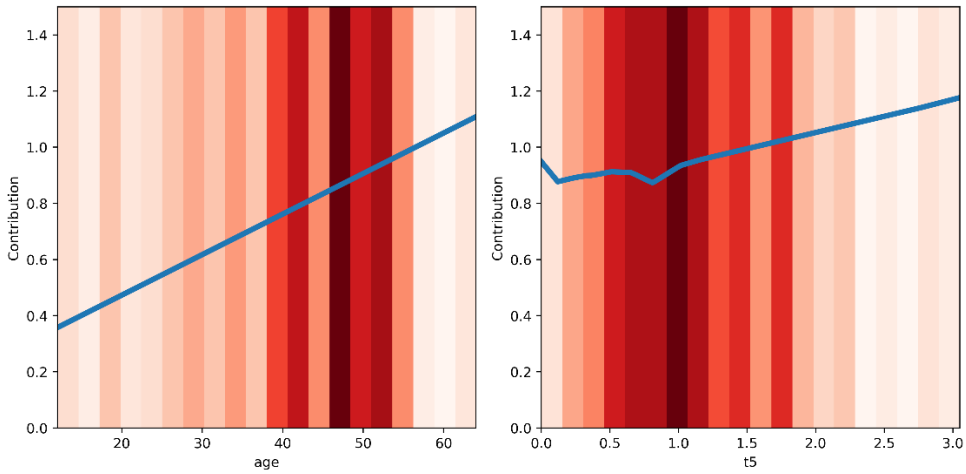


Рисунок 1– Функции влияния входных признаков в случае глобального объяснения для датасета Stanford Heart Transplant data. График раскрашен в соответствии с плотностью данных по каждому признаку. Чем больше данных для этого значения признака, тем темнее полоса.

Из рисунка видно, что наиболее значимым признаком является 'age', так как соответствующая функциональная зависимость изменяется в наибольшей степени.

ЛИТЕРАТУРА

1. M.S. Kovalev and L.V. Utkin. A robust algorithm for explaining unreliable machine learning survival models using the Kolmogorov-Smirnov bounds. arXiv:2005.02249, May 2020.
2. R. Agarwal, N. Frosst, X. Zhang, R. Caruana, and G.E. Hinton. Neural additive models: Interpretable machine learning with neural nets. arXiv:2004.13912, April 2020.

УДК 004.023

Семакина Д.Н. (4 курс бакалавриата)
Черноруцкий И.Г., д.т.н., профессор

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ МЕТОДА ПОИСКА ГАРМОНИИ ДЛЯ НАХОЖДЕНИЯ ЭКСТРЕМУМОВ ЖЕСТКИХ ЦЕЛЕВЫХ ФУНКЦИОНАЛОВ

С развитием технологий инженерные задачи оптимизации все более усложняются и чаще всего имеют нелинейные и невыпуклые целевые функции с жесткими ограничениями на равенства и неравенства. Решение таких задач традиционными методами оптимизации может привести к большим трудностям, так как аналитические методы чаще всего предназначены для узкого круга задач, где необходимо знать пространство поиска, поэтому широкое распространение получили алгоритмы метаэвристической оптимизации, которые не накладывают никаких ограничений на целевую функцию, реализуя случайный поиск возможных решений.

Один из таких алгоритмов – метод поиска гармонии (HS – Harmony Search), который основывается на подражании процессу импровизации музыкантов в поиске приятной гармонии, разработанный в 2001 году ученым Zong Woo Geem. Этот метод выбран для рассмотрения, так как он легко реализуем и при этом отлично иллюстрирует работу метаэвристических алгоритмов. HS оказался эффективной альтернативой аналитическим методам, поэтому были разработаны его различные модификации для решения самых разнообразных задач.

Целью работы является оценка производительности метода поиска гармонии в условиях жесткости целевых функционалов. В работе решаются следующие задачи: изучение концепции алгоритма поиска гармонии, реализация алгоритма и оценка его производительности на тестовых функциях разной степени овражности.

За основу исследования взята статья Harmony Search Algorithm: Basic Concepts and Engineering Applications, ее авторы: Alireza Askarzadeh и Esmat Rashedi. В статье рассмотрен базовый алгоритм поиска гармонии и его применение в различных сферах.

В рассматриваемом методе допустимое решение называется гармонией, и каждая переменная в решении соответствует ноте. В начале работы алгоритма в память гармоний записывается N случайных гармоний и значение целевого функционала для них.

Чтобы сымпровизировать новую гармонию, для каждой ее компоненты выполняется следующее: генерируется случайное число, равномерно распределенное от нуля до единицы. Если это число меньше, чем параметр HMCR, заданная вероятность выбора значений из памяти гармонии, то в текущую компоненту записывается соответствующая компонента из случайно выбранной гармонии из памяти, в противном случае компонента генерируется случайно.

Чтобы выполнить механизм регулировки шага, снова генерируется случайное число rand. Если оно меньше либо равно заданной скорости регулировки основного тона PAR, то импровизированная нота сдвигается на соседнее значение в соответствии с формулой (1), где bw – еще один параметр алгоритма, задающий величину модификации значения в пределах от l до u:

$$x_{new,j}=x_{new,j}+bw*(rand-0.5)*|u_j-l_j| \quad (1)$$

Если значение целевого функционала для новой гармонии лучше значения целевого функционала для худшей гармонии из памяти, худшая гармония из памяти заменяется на новую. Импровизация новой гармонии повторяется, пока не достигнуто минимальное значение целевой функции с заданной точностью или заданное максимальное количество импровизаций. Лучшая гармония, сохраненная в памяти, возвращается как оптимальное решение.

Алгоритм был реализован на языке MATLAB в среде MATLAB R2020b.

На вход алгоритма подается ссылка на целевую функцию двух переменных, верхняя и нижняя границы переменных, размер памяти гармоний, параметры алгоритма – HMCR, PAR, bw и максимально возможное количество импровизаций. Также задается значение исследуемой функции в точке минимума и точность, с которой алгоритм должен к ней приблизиться.

На выходе программы выводится количество итераций необходимое для достижения точки минимума с заданной точностью или максимальное количество итераций, если точка минимума не была найдена при данной точности, а также значения переменных решения в найденной точке и значение целевой функции. Также строится зависимость значения целевой функции от количества итераций.

В качестве тестовой функции была выбрана «банановая долина» Розенброка, на которой можно проиллюстрировать работу алгоритма в условиях жесткости целевых функционалов, так как линии уровня этой функции имеют овражную структуру с криволинейным дном оврага в виде параболы.

$$F=A\left(x_1^2-x_2^2\right)+(1-x_1)^2;\ x^*=(1;1);\ F\left(x^*\right)=0. \qquad (2)$$

Сначала программа была проверена на классической функции Розенброка (в формуле (2) A = 100), затем коэффициент A был заменен на 10^3, 10^5 и 10^7, чтобы посмотреть, как ведет себя алгоритм при увеличении степени овражности целевого функционала.

Трудоемкость алгоритма оценивается в Горнерах. Так как одной импровизации соответствует однократное вычисление значения целевой функции, количество Горнеров равно числу импровизаций. Полученные результаты приведены в Таблице 1.

Таблица 1 – количество Горнеров, необходимых для вычисления минимума тестовой функции с различными коэффициентами A с точностью 10^{-4}

Коэффициент A в формуле (2)	100	10^3	10^5	10^7
Количество Горнеров	29318	97395	893062	3318223

В ходе работы сделаны следующие выводы:

– алгоритм поиска гармонии успешно находит точку минимума с заданной точностью в определенном небольшом диапазоне;

– алгоритм остается эффективным даже для нетривиальных функций – при увеличении коэффициента 100 в «банановой долине» Розенброка до 10^7 алгоритм успешно находит решение, однако при увеличении коэффициента больше чем 10^5 трудоемкость алгоритма существенно возрастает – количество Горнеров становится на порядок больше;

– при большом диапазоне по x1 и x2 – от -1000 до 1000 алгоритм ошибочно заклинивает в неверной точке, в таком случае может понадобиться корректировка параметров алгоритма;

– в случае небольшого пространства поиска алгоритм эффективно обнаруживает потенциальное решение, но способность к локальному поиску у данного метода слабая, и при увеличении количества импровизаций лучшего решения не ожидается.

Таким образом, алгоритм поиска гармонии является эффективным алгоритмом для решения сложных задач оптимизации с жесткими целевыми функционалами, он прост в реализации, быстро попадает в окрестность точки минимума, но не подходит для поиска решений высокой точностью.

ЛИТЕРАТУРА

1. Черноруцкий И. Г., Воинов Н. В., Котлярова Л. П. Методы оптимизации. - СПб.: СПбПУ, 2020. - 167 с.
2. Alireza Askarzadeh, Esmat Rashedi Harmony Search Algorithm // Recent Developments in Intelligent Nature-Inspired Computing. 2017.
3. X.-S. Yang Harmony Search as a Metaheuristic Algorithm // Music-Inspired Harmony Search Algorithm: Theory and Applications. 2009.
4. Harmony Search Algorithm URL: https://sites.google.com/a/hydroteq.com/www/

УДК 004.023

Соколова А. А. (4 курс бакалавриата)
Черноруцкий И. Г., д. т. н., профессор

ИССЛЕДОВАНИЕ АЛГОРИТМА СВЕТЛЯЧКОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ ОПТИМИЗАЦИИ

Целью работы является реализовать данный алгоритм и протестировать его перспективность в решении нелинейных ограниченных оптимизационных задач.

Алгоритм светлячков был разработан Янгом в 2007 г. и основывается на следующем поведении:

- светлячки привлекают друг друга вне зависимости от своего пола;
- привлекательность светлячка для других особей пропорциональна его яркости;
- менее привлекательные светлячки перемещаются по направлению к более привлекательному светлячку;
- яркость излучения данного светлячка, видимая другим светлячком, уменьшается с увеличением расстояния между светлячками;
- если светлячок не видит возле себя светлячка более яркого, чем он сам, то он перемешается случайным образом.

После реализации на языке программирования MATLAB, алгоритм был протестирован на функции Экли:

$$f(x) = -20\exp[-\frac{1}{5}\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}] - \exp[\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)] + 20 + e.$$

В случае d = 2, где d – количество измерений, имеем 4 равных максимума при $(\frac{1}{2}, \frac{1}{2})$, $(\frac{1}{2}, -\frac{1}{2})$, $(-\frac{1}{2}, \frac{1}{2})$, $(-\frac{1}{2}, -\frac{1}{2})$ и уникальный глобальный минимум при (0, 0). Эти точки показаны на рисунке 1.

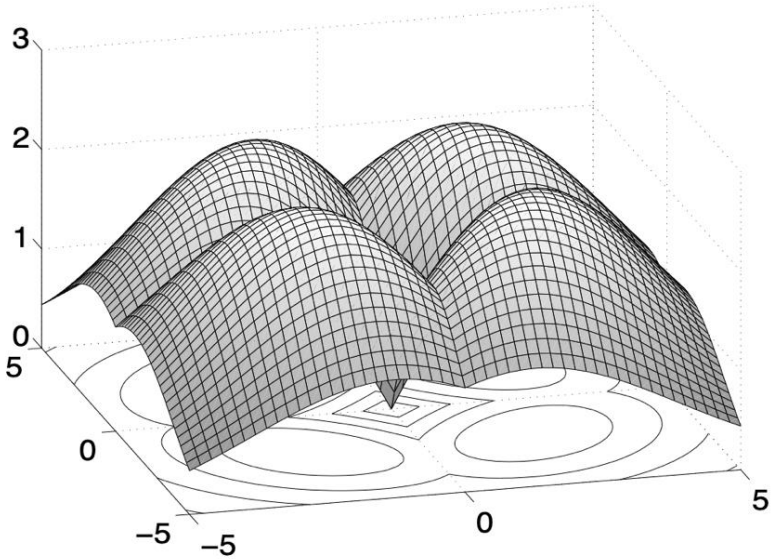


Рисунок 1 – Четыре глобальных минимума в (±1/2, ±1/2).

Их можно найти с помощью реализованного алгоритма с 25 светлячками. Начальные положения показаны на рисунке 2, а их конечные положения после 20 итераций – на рисунке 3.

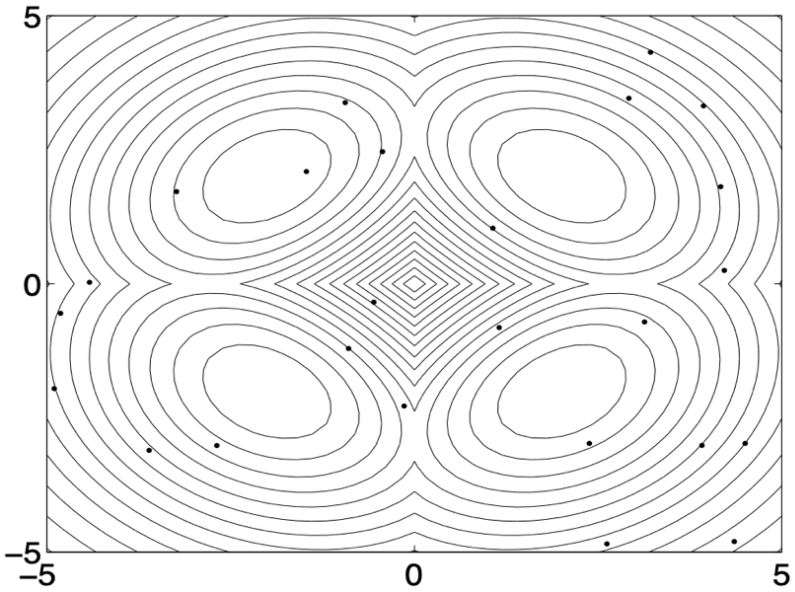


Рисунок 2 – Начальные позиции 25 светлячков.

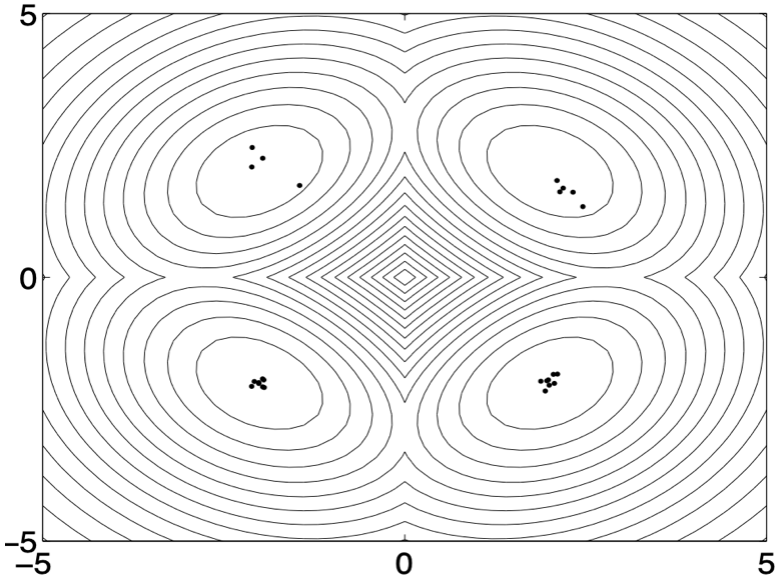


Рисунок 3 – Конечные позиции после 20 итераций.

В конечном счете можем сделать вывод, что алгоритм светлячков является очень эффективным.

ЛИТЕРАТУРА

1. Lukasik, S. and Zak, S. (2009) ‘Firefly algorithm for continuous constrained optimization tasks’, ICCCI 2009, Lecture Notes in Artificial Intelligence (Eds. N. T. Ngugen, R. Kowalczyk, S. M. Chen), 5796, pp. 97-100.

2. Yang, X. S. (2009) ‘Firefly algorithms for multimodal optimization’, in: Stochastic Algorithms: Foundations and Applications (Eds O. Watanabe and T. Zeugmann), SAGA 2009, Lecture Notes in Computer Science, 5792, Springer-Verlag, Berlin, pp. 169-178.

УДК 004.838.2

Стояноски К. (4 курс бакалавриата)
Дробинцев П.Д., к.т.н., доцент

ГОТОВЫЕ РЕШЕНИЯ ДЛЯ ИСПОЛЬЗОВАНИЯ МАШИННОГО ОБУЧЕНИЯ В МОБИЛЬНЫХ ПРИЛОЖЕНИЯХ С ИСПОЛЬЗОВАНИЕМ FIREBASE ML KIT

Качественное машинное обучение для решения задач – дело длительное, трудоемкое и затратное. Начало работы с машинным обучением может быть трудным для многих разработчиков. Новые ML разработчики проводят бесчисление часы, изучая тонкости внедрения моделей ниского уровня итд. Даже для опытного специалиста, адаптация и оптимизация моделей для работы на мобильных устройствах может быть огромной задачей. Поэтому возникла потребность в полуготовых решениях, которые можно внедрить в функционал любого продукта и просто использовать. Задача готового решения – предоставить уже готовый набор функций, который можно просто интегрировать в мобильное приложение, внеся несколько строчек кода и доучивать его в процессе работы.

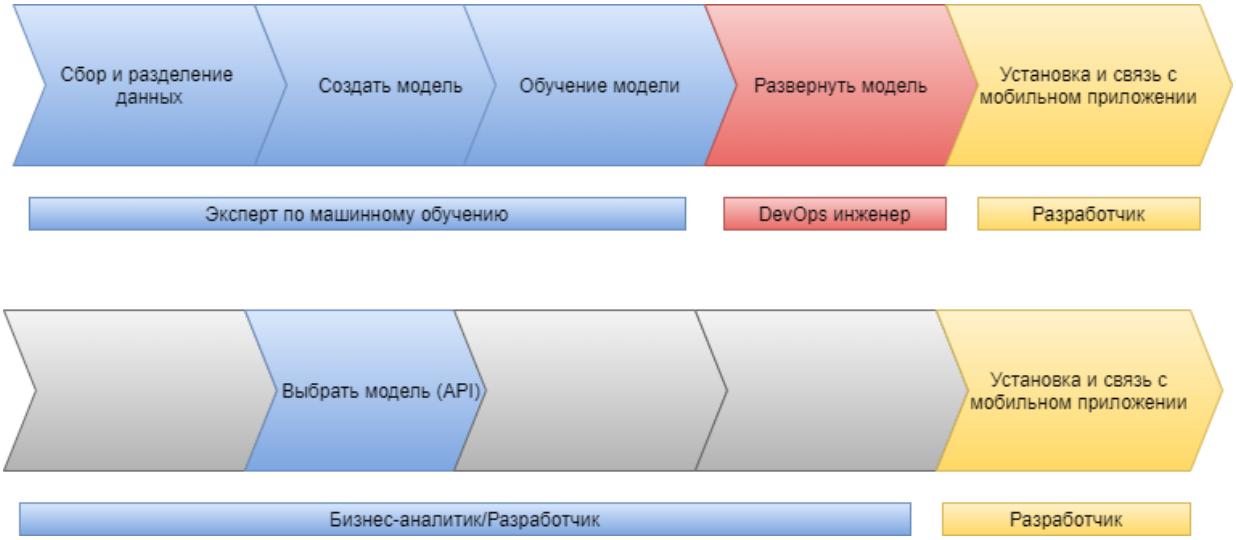


Рисунок 1 – Разница между обучением AI с нуля и использование готового продукта[1].

Есть несколько способов реализовать машинное обучение в Android, например, создать и обучить собственную модель. Но многие разработчики сталкиваются с трудностями при реализации даже низкоуровневых моделей в машинном обучении. Кроме того, процесс обучения требует много времени и средств.

На Рисунке 1 представленая разница между обучением AI с нуля и использование готового продукта. Если мы решим сначала обучить новую модель, потребуется эксперт по машинному обучению для создания и обучения модели. Затем нам нужен инженер DevOps для развертывания модели, чтобы мобильный разработчик мог использовать ее в своем приложении. Чтобы пропустить все эти ненужные шаги, Firebase создал самый мощный и удобный способ реализации машинного обучения – Firebase ML Kit.

ML Kit - это мобильный SDK, который предоставляет опыт машинного обучения Google для приложений Android и iOS в мощном, но простом в использовании пакете.[2] ML Kit соединяет технологий машинного обучения от Google, как Google Cloud Vision API, TensorFlow Lite и Android Neural Networks API в одном пакете. Все функции продукта могут быть доступны как в приложении, так и в облаке через API (во втором случае обработка запроса потребует больше времени, но обеспечит более точный результат).

ML Kit дает множество преимуществ в мире машинного обучения. Основным преимуществом является то, что Firebase ML Kit обеспечивает удобную среду для внедрения машинного обучения бесплатно. Разработчикам нужно написать всего несколько строк кода и передать данные в ML Kit API. В зависимости от того какая модель была выбрана, результаты будут видны через несколько милисекунд. Стоит отметить, что некоторые API-интерфейсы также можно использовать в автономном режиме без подключения к Интернету. ML Kit в себя включает распознавание текста, рапознавание лиц, языка, точек на местности итд. Единственный недостаток ML Kit заключается в том, что он находится в бета-версии, поэтому разработчикам приходится сталкиваться с некоторыми проблемами при его реализации.

Машинное обучение - это только начало автоматизации. Компании использовали ML в течение последних несколько лет в своих интересах, и Firebase ML Kit делает то же самое. Использование готового инструмента Firebase ML Kit позволяет существенно сократить время и снизить расходы разработки. Согласно статистике, 13% компаний затрачивают от 90-365 дней на создание модели машинного обучения, у 22% это занимает более месяца, у 28% от 8 до 30 дней, и лишь 14% могут сделать это за 7 дней и меньше.[3] Таким образом Firebase ML Kit создана для существенного упрощения процесса создания готового продукта и способствует его более скорому выходу на рынок.

ЛИТЕРАТУРА

1. Составлено по: Официальной документаций от Google для Firebase ML Kit. [В Интернете]. Available: https://firebase.google.com/docs/ml-kit/
2. Официальная документация от Google для Firebase ML Kit. [В Интернете]. Available: https://firebase.google.com/docs/ml-kit/
3. Иследовательская работа Algorithmia's "2020 State of Enterprise ML". [В Интернете]. Available: https://info.algorithmia.com/hubfs/2019/Whitepapers/The-State-of-Enterprise-ML-2020/Algorithmia_2020_State_of_Enterprise_ML.pdf

УДК 004.023

Султанова Д. В. (4 курс бакалавриата)
Черноруцкий И. Г., д.т.н., профессор

ИССЛЕДОВАНИЕ АЛГОРИТМА ЛЕТУЧЕЙ МЫШИ ДЛЯ РЕШЕНИЯ ЗАДАЧ ОПТИМИЗАЦИИ

Целью работы является рассмотрение метаэвристического алгоритма оптимизации, вдохновленного эхолокационным поведением летучих мышей – Bat Algorithm.

Во время полета и охоты летучие мыши излучают короткие ультразвуковые импульсы в окружающую среду и улавливают отраженные сигналы. Таким образом, информация, полученная с помощью эхо-сигналов, позволяет летучим мышам построить точное изображение своего окружения и точно определить расстояние, форму и местонахождение потенциальной жертвы. Возможность такой эхолокации летучих мышей удивительна, поскольку они могут находить свою добычу и различать различные виды насекомых даже в полной темноте.

Алгоритм летучей мыши подчиняется нескольким правилам:

- все летучие мыши используют эхолокацию, чтобы анализировать расстояние, а также иметь различие между добычей и природными препятствиями;

- летучие мыши перемещаются случайным образом со скоростью v_i в позицию x_i с частотой f_{min}, изменяемой длиной волны и громкостью A_0 для поиска добычи. Они могут автоматически регулировать длину волны (или частоту) испускаемого импульса и регулировать его скорость $r \in [0, 1]$ в зависимости от близости их цели;

- хотя громкость может варьироваться разными способами, мы предполагаем, что громкость изменяется от большого (положительного) A_0 до минимального постоянного значения A_{min}.

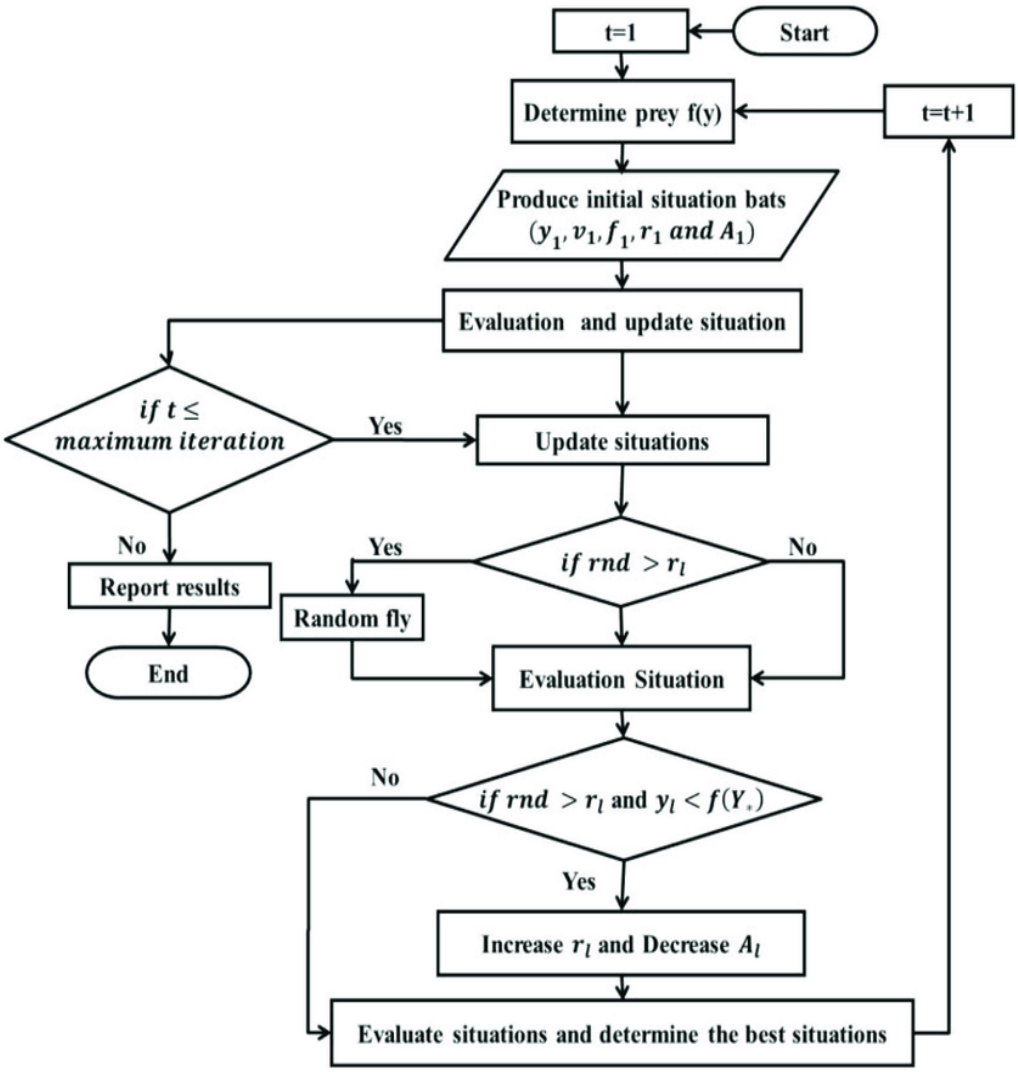


Рисунок 1 – Блок-схема алгоритма летучей мыши

Для проверки производительности данного алгоритма предполагается провести два численных эксперимента. Первый – это сравнение алгоритма летучей мыши с другими стандартными метаэвристическими алгоритмами на классических тестовых функциях. Для второго был рассмотрен набор тестов CEC'2005. Все функции CEC'2005 генерируются из основных функций, перечисленных в (Suganthan et al., 2005) путем смещения, вращения или гибридизации этих функций.

ЛИТЕРАТУРА

1. Yang, X.-S. (2010b). A New Metaheuristic Bat-Inspired Algorithm. In J. González, D. Pelta, C. Cruz, G. Terrazas & N. Krasnogor (Eds.), *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)* (Vol. 284, pp. 65-74): Springer Berlin Heidelberg.

2. Suganthan, P., Hansen, N., liang, J., Deb, K., Chen, Y., Auger, A., & Tiwari, S. (2005). *Problem definitions and evaluation criteria for the CEC'2005 special session on real parameter optimization*: KanGAL Report, 2005005.

УДК 004.42

Шалгуева С. Л. (4 курс бакалавриата)
Воскобойников С. П., к.ф.м.н., доцент
Попов Е.Н., к.ф.м.н., н.с.

РАЗРАБОТКА ПРОГРАММЫ ДЛЯ МОДЕЛИРОВАНИЯ ВЗАИМОДЕЙСТВИЯ ЭЛЕКТРОМАГНИТНЫХ РАССЕИВАТЕЛЕЙ

Задача о коллективных эффектах внутри ансамбля электромагнитных рассеивателей имеет большую актуальность в связи с развитием прецизионного измерительного оборудования. Современные зонды для наблюдения точечных электромагнитных излучателей, атомов или молекул, должны быть рассчитаны на слабое детектируемое поле, до нескольких фотонов. Поэтому расположение диполей на поверхности зонда следует подбирать таким образом, чтобы мощность рассеиваемого поля слабо отличалось от мощности детектируемого поля. Также задача представляет большой интерес для фундаментальной науки, так как в ней моделируется процесс рассеяния электромагнитного поля группой диполей, связанных между собой ближним полем.

Целью данной работы является разработка программы, позволяющей проводить моделирование взаимодействия электромагнитных рассеивателей в условиях монохроматической накачки.

Математическая модель взаимодействия N рассеивателей в условиях монохроматической накачки представляет собой систему из N нелинейных дифференциальных уравнений второго порядка, где каждое уравнение описывает динамику возбуждения i-того рассеивателя y_i:

$$\frac{d^2y_i}{dt^2} + \gamma\frac{dy_i}{dt} + \omega^2 y_i = \frac{e}{m_e}Esin(\omega t) + \frac{e}{m_e}\sum_{1,i\neq m}^{N} E_{mi} \quad (1)$$

$$E_{mi} = \frac{ec^2\left(\frac{y_m}{r_{mi}}+\frac{v_m}{c}\right)+ea_m r_{mi}}{c^2 r_{mi}^2\left(1+\frac{y_m v_m}{r_{mi}c}\right)^3} \quad (2)$$

$$r_{mi} = |x_m - x_i| \quad (3)$$

$$v_m = \frac{dy_m}{dt},\ a_m = \frac{dv_m}{dt} = \frac{d^2y_m}{dt^2} \quad (4)$$

где γ – константа затухания, ω – собственная частота рассеивателя, e – заряд электрона, m_e – масса электрона, c – скорость света, E – амплитуда напряжённости внешнего поля, E_{mi} – нелинейная неоднородность, вызванная влиянием поля рассеивателя m на рассеиватель i. Функция y_i имеет смысл смещения электрона относительно положения равновесия, v_i – его скорость, a_i – ускорение.

В задаче принято, что все рассеиватели находятся на отрезке единичной длины в точках с координатами x_i:

$$0 \leq x_i \leq 1 \quad (5)$$

В начальный момент времени все смещения y_i и скорости v_i равны нулю. Параметры задачи имеют следующие численные значения: $e = 4.8 \cdot 10^{-10} СГСЭ$, $c = 3 \cdot 10^{10}$см/с, $\omega = 10^{12}c^{-1}, \gamma = 10^8 c^{-1}, m_e = 9.1 \cdot 10^{-28} г, E = 3.3 \cdot 10^{-4} В/см$.

Так как большинство существующих алгоритмов решения систем нелинейных дифференциальных уравнений работают с системами первого порядка, то система (1) путём ввода замены (4) была преобразована к системе уравнений первого порядка размерности $2N$. Особенностью системы является её нелинейность за счёт третьей степени в знаменателе (2) и неразрешённость относительно производной (4). В общем виде система может быть записана так:

$$G(y', y, t) = 0 \quad (6)$$

В работе был проведён анализ существующих программ для решения жёстких систем [1] нелинейных дифференциальных уравнений вида (6), из которых были выбраны программы DDASSL и DDASPK из библиотеки NETLIB [2] являющихся эффективными для решения жёстких систем дифференциальных уравнений [3].

Программа DDASSL [3] основана на формулах дифференцирования назад. Программа DDASPK [4] так же использует комбинацию формул дифференцирования назад и предоставляет выбор из двух методов решения линейных систем алгебраических уравнений: прямого для матриц общего или ленточного вида, или итерационного метода обобщённых минимальных невязок GMRES для решения больших систем уравнений с возможностью предобусловливания.

В работе приводятся результаты применения программ для моделирования взаимодействия электромагнитных рассеивателей, сравнение результатов моделирования с использованием существующих программ и выводы о целесообразности применения различных подходов.

Разработанная программа будет использована для определения координат диполей, при которых поглощаемая за единицу времени энергия является минимальной.

Разработка программы осуществлена на языке программирования Fortran, входящего в состав MinGW gfortran, который интегрируется с Code::Blocks.

ЛИТЕРАТУРА

1. Ракитский Ю.В., Устинов С.М., Черноруцкий И.Г. Численные методы решения жёстких систем.М.:Наука, 1979. 199 с.
2. https://www.netlib.org
3. A description of DASSL: a differential/algebraic system solver, L. R. Petzold, SAND82-8637 Sandia National Laboratories, September 1982.
4. Peter N. Brown, Alan C. Hindmarsh, and Linda R. Petzold, Using Krylov Methods in the Solution of Large-Scale Differential- Algebraic Systems, SIAM J. Sci. Comput., 1994, pp. 1467-1488.

УДК 004.023

Шумковская Э.О. (4 курс бакалавриата)
Черноруцкий И.Г., д.т.н., профессор

РЕАЛИЗАЦИЯ УЛУЧШЕННОГО КИТОВОГО АЛГОРИТМА ОПТИМИЗАЦИИ И ИССЛЕДОВАНИЕ ЕГО ЭФФЕКТИВНОСТИ НА ЖЕСТКИХ ЦЕЛЕВЫХ ФУНКЦИОНАЛАХ

Китовый алгоритм оптимизации (WOA) – метаэвристический алгоритм поиска минимума функции, основанный на охотничьем поведении стаи горбатых китов. Улучшенный китовый алгоритм оптимизации (IWOA) является комбинацией китового алгоритма оптимизации и дифференциальной эволюции (DE). Улучшение решает проблему преждевременной сходимости исходного алгоритма к локальному оптимуму.

Цель работы – реализовать улучшенный китовый алгоритм оптимизации и протестировать его работу на жёстком целевом функционале.

В начале программы задаются следующие параметры: число особей в популяции NP, количество поколений t_max, размерность задачи n, вероятность мутации CR. Далее генерируется популяция X – двумерный массив случайных чисел размерностью NP×n. В цикле, пока число поколений t не достигло t_max, или не выполнен критерий останова, для каждого варианта решения X выполняются следующие действия:

1. Вычисляется параметр lam, который уменьшается от 1 до 0 с каждой итерацией, и генерируется случайное число p в диапазоне от 0 до 1.

2. Если p<lam, то с вероятностью CR выполняется оператор мутации из DE. Если не выполнилась мутация, то выполняется «поиск добычи» - приближение всех решений к случайному.

3. Если p≥lam, то с равной вероятностью выполняется «окружение добычи» - приближение всех решений к лучшему, или спиральное обновление позиции.

4. Новое решение сравнивается с текущим, и, если новое решение лучше текущего (значение целевого функционала меньше), заменяет его. Таким же образом обновляется лучшее решение.

На выходе программы получаем лучшее решение из имеющихся – точку, в которой значение целевого функционала близко или равно минимальному. Если у функции несколько точек минимума, алгоритм находит одну из них. Также выводится значение функции в найденной точке и трудоёмкость вычисления.

Для тестирования алгоритма была выбрана функция Розенброка

$$F = (1 - x_1)^2 + A * (x_2 - x_1^2)^2 , \qquad (1)$$

где A – численный коэффициент. Обычно он принимается равным 100, но в данной работе он также принимался равным 10^3, 10^4, 10^5.

Для сравнения был взят метод вращения осей Розенброка. Трудоёмкость алгоритма оценивалась в Горнерах (1 Горнер – однократное вычисление целевого функционала). Будем считать, что алгоритм верно нашёл решение, если расстояние от вычисленной точки до точки минимума меньше, чем 10^{-4}. Расстояние между точками вычисляется по формуле

$$d = \sqrt{(x_1^* - x_1)^2 + (x_2^* - x_2)^2}, \qquad (2)$$

где x_1^*, x_2^* – точные координаты точки минимума, x_1, x_2 – вычисленные.

В среднем для поиска точки минимума функции Розенброка методу вращения осей требуется на порядок меньше Горнеров, чем алгоритму IWOA.

Чем хуже обусловленность задачи, тем больше ошибок допускает алгоритм IWOA. При максимальной допустимой трудоёмкости 1000500 Горнеров и A=10^4 IWOA нашёл верное решение в 99 случаях из 100, при A=10^5 – в 83 случаях из 100. При увеличении максимальной допустимой трудоёмкости до 2000500 Горнеров алгоритм IWOA нашёл верное решение в 99 случаях из 100.

ЛИТЕРАТУРА

1. IWOA: An improved whale optimization algorithm for optimization problems / Seyed Mostafa Bozorgi, Samaneh Yazdani // Journal of Computational Design and Engineering. – 2019.- 6. - 243–259.
2. (2016). A hybrid whale algorithm and pattern search technique for optimal power flow problem / Bentouati, B., Chaib, L., & Chettih, S. // 8th international conference on modelling, identification and control (ICMIC). - 2016. - 1048–1053
3. Self-adaptive differential evolution algorithm in constrained real-parameter optimization / Brest, J., Zumer, V., & Maucec, M. S. // IEEE int. conf. evol. comput. – 2006. - 215–222
4. Методы оптимизации / И.Г. Черноруцкий, Н.В. Воинов, Л.П. Котлярова. - СПб.: СПбПУ, 2020.- 167с.

УДК 004.42

Юсупова О. А.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ И АНАЛИЗ МЕТОДА ОПРЕДЕЛЕНИЯ ДОМИНИРУЮЩЕЙ ЦБ ДЛЯ УПРАВЛЕНИЯ ПОРТФЕЛЕМ ЦБ

Как показывает практика, в краткосрочном периоде рынок имеет стохастический характер [1]. Существующие методы прогноза зачастую не позволяют получить достаточно точный прогноз, что приводит к потере прибыли [1, 3, 4]. Под доминирующими понимаем ЦБ, тренд которых опережает тренды некоторой группы бумаг [2]. При этом достаточно спрогнозировать тренд на один день для получения приемлемой доходности.

Программная реализация метода позволяет визуализировать результаты и делать выводы относительно прогнозов роста или падения цен ЦБ и состоит из двух этапов.

На первом этапе строится матрица взаимных корреляций. Для этого сначала вычисляются взаимные корреляции r между бумагами, торгуемыми на рынке. Значение корреляции r_{ij} между двумя бумагами определяется при помощи корреляционного отношения Бравэ – Пирсона (1) по средневзвешенным ценам бумаг i и j за все предшествующие дате анализа дни.

$$\Delta r_{ij} = \frac{\sum_{k=1}^{n}(P_{ik}-\bar{P}_{ik})\cdot(P_{jk}-\bar{P}_{jk})}{\sqrt{\sum_{k=1}^{n}(P_{ik}-\bar{P}_{ik})^2\cdot\sum_{k=1}^{n}(P_{jk}-\bar{P}_{jk})^2}} \qquad (1)$$

Здесь P_{ik} и P_{jk} – цены бумаг i и j соответственно, по которым происходит усреднение, $\bar{P}_{ik}$ и $\bar{P}_{jk}$– средние значения цены бумаг за период времени n.

Именно на основе значений r_{ij} и заполняется матрица взаимных корреляций. Она квадратная, симметричная, с единицами по диагонали. На основе матрицы определяется доминирующая бумага, по динамике изменения цены которой будет осуществлено прогнозирование тренда цен остальных бумаг набора.

На втором этапе происходит визуализация полученных результатов. Строятся графики движения цен доминирующей и доминируемых ценных бумаг. Далее следует сделать вывод о покупке или продаже текущей ценной бумаги на основе движения доминирующей ценной бумаги. В качестве примера приведены графики движения цен на рисунке 1. Слева выведен график движения доминирующей бумаги, справа – доминируемой. В точке, соответствующей дате 1 июля, от доминирующей бумаги поступает прогноз на рост, соответственно, следует осуществлять покупку доминируемой акции (Газпромнефть на рисунке 1) в точке, соответствующей дате 4 июля, хотя цена доминируемой бумаги падает.

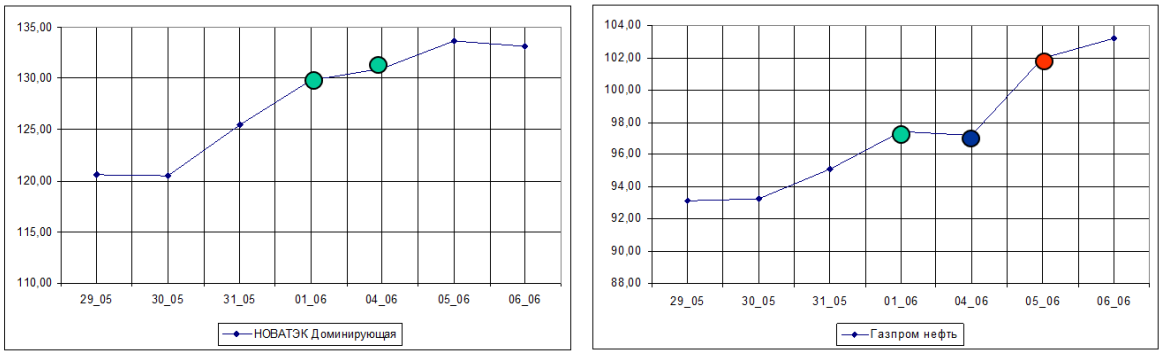


Рисунок 1 – Движение цен доминирующей акции НОВАТЭК и доминируемой акции Газпром нефть.

Видно, что на графике движения цен Газпромнефти в точке, соответствующей 5 июля, прогноз оправдался. Доход составил 4% от одной бумаги за один день инвестирования.

Экспериментальные исследования показали высокую практическую ценность метода, помогающего принимать верные решения. А в комплексе с анализом состояния отраслей, на основе фундаментального анализа отдельных компаний, показатели эффективности ещё возрастут.

ЛИТЕРАТУРА

1. Боди З., Кейн А., Маркус А. Принципы инвестиций. 4-е издание М.: Издательский дом «Вильямс», 2019. – 984 стр.
2. Лыпарь Ю.И. Теория системного структурного синтеза. Тезисы докладов на Международной научно-практической конференции «Системный анализ в технических и социальных задачах», 2001г., стр.43-45, СПбГТУ
3. Ценные бумаги: Учебник. Под ред. В.И. Колесникова и В.С.Торкановского. – М.: ФиС, 2002.
4. Хелферт Э. Техника финансового анализа. М: «ЮНИТИ», 2005.

Секция «Технологии обработки и анализа больших массивов данных на основе работ EPAM»

УДК 004.9

Елисеев П.В. (3 курс бакалавриата)
Петров О.Н., к.т.н., доцент

ИНТЕГРИРОВАННАЯ СИСТЕМА КОНТРОЛЯ СЕРВИСА

Проблемой, инициировавшей создание данного решения, стала ситуация, в которой в организации при переходе клиента от одного менеджера к другому невозможно было удобно и быстро просмотреть прежнюю историю переписки ВКонтакте с клиентом.

Было решено создать систему, которая бы позволяла сохранять переписку ВКонтакте менеджеров с клиентами, при этом делать её недоступной для удаления менеджерами, оказывать к ней изолированный доступ и отображать совмещённую историю общения среди всех менеджеров для каждого клиента.

Основополагающей идеей стало хранить переписку там же, во ВКонтакте, посредством пересланных сообщений. Сообщения, в составе пересланного, во ВКонтакте сохраняются даже после удаления их оригиналов.

Была создана специальная закрытая группа для менеджеров, которая послужила агрегатором сообщений из диалогов с клиентами. Система контроля сервиса стала разрабатываться в виде микросервиса на базе микросервисной архитектуры ERP, используемой в организации.

Реализация такой системы организована в виде работы процессов, распределённых по времени и достигающих следующих целей:

- сканирования диалогов менеджеров с помощью API VK;
- определения диалогов с клиентами на основе данных из CRM BlueSales;
- пересылки сообщений из клиентских диалогов блоками через API VK.

Программно это реализовано на PHP в виде методов API микросервиса, которые частями, порционно, выполняют работу процессов, описанных выше. Для выполнения задач был написан также отдельный алгоритм, отвечающий за распределение выполнения задач для равномерного обслуживания аккаунтов менеджеров. Для запуска этого алгоритма используется сервис CronJob вместо внутреннего Cron на сервере, так как внешний сервис предлагал удобный встроенный мониторинг выполнения запросов. Также, в ERP организации была интегрирована возможность авторизации во ВКонтакте через специальное приложение, чтобы иметь доступ к работе с аккаунтами менеджеров через API.

После успешного запуска фоновых процессов был добавлен интерфейс для просмотра сообщений из группы ВКонтакте с объединением переписки от всех менеджеров для каждого клиента. Также были добавлены сортировка и поиск по клиентам с возможностью выборки по датам существования переписки.

В ходе разработки, в основном, были обнаружены три проблемы: API CRM BlueSales не позволяло удобно осуществлять получение данных клиентов; иногда API ВКонтакте возвращало ошибки от самой простой, связанной с ограничением по количеству запросов в секунду, вплоть до «500 Internal Server Error», из-за чего некоторые сообщения были не

пересланы; не всегда можно было получить доступ к актуальной переписке из-за задержки в работе распределённых процессов.

Для решения первой проблемы с BlueSales была добавлена дополнительная задача, которая, работая с API BlueSales, создаёт копию базы данных с информацией клиентов и сохраняет её отдельно с помощью API микросервиса ClientManager. Это позволило обеспечить гораздо более быстрый и качественный доступ к клиентской информации.

Для решения второй задачи, чтобы заполнить пробелы в пересланных, были добавлены дополнительные процессы, которые проверяют качество пересылки сообщений, восполняют пробелы, а также сохраняют некоторую дополнительную метаинформацию, полезную и для статистики в том числе.

Третья проблема была решена путём добавления при просмотре сообщений режима агрегации сообщений напрямую из диалогов менеджеров. Это позволило избежать ситуации, в которой было невозможно ознакомиться с актуальной перепиской в течение рабочего дня.

Таким образом, создано коммерческое решение, которое позволяет: собирать переписку менеджеров с клиентами во ВКонтакте; сохранять эту переписку в полном начальном объёме; обеспечивать к ней актуальный доступ, объединяющий сообщения всех менеджеров для каждого клиента; осуществлять мониторинг работы менеджеров и более качественно и удобно решать возникающие конфликтные ситуации.

В процессе разработки были использованы следующие решения и технологии: VK API, BlueSales API, Meedoo, CronJob, MySQL, phpMyAdmin, Laravel, HTML, CSS, Bootstrap, JavaScript, JQuery, PHP OOP, Microservice App Architecture, JSON API, Ubuntu, SSH, SFTP, PhpStorm.

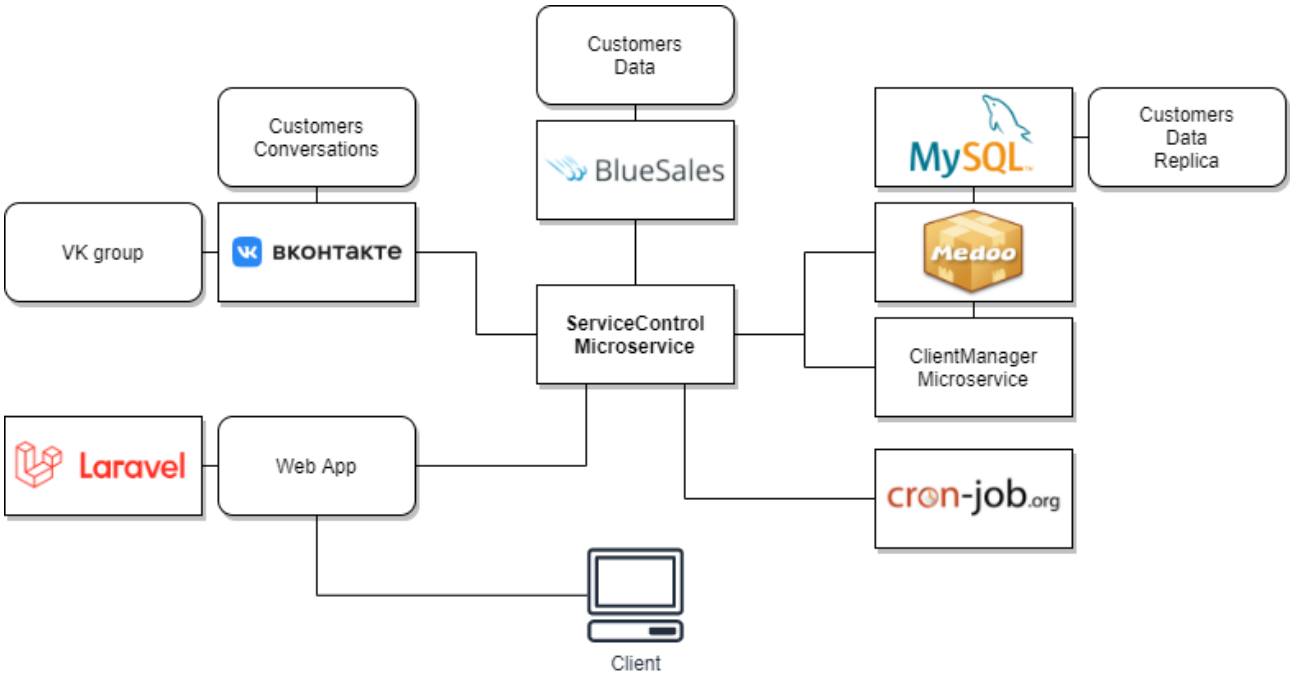


Рисунок 1 – Схема работы Интегрированной системы контроля сервиса

ЛИТЕРАТУРА

1. Ньюмен С. Создание микросервисов. – СПб.: Питер, 2016. – 304 с.: ил. – (Серия «Бестселлеры O'Reilly»)
2. ВКонтакте. Разработчикам [Электронный ресурс] – URL: https://vk.com/dev (дата обращения: 10.11.2020)
3. Laravel - The PHP Framework For Web Artisans [Электронный ресурс] – URL: https://laravel.com/docs/ (дата обращения: 10.11.2020)

УДК 004.738.5: 004.451.55

Златоустов В.А. (1 курс магистратуры)
Горавнева Т.С., к.т.н., доцент
Пшеничная К.В., к.т.н., доцент

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ФУНКЦИОНИРОВАНИЯ «УМНОЙ ЛАБОРАТОРИИ»

Целью работы является разработка Интернет-проекта для сбора данных по параметрам микроклимата и вредных производственных факторов в помещении, используя экосистему программирования микроконтроллеров Espruino на языке JavaScript, их наглядное представление в одностраничном динамическом WEB-приложении, а также предоставление рекомендаций по улучшению микроклимата в помещении для сохранения здоровья и работоспособности людей в «умной лаборатории».

Вся информация о параметрах микроклимата и вредных производственных факторах хранится в XML-базе данных, информация из которой выводится в приложение с помощью языка XSLT без необходимости перезагрузки всей страницы.

В состав базы данных входит следующая информация:

– Оптимальные и предельные значения микроклимата для помещения «умной лаборатории»;
– Оптимальные и предельные значения вредных производственных факторов для помещения «умной лаборатории»;
– Результаты измерений, полученные с датчиков микроконтроллеров, установленных в помещении «умной лаборатории».

Основные параметры микроклимата и вредных производственных факторов, которые были использованы при создании базы данных:

– температура воздуха (°C);
– температура поверхностей (°C);
– относительная влажность воздуха (%);
– шум (дБА);
– освещенность (лк).

Для создания и работы с XML-базой данных была использована интегрированная среда разработки (Integrated Development Environment) Microsoft Visual Studio. Эта IDE позволяет быстро и качественно создавать XML, XSD и XSL-документы, выполняет валидацию XML-документа на основе XML-схемы и указывает точное место ошибки, имеет автодополнение тегов, генерирует модель XSD-документа для наглядного представления структуры данных.

Для измерения параметров микроклимата и вредных производственных факторов были определены следующие датчики: цифровой датчик температуры и влажности AMP-B045, датчик освещённости AMP-B004, датчик шума AMP-B087.

После получения измерений датчиков микроконтроллеров Espruino для записи их в XML-файл был разработан JavaScript-скрипт.

Для представления и анализа характеристик, получаемых в «умной лаборатории», было выполнено наглядное их представление в одностраничном динамическом WEB-приложении.

Требования, предъявляемые к WEB-приложению:

– WEB-приложение должно предоставлять интерфейс для просмотра и фильтрации информации из базы данных;
– WEB-приложение должно работать в локальном режиме, т.е. без его загрузки в сеть Интернет.
– Обновление частей WEB-приложения, полученных с помощью XSLT-преобразований, должно выполняется без перезагрузки WEB-страницы.

При запуске WEB-приложения отображается WEB-страница, в верхней части которой располагается раздел заголовка, в котором находится навигационная панель с навигацией по

разделам WEB-страницы и вступительный текст. Навигация осуществляется по 4 основным разделам: домой, Интернет-вещей, интерфейс базы данных, умная лаборатория.

При открытии модального окна отображается информация из базы данных в виде таблицы (рис. 1). Для фильтрации выводимой информации нужно выбрать в выпадающем списке нужный критерий. Для сортировки столбца таблицы по возрастанию или убыванию нужно кликнуть по заголовку столбца таблицы необходимое количество раз для сортировки в разных направлениях.

Интерфейс базы данных

Все

№	Название	Единица измерения	Значение	Дата
1	Освещенность	лк	510	11.05.2020 11:00:07
2	Относительная влажность воздуха	%	70	11.05.2020 11:00:03
3	Температура поверхностей	°C	29.0	11.05.2020 11:00:02
4	Температура воздуха	°C	29.0	11.05.2020 11:00:00
5	Освещенность	лк	350	11.05.2020 10:50:07

Рисунок 1 – Фрагмент модального окна с таблицей

В модальном окне умной лаборатории отображается результаты измерений с датчиков от самых новых к более старым (рис. 2). В зависимости от полученных с датчиков значений указывается, находится ли параметр на оптимальном уровне, на допустимом уровне, превышает или не достает до допустимого уровня. Также здесь можно получить рекомендации по улучшению значений параметров, находящихся вне допустимого уровня.

Умная лаборатория

Освещенность	Относительная влажность воздуха	Температура поверхностей	Температура воздуха	Освещенность
Значение: 510лк	Значение: 70%	Значение: 29.0°C	Значение: 29.0°C	Значение: 350лк
11.05.2020 11:00:07	11.05.2020 11:00:03	11.05.2020 11:00:02	11.05.2020 11:00:00	11.05.2020 10:50:07
Значение выше допустимого Уменьшите уровень света	Значение в предалах допустимого	Значение в предалах допустимого	Значение выше допустимого Включите кондиционер	Значение оптимально

Рисунок 2 – Фрагмент модального окна с показаниями датчиков и рекомендациями

Средства создания разработки:

- Для WEB-приложения – HTML5, CSS3, JavaScript.
- Для отображения содержимого XML-базы данных – XSLT, JavaScript.
- Для работы с XML-базой – интегрированная среда разработки Microsoft Visual Studio.
- Для получения и обработки значений с датчиков – экосистема программирования микроконтроллеров Espruino на языке JavaScript.

ЛИТЕРАТУРА

1. Учебники и самоучители по HTML, CSS, XML, XSD, XPath, XSLT, jQuery, JavaScript. [URL]: https://msiter.ru/tutorials – (дата обращения: 18.03.2021).
2. Начало работы с Espruino: JavaScript в микроконтроллере [URL]: http://wiki.amperka.ru/js:start – (дата обращения: 18.03.2021).

УДК 004

Конкин В.В., Субцельная О.Д. (4 курс бакалавриата)
Дробинцев П.Д., к.т.н., доцент

РАЗРАБОТКА СИСТЕМЫ УМНОЙ ПАРКОВКИ

Целью работы является создание полноценной системы так называемой умной парковки на основе исследования существующих похожих систем, обладающих определёнными недостатками [1][2]:

– отсутствие визуальной индикации свободных мест;

– отсутствие возможности бронирования места;

– отсутствие визуального контроля движения автомобиля при осуществлении парковки;

– необходимость покидать автомобиль для оплаты парковки через терминал:

– необходимость регулярного осмотра оборудования.

Недостатки в большинстве случаев вызваны тем, что компании-производители систем пытаются лишь улучшить готовую технологию [2], которая была изобретена примерно в середине XX века [3], новыми программными средствами, а не стараются переосмыслить саму идею и исследовать потребности современных автомобилистов. Умная парковка как система, включающая в себя актуальные аппаратные и программные средства обработки и отображения информации, а также отвечающая нуждам автомобильных реалий, является достойным альтернативным решением. Таким образом можно выделить следующие задачи умной парковки для устранения вышеперечисленных недостатков:

– облегчение процедуры поиска свободного места на паркинге;

– обеспечение возможности бронирования желаемого места;

– обеспечение визуального контроля при осуществлении парковки автомобиля;

– автоматизация учёта свободных и занятых парковочных мест;

– автоматизация оплаты за время пользования парковкой (опционально);

– информирование о собственных неисправностях.

Демонстрация выполнения поставленных задач обеспечивается с помощью платформы разработки электронных устройств Arduino, базы для хранения данных Firebase и Android приложения. Платформа Arduino выбрана как наиболее доступный инструмент для создания функционирующего прототипа [4], отвечающего заявленным целям.

Пользователь с помощью приложения на портативном устройстве (смартфоне или планшете на базе Android) имеет возможности просмотра ближайших к нему парковок, бронирования места, просмотра карты для навигации, оплаты парковки и наблюдения за автомобилем для безопасной парковки. Для полноценного доступа ко всем функциям приложения требуется наличие интернет-соединения.

Контроль системы осуществляется путём постоянного обращения к базе данных Firebase, которая получает необходимые данные от датчиков и других компонентов аппаратной составляющей на базе Arduino. Связь с сервером осуществляется при помощи Wi-Fi модуля или Ethernet кабеля. Это сделано для предоставления актуальной информации, которая отображается в пользовательском приложении. Поиск свободного места упрощается с помощью индивидуального светового сигнала на парковочном месте, который может находиться в трёх состояниях: красный – место занято, жёлтый – место забронировано, зелёный – место свободно.

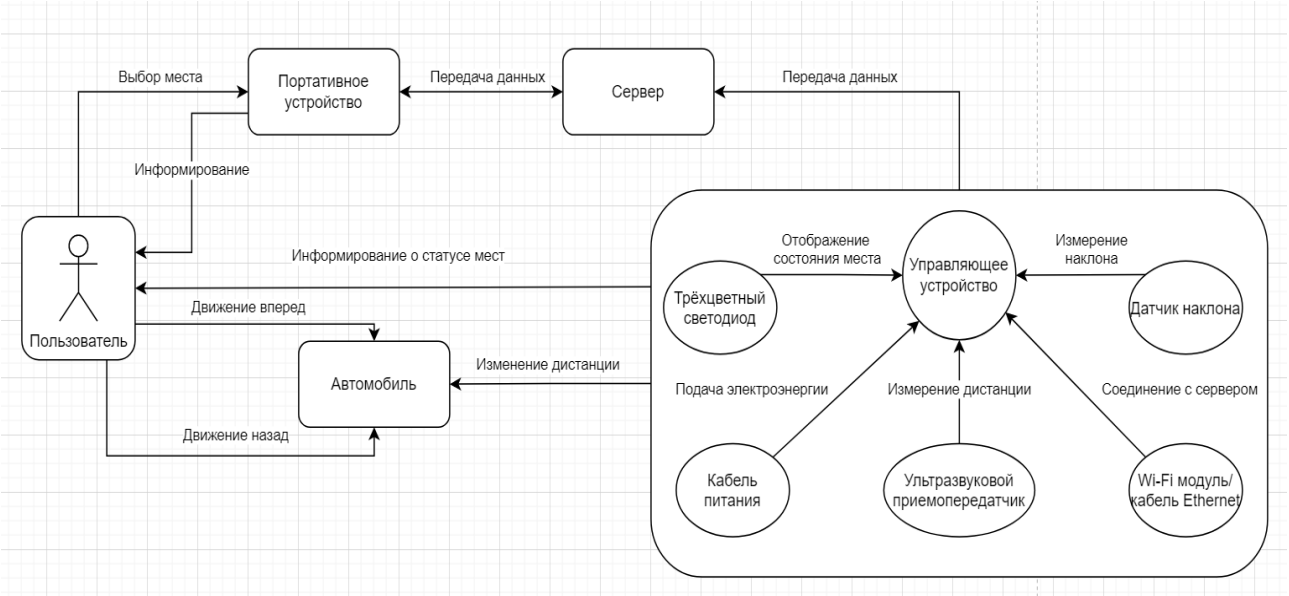


Рисунок 1 – Схема работы системы.

Для обеспечения информационной безопасности каждому новому пользователю необходимо зарегистрироваться в приложении, а затем пользоваться им уже со своего аккаунта. При переустановке приложения или переходе на новое устройство пользователь должен авторизоваться с помощью своего логина и пароля, таким образом все данные будут сохранены.

Описанный принцип работы системы разрешает недостатки существующих аналогичных автоматизированных комплексов контроля парковки. Более того, представленная система не требует больших экономических затрат [4], понятна и проста в создании и эксплуатации.

Демонстрация работы на данный момент создана в виде миниатюрного прототипа на базе Arduino с отображением состояния с помощью платформы ThingsBoard. В дальнейшем планируется разработка полноценного мобильного приложения, совершенствование аппаратной составляющей и добавление новой функциональности для достижения поставленных целей работы.

ЛИТЕРАТУРА

1. Информационный городской портал «Парковки Санкт-Петербурга»: официальный сайт. – Санкт-Петербург. – URL: https://parking.spb.ru/ru/ (дата обращения: 12.03.2021). – Текст: электронный.
2. Принцип работы автоматической парковки / Текст: электронный // ПаркСервисПлюс: официальный сайт компании. – URL: https://parkplus.ru/princip-raboty-parkovki (дата обращения: 12.03.2021).
3. Эволюция паркингов от дома Гауди (1910г.) до Porsche Design Tower (2017г.) / Текст: электронный // Engel & Volkers: официальный сайт компании. – URL: https://evspb.ru/istoriya-evolyucii-parkingov (дата обращения: 12.03.2021).
4. Преимущества использования Arduino / Текст: электронный // Робототехника и робототроение: информационный ресурс. – URL: https://robolive.ru/preimushhestva-ispolzovaniya-arduino (дата обращения: 12.03.2021).

УДК 004.042

Парахин К. А. (4 курс бакалавриата)
Никифоров И. В., к.т.н., доцент

РАЗРАБОТКА ГИБКОЙ И ВЫСОКОПРОИЗВОДИТЕЛЬНОЙ ПЛАТФОРМЫ ДЛЯ АНАЛИЗА МАШИННЫХ ДАННЫХ

В связи с ростом количества пользователей Интернета для компаний всё более актуальной становится задача обработки больших объемов данных [1]. Для решения задачи анализа данных существует целый ряд программных продуктов. Среди них такие продукты, как:

- Hue [2] – SQL ассистент с открытым исходным кодом для запросов к любым базам и хранилищам данных. Помимо отображения таблиц с данными, данное решение позволяет строить различные диаграммы.

- DataDog [3] – мониторинговый сервис, агрегирующий метрики и события с серверов, из баз данных, приложений, инструментов и сервисов, на основании которых формируется комплексное представление инфраструктуры.

- Graylog [4] – централизованное решение для управления журналами, построенное на открытых стандартах сбора, хранения и анализа терабайтов машинных данных в режиме реального времени. Аналогичных средств множество, но Graylog предлагает отличную производительность, удобную аналитику и красивый интерфейс.

- Splunk [5] – платформа для сбора, хранения, обработки и анализа машинных данных. Одной из главных особенностей платформы является то, что она может работать с данными практически из любых источников, и поэтому список возможных применений системы очень широк.

Splunk поддерживает Search Processing Language (SPL) – язык запросов для анализа данных [6]. С его помощью можно строить различные выборки и таблицы, сортировать, фильтровать, агрегировать, строить отчеты, создавать вычисляемые поля и многое другое. Также SPL поддерживает конвейерную обработку – каждая команда может принимать на вход результат выполнения другой команды.

Проведем сравнительный анализ среди свободно распространяемых лицензий ранее упомянутых программных средств по следующим основным критериям:

- Максимальный объем обрабатываемых данных. Ограничения есть только у Graylog – 5 Гб и у Splunk – 500 Мб.

- Возможность расширяемости языка поисковых запросов путем создания новых команд. Hue использует традиционный SQL, который не может быть расширен пользователем. DataDog и Graylog используют свои собственные языки запросов, которые также нельзя расширить. Касаемо Splunk, в нём есть возможность создавать пользовательские команды, используя SDK Python.

В результате, на основе сравнительного анализа видно, что все рассматриваемые системы обладают следующими недостатками:

- отсутствуют средств для расширения языка запросов и создания собственных команд;

- ограничения на максимальный объем обрабатываемых данных (с открытой лицензией).

Поэтому целью работы является разработка прототипа свободно распространяемой аналитической платформы, отличительной особенностью которой является расширяемость языка запросов и поддержка объема данных 1 Тб.

Рассмотрим архитектуру разрабатываемой платформы на рисунке 1.

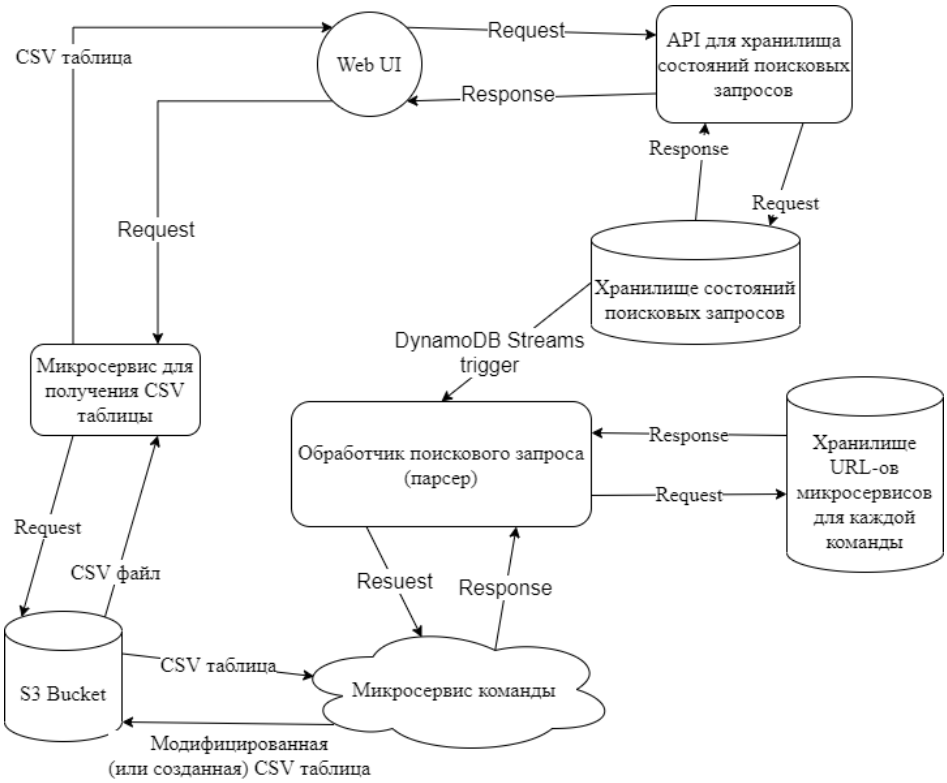


Рисунок 1 – Архитектура платформы.

Данная архитектура позволяет:

- расширить язык запросов путем создания новых команд;
- обрабатывать большие объемы данных.

Стек технологий:

- Python 3;
- Vue.js;
- AWS: DynamoDB, Lambda, API Gateway.

На данном этапе система реализована на уровне работающего прототипа, с применением средств платформы AWS. В дальнейшем планируется провести тестирование производительности прототипа на нескольких размерах данных.

ЛИТЕРАТУРА

1. Ивлев В.А., Никифоров И.В., Леонтьева Т.В. Обработка данных в геоинформационных системах для выбора местоположения рекламы. Современные технологии в теории и практике программирования: сборник материалов конференции. 2019. С. 27-30.
2. Hue [Электронный ресурс]. URL: https://gethue.com/ (дата обращения: 11.03.2021).
3. DataDog [Электронный ресурс]. URL: https://www.datadoghq.com/product/ (дата обращения: 11.03.2021).
4. Graylog [Электронный ресурс]. URL: https://www.graylog.org/ (дата обращения: 11.03.2021).
5. Why Splunk? [Электронный ресурс]. - Splunk Inc. - URL: https://www.splunk.com/en_us/about-us/why-splunk.html (дата обращения: 11.03.2021).
6. Парахин К. А., Никифоров И.В., Степанов С.Н., Подход к разработке команд для Search Processing Language на базе платформы Splunk, BI-технологии и корпоративные информационные системы в оптимизации бизнес-процессов цифровой экономики, Издательство Уральский государственный экономический университет (Екатеринбург), 2020 (в печати).

УДК 004.05

Романов К.М. (4 курс бакалавриата)
Никифоров И.В., к.т.н., доцент
Леонтьева Т. В., к.т.н., доцент

ПРОГРАММНЫЙ СЕРВИС АВТОМАТИЧЕСКОГО ОПРЕДЕЛЕНИЯ ОШИБОК В ПОЛЬЗОВАТЕЛЬСКИХ СЕССИЯХ КРОССПЛАТФОРМЕННОГО ПРИЛОЖЕНИЯ

На сегодняшний день количество пользователей сети Интернет растет постоянно [1], и, как следствие, наблюдается рост числа интернет-сервисов, предоставляющих различные услуги [2]. Для удержания аудитории и привлечения новой, таким компаниям требуется поддержание высокого стандарта качества предоставляемого пользователям продукта. Один из факторов, непосредственно влияющий на пользовательский опыт при взаимодействии с приложением – встречающиеся ошибки при работе с приложением. Своевременное нахождение и устранение сбоев в работе приложении является важной задачей, для выполнения которой существуют несколько подходов. В данной статье мы остановимся на способе анализа данных, получаемых в результате обработки пользовательских сессий.

Существуют несколько разных сценариев возникновения ошибок. Некоторые сбои могут быть единичными, другие – иметь массовый характер. Локализация массовых ошибок у пользователей является трудоемкой задачей т.к. требует анализа большого числа данных. Нахождение же малочисленных проблем и выявление возможных причин является еще более сложным и длительным процессом.

Для уменьшения затрачиваемого времени на мониторинг состояния сервиса и ошибок, возникающих в процессе его работы, актуальной является задача автоматизации сбора и анализа действий пользователя. В ходе решения данной проблемы необходимо учесть несколько факторов: решение должно быть устойчивым к высоким нагрузкам, иметь функционал для подсчета определенных критериев из получаемых сессий и вывода результатов на интерактивную географическую карту.

В работе для решения указанной задачи предлагается реализовать сервис, мониторящий качество обслуживания (QOS) [3]. Основная цель подобных программ состоит в том, чтобы подсчитать и предоставить метрики, необходимые для оценки работы кроссплатформенного приложения.

На рынке существует ряд решений, которые могут подойти под требования данной задачи. Сначала были изучены продукты, которые непосредственно выявляют ошибки и сбои в работе приложений. В список таких сервисов попали решения от компаний VisitorLab, SessionCam, Instabug. Однако, при детальном рассмотрении выбранных платформ появилась проблема с поддержкой кроссплатформенности. Все предоставляемые решения являются направленными на определенный сегмент (например, веб-сайты или мобильные приложения) и будут требовать либо совмещения, либо доработки, если такая будет возможна.

Другим вариантом было создание системы на основе ELK [4] или Grafana и Prometheus [5]. Данные платформы несколько отличаются по назначению, но предполагалось, что они подходят под сценарии использования. Основная цель применения ELK – сбор данных, их анализ и визуализация. Возможности связки Grafana и Prometheus позволяют осуществлять получение и мониторинг метрик, а также уведомлять при превышении значениями некоторого заданного порога.

В компании “ОККО”, где требовалось создание данного сервиса, уже существовала архитектура, которая использовалась предыдущей версией QOS. Предыдущее решение было написано с использованием vanilla python 2.7 и имело ряд проблем, связанных с производительностью. Инструменты же типа ELK или Grafana требуют внесения некоторых изменений в систему логов и сбора сессий, что было нежелательно. По этим причинам предлагается новая реализация собственного QOS сервиса, удовлетворяющего потребностям компании и не требующего изменений существующей архитектуры.

На Рисунке 1 представлены шаги обработки пользовательских сессий, которые были выбраны для автоматизации создаваемым продуктом. Работающие клиенты приложений на протяжении всей пользовательской сессии присылают промежуточные данные о состоянии приложения, сети, возникших ошибках, действиях пользователя. Эти данные требуется принять, после чего выполнить сортировку, сопоставив с информацией, полученной ранее и вывести итоговый результат.

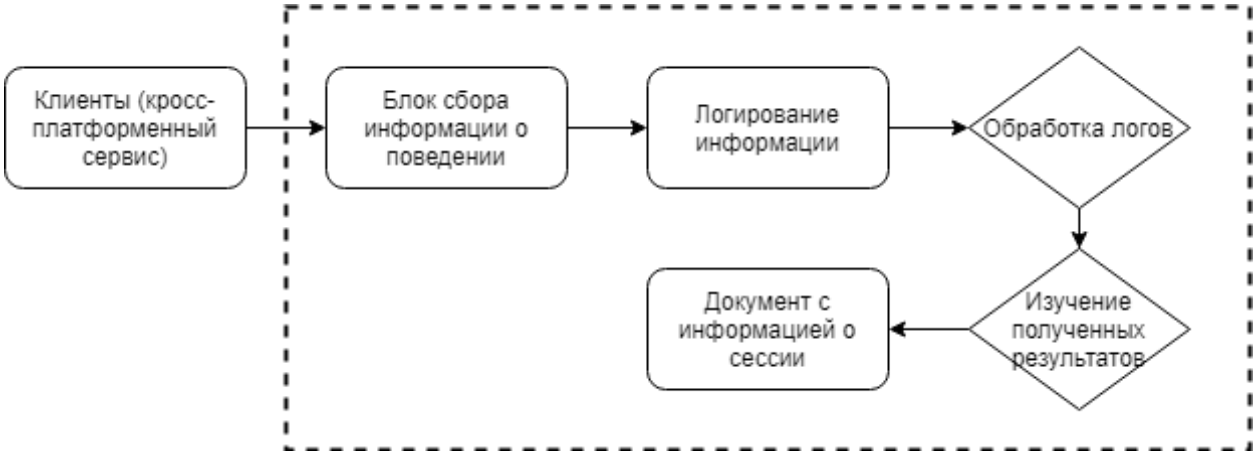


Рисунок 1 – Подход автоматизации определения ошибок.

На Рисунке 2 показана разработанная схема обработки данных, начиная с клиента и заканчивая таблицами с подсчитанными результатами. В компании “ОККО” уже существовала связка Collector и Kafka. Все данные, отправляемые клиентами, изначально приходят в сервис-сборщик, которые осуществляет их конвертацию для дальнейшей передачи брокеру сообщений. В Kafka все сессии могут быть взяты создаваемым сервисом и преобразованы в требуемые форматы вывода.

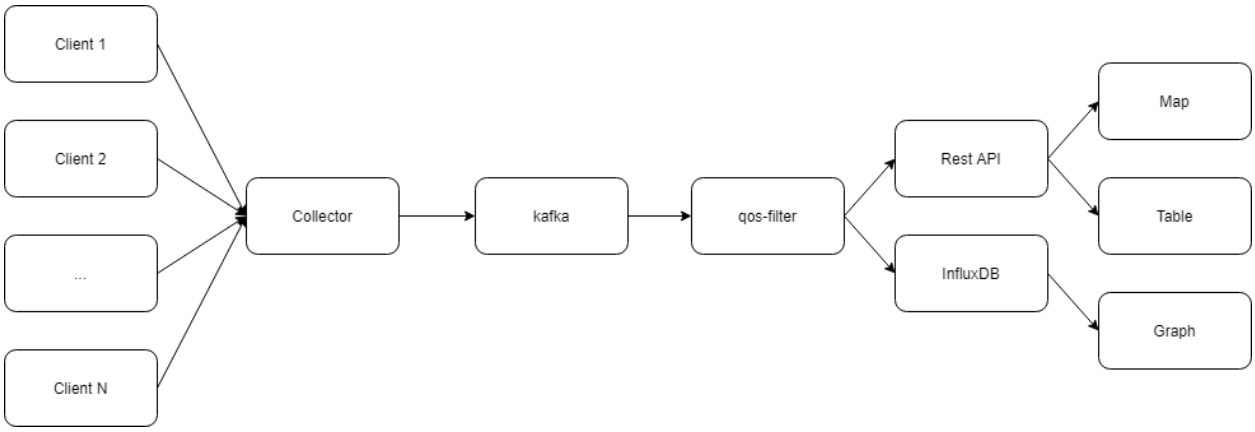


Рисунок 2 – Схема обработки данных.

Перед разработкой приложения был сформирован стек предполагаемых технологий, которые будут использованы в проекте. Языком программирования для BackEnd-составляющей сервиса стал GoLang [6]. Сформированный исполняемый файл запускается в среде для контейнеризации Docker. В процессе разработки использовались системы Jira и Bitbucket для формирования задач и осуществления контроля версий проекта.

В результате разработки приложения была создана серверная часть, предоставляющая REST API и записывающая сортированные данные в базу InfluxDB. Для вывода результатов работы сервиса был разработан графический интерфейс с использованием HTML, CSS и JavaScript. На Рисунке 3 представлено несколько возможных выводов данных, рассчитанных QOS.

Используя данный интерфейс, можно получить информацию об общем числе сессий с классификацией по серьезности и количеству ошибок, их примерное местонахождение, оценить нагрузку системы и само значение качества сервиса.

В ходе работы был проведен сравнительный анализ существующих решений для обнаружения и сбора ошибок в приложениях на рынке, которые показал сложность внедрения готовых реализаций в уже существующую инфраструктуру. Для разработки приложения был определен стек современных технологий, которые были использованы при выполнении задачи. Полученный прототип приложения доказал работоспособность предлагаемого решения для автоматизации определения ошибок в пользовательских сессиях.

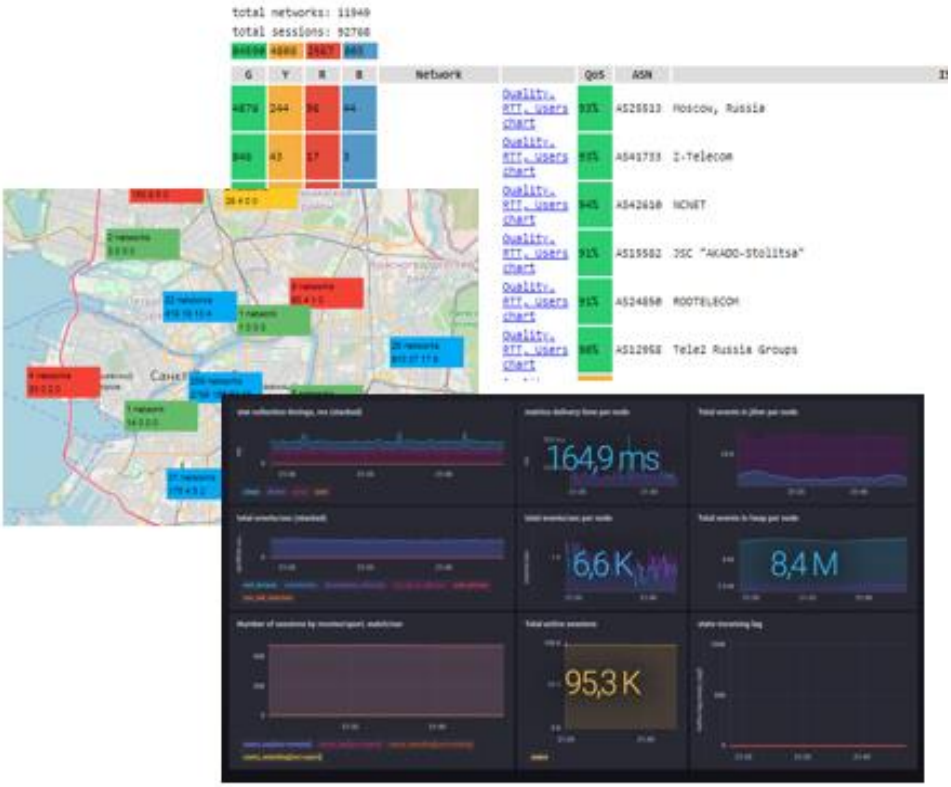


Рисунок 3 – Возможные интерфейсы вывода сервиса.

ЛИТЕРАТУРА

1. Тимощук Е.В., Никифоров И.В., Дробинцев Д.Ф., Анализ данных профилей сотрудников в социальной сети linkedin, В сборнике: Современные технологии в теории и практике программирования. сборник материалов конференции. Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. Санкт-Петербург, 2020. С. 133-135.
2. Сычев В.К., Ленькова Ю.В., Цветкова Е.А., Никифоров И.В., Анализ данных трендов YouTube, В сборнике: Современные технологии в теории и практике программирования. сборник материалов конференции. Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. Санкт-Петербург, 2020. С. 131-132.
3. Деарт В.Ю., Кожухов И.С., Исследование параметров качества обслуживания (QOS), определяющих качество восприятия пользователем (QOE) потокового видео при передаче через Интернет. – T-Comm: Телекоммуникации и транспорт, 2013. – т.7 №7 с. 28.
4. Петров В.В. Сбор, фильтрация и анализ больших данных с помощью стека ELK. – Colloquium-journal, 2019 – № 5-1 (29) с. 46-47.
5. Уруков С.Д. Внедрение открытой платформы “Grafana” для аналитики time-series данных в информационную систему автоматизированных зданий. – Молодежная школа-семинар по проблемам управления в технических системах имени А.А. Вавилова, 2018 – т. 1. С. 50-53.
6. GoLang documentation [электронный ресурс]. URL: https://golang.org/doc/ (дата обращения 08.03.2021)

УДК 004.457

Сафронов Д., Соломонов М.С., Стоноженко К.М., (1 курс магистратуры)
Никифоров И.В., к.т.н., доцент

РАСШИРЕНИЕ ФУНКЦИОНАЛА ЭЛЕКТРОННОЙ БИБЛИОТЕКИ СПБПУ ЗА СЧЕТ ВНЕДРЕНИЯ МОДУЛЯ АНАЛИЗА ДАННЫХ

Электронная библиотека СПбПУ содержит комплекс учебных материалов, таких как научные статьи, выпускные квалификационные работы, периодические издания и т.д. Библиотека ориентирована на предоставление учебных материалов пользователям в процессе их обучения и научной деятельности [1]. На текущий момент электронная библиотека предоставляет только поисковой механизм по различным атрибутам и не предоставляет возможности решать более сложные аналитические задачи [2]. К таким задачам, например, относятся извлечение дополнительной семантической информации из текста работ, классификация и кластеризация, восполнение недостающей информации [3].

Так как потребность пользователей и администраторов библиотеки в получении более полной и подробной информации о научных материалах растет, задача внедрения функционала анализа данных является актуальной.

Целью работы является уменьшение трудоемкости и времени анализа научных материалов библиотеки СПБПУ за счет реализации дополнительного модуля аналитики данных. Данная цель достигается за счет решения следующих задач:

1. Реализация модуля считывания данных из электронной библиотеки СПБПУ.
2. Разработка модуля обработки и анализа данных.
3. Разработка графического интерфейса.
4. Интеграция сервиса в существующую инфраструктуру электронной библиотеки.

В ходе работы была разработана следующая архитектура программного модуля, представленная на рисунке 1.

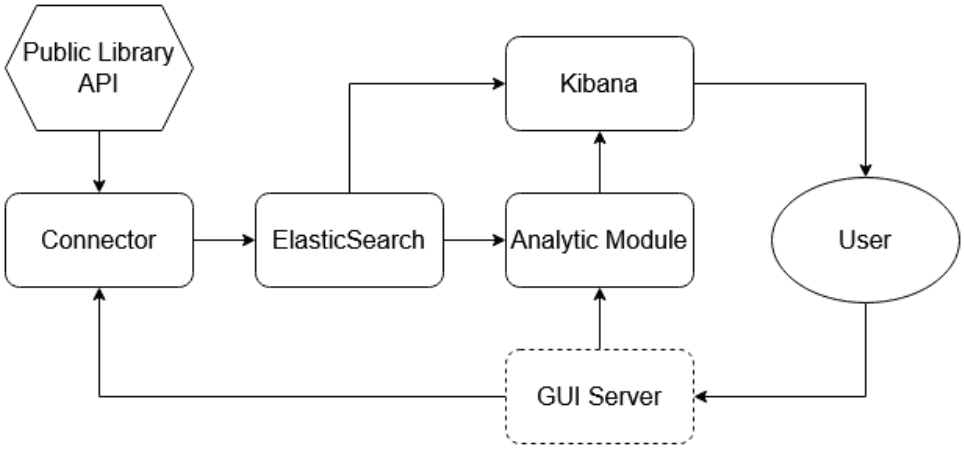


Рисунок 1 – Архитектура системы анализа данных электронной библиотеки.

Электронная библиотека использует автоматизированную библиотечную информационную систему (АБИС) «Руслан» [4]. АБИС используются для упрощения работы с библиотечным фондом и управления различными библиотечными ресурсами [5]. «Руслан» предоставляет публичный API, с помощью которого аналитическая система получает данные о научных работах [6]. Пример запроса, позволяющего получить единичную запись из базы данных АБИС, продемонстрирован на рисунке 2.

```
GET https://ruslan.library.spbstu.ru/rrs-web/db/EBOOKS/RU%5CSPSTU%5Cedoc%5C20151?recordSchema=gost-7.0.100
```

Рисунок 2 – Запрос к публичному API АБИС.

В частности, API можно использовать для выгрузки выпускных квалификационных работ студентов. Для сбора таких данных был написан сервис «Connector» на языке программирования Golang версии 1.14. Он обращается по HTTP к АБИС и загружает информацию в формате JSON и тексты в формате PDF в локальную файловую систему и в ElasticSearch, который используется для поиска, хранения и анализа данных [7].

Модуль аналитики данных написан на Python 3. На текущий момент он обрабатывает полученные тексты в формате PDF с использованием библиотеки nltk, выделяет ключевые слова и вычисляет частоты их встречаемости в тексте.

Для визуализации результатов используется инструмент Kibana [8]. Он дает возможность представлять различные статистические зависимости между метаданными JSON-объектов в графическом виде. Для автоматизации построения таких графиков в Kibana был написан скрипт на Python 3, который загружает созданные заранее шаблоны с использованием REST API. Пример результирующего графика представлен на рисунке 3.

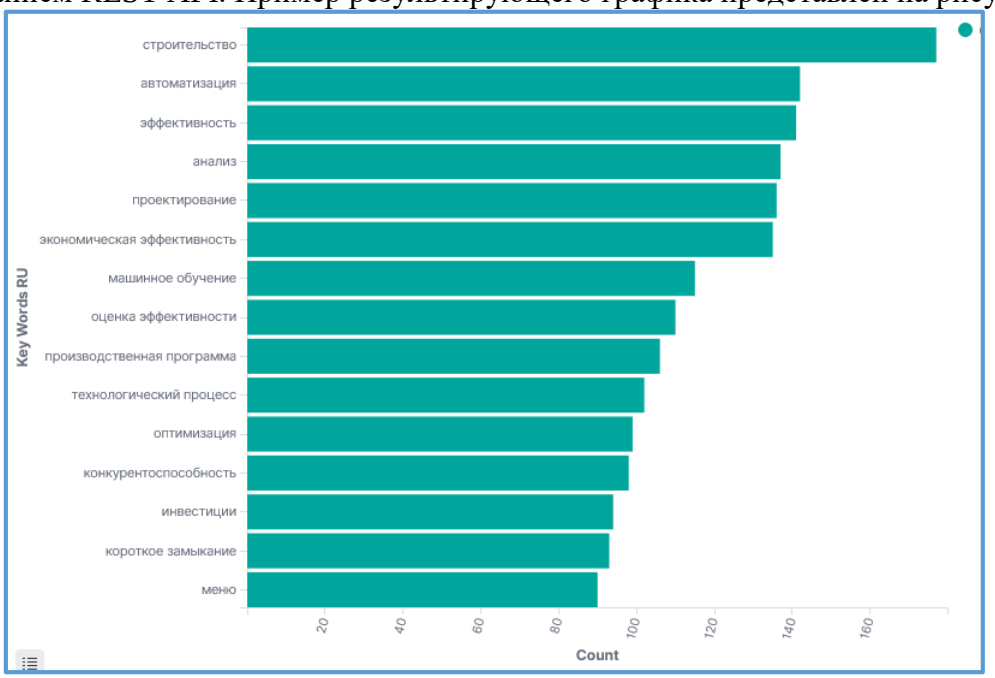


Рисунок 3 – Распределение частоты появления ключевых слов.

Таким образом была реализована система, которая предоставляет возможность локально загрузить, развернуть, обработать и визуализировать данные, полученные из электронной библиотеки [9]. В дальнейшем планируется увеличить количество решаемых аналитических задач, разработать собственный графический интерфейс и интегрировать модуль в электронную библиотеку.

ЛИТЕРАТУРА

1. Соколова Н.В., Достовалов С.С., БИБЛИОТЕЧНЫЙ ПОРТАЛ - ЕДИНОЕ ОКНО ДЛЯ РАБОТЫ С ИНФОРМАЦИОННЫМИ РЕСУРСАМИ, В сборнике: Корпоративные библиотечные системы: технологии и инновации. Материалы международной научно-практической конференции. 2016. С. 4-17.

2. Никифоров И.В., Курсовое проектирование по учебной дисциплине "Наука о данных и аналитика больших объемов информации", Учебное пособие / Санкт-Петербург, 2017.

3. Донцов А.Д., Селин И.А., Анализ медицинских данных с помощью методов машинного обучения, Современные технологии в теории и практике программирования. сборник материалов конференции. Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. Санкт-Петербург, 2020. С. 114-116.

4. Достовалов С.С., Кедрин А.В., Племнек А.И., Соколова Н.В., Усманов Р.Т., АВТОМАТИЗИРОВАННАЯ БИБЛИОТЕЧНО-ИНФОРМАЦИОННАЯ СИСТЕМА "РУСЛАН-НЕО", Свидетельство о регистрации программы для ЭВМ RU 2016611064, 26.01.2016. Заявка № 2015661861 от 04.12.2015.

5. Соколова Н.В., ВОПРОСЫ МОДЕРНИЗАЦИИ СИСТЕМЫ ИНФОРМАЦИОННОГО ОБСЛУЖИВАНИЯ БИБЛИОТЕК ПРИ РАБОТЕ С РЕСУРСАМИ ГЛОБАЛЬНОЙ СЕТИ, Информационное обслуживание в век электронных коммуникаций. Сборник материалов. Комитет по культуре Санкт-Петербурга; Центральная городская публичная библиотека им. В. В. Маяковского. 2016. С. 57-60.
6. Соколова Н.В., Усманов Р.Т., Абрамов А.М., СИСТЕМА УПРАВЛЕНИЯ БИЗНЕС- ПРОЦЕССАМИ КАК ПЛАТФОРМА ДЛЯ СОВМЕСТНОЙ РАБОТЫ БИБЛИОТЕЧНЫХ СПЕЦИАЛИСТОВ, Корпоративные библиотечные системы: технологии и инновации. Материалы международной научно-практической конференции. 2016. С. 96-103
7. Документация к ElasticSearch [Электронный ресурс]. Режим доступа: https://www.elastic.co/guide/index.html
8. Документация к Kibana [Электронный ресурс]. Режим доступа: https://www.elastic.co/webinars/getting-started-kibana
9. Исходный код проекта [Электронный ресурс]. Режим доступа: https://github.com/safronovD/spbstu-smart-library

УДК 004.42

Шевцов Д. Ю. (2 курс магистратуры)
Никифоров И. В., к.т.н., доцент

МЕТОД АВТОМАТИЗАЦИИ ПРОЦЕССА ИЗУЧЕНИЯ ИНФОРМАЦИИ ЗА СЧЕТ ПРИМЕНЕНИЯ ПОДХОДА ИНКРЕМЕНТАЛЬНОГО ЧТЕНИЯ

В современном мире объём доступной человеку информации стремительно растёт, и всё больше требуются эффективные способы её потребления и изучения [1, 2]. В некоторых областях, например, в изучении иностранных языков, обрёл популярность метод интервальных повторений. Но этот метод обладает существенным недостатком, он эффективен только при изучении свободной от контекста информации [3]. Обычно, когда мы что-то изучаем, почти каждое новое определение основываются на неких более фундаментальных принципах.

Учитывая сильные и слабые стороны метода интервальных повторений, был разработан метод инкрементального чтения. Он позволяет структурировать информацию и привести её в подходящий вид для изучения методом интервальных повторений [4].

Целью данной работы является применение метода инкрементального чтения в процессе автоматизации изучения информации.

Кривая забывания и интервальные повторения

Одной из основных моделей человеческой памяти является экспоненциальная кривая забывания. Вероятность вспомнить представляется в виде экспоненциально убывающей функции от времени, прошедшего с последнего повторения [5].

Метод интервальных повторений основан на явлении, называемом интервальный эффект. Человек лучше запоминает информацию, если она повторяется за отдалённые во времени сеансы повторения. Причём, после каждого повторения, вероятность вспомнить информацию убывает медленней [6].

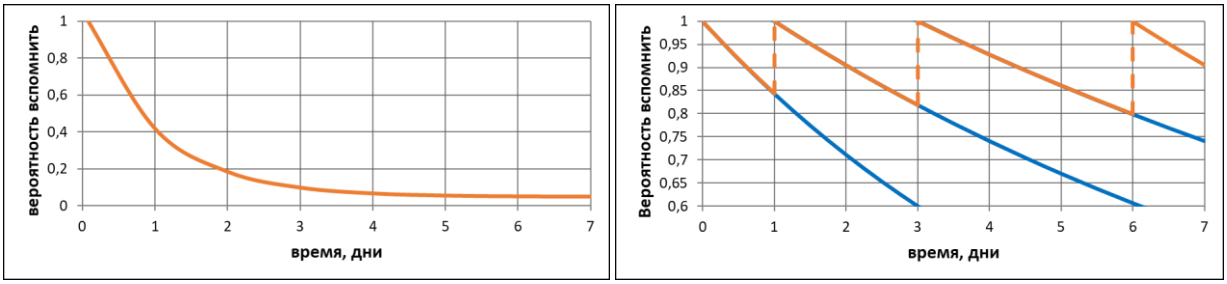


Рисунок 1 – Кривая забывания и интервальный эффект.

При использовании метода интервальных повторений информация представляется как набор физических или виртуальных карточек вопрос-ответ, пользователь даёт оценку своему ответу от нуля до пяти, на основе этих оценок рассчитывается интервал до следующего повторения.

Расчёт количества дней до следующего повторения $f(X)$ в зависимости от оценок ответов пользователя $X = (x_1, \dots, x_n)$ осуществляется по следующей формуле:

$$f(X) = a\left(\sum_{i=0}^{n}(b + cx_i + dx_i^2)\right)^{\theta r - 1},$$

где r – число верных ответов подряд;

$a = 6.0$, $b = -0.8$, $c = 0.28$, $d = 0.02$ и $\theta = 0.2$ – эталонные коэффициенты метода интервальных повторений [6].

Но этот метод оказывается эффективным только на информации, не требующей контекста, как например слово и его перевод на иностранный язык [3].

Инкрементальное чтение

Для изучения контекстно-зависимой информации был разработан метод инкрементального чтения. Информация повторяется через некоторые интервалы времени. Пользователь разделяет её на отрывки, например, первый отрывок – определение, которое ввели и дальше будут использовать. При этом отрывки имеют порядок, следующие не будут планироваться раньше повторения предыдущих.

Отрывки, в свою очередь, дробятся в более подробные части, пока информация не станет пригодной для создания карточки вопрос-ответ, которые уже подходят для метода интервальных повторений. Этот процесс растянут во времени для разных единиц информации, то есть, интервальный эффект задействуется не только при изучении готовых атомарных карточек, но и при работе над статьёй.

Реализация

На данный момент в прототипе реализовано добавление статей, их хранение и разделение на отрывки. В дальнейшем планируется реализовать превращение отрывка в карточку и планирование их повторения с помощью метода интервальных повторений.

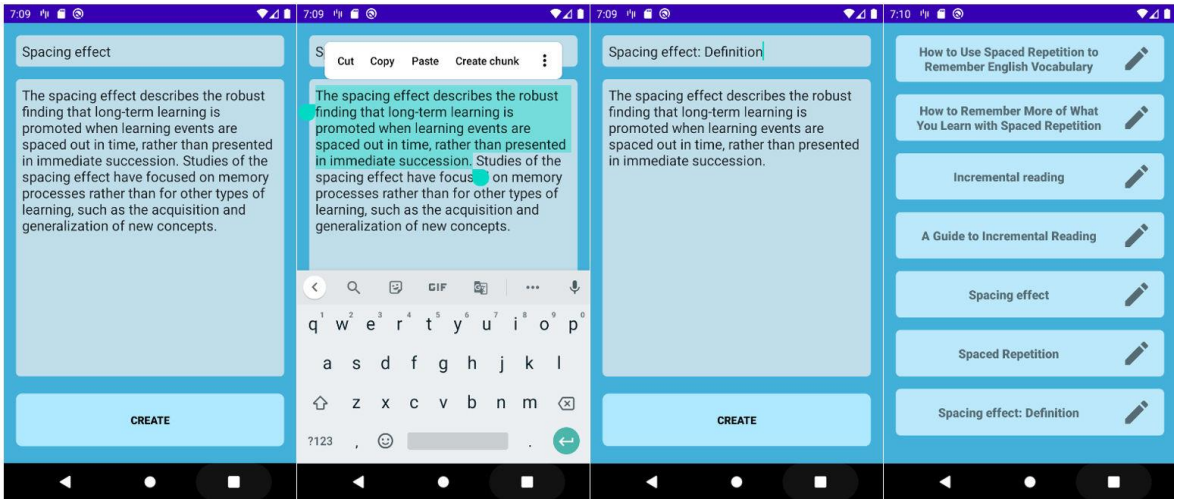


Рисунок 2 – Прототип приложения инкрементального чтения.

Приложение разрабатывается для платформы Android на языке Kotlin. При построении архитектуры приложения используются MVVM и Clean Architecture. Для асинхронного взаимодействия используются Kotlin Coroutines и Kotlin Flow. В качестве базы данных используется SQLite, работа с ней осуществляется через Room. При создании модульных тестов используются Junit5 и Mockk, для e2e тестов - AndroidJunit4 и Kakao.

ЛИТЕРАТУРА

1. Никифоров И.В., Курсовое проектирование по учебной дисциплине "Наука о данных и аналитика больших объемов информации", Учебное пособие / Санкт-Петербург, 2017.

2. Птицына Л.К., Птицын Н.А., Расширение знаний о раннем обнаружении появляющихся изменений, Наука. Информатизация. Технологии. Образование. Материалы XIII международной научно-практической конференции. 2020. С. 248-252.
3. T. Nakata и I. Elgort, «Effects of spacing on contextual vocabulary learning: Spacing facilitates the acquisition of explicit, but not tacit, vocabulary knowledge», Second Language Research, June 2020.
4. SuperMemo // History of incremental learning. https://super-memory.com/help/il.htm
5. H. Ebbinghaus, «Memory: A contribution to experimental psychology», New York City: Teachers College, Columbia University, 1913.
6. SuperMemo // Principles of spaced repetition. https://www.supermemo.com/en/archives1990-2015/english/princip
7. SuperMemo // Application of a computer to improve the results obtained in working with the SuperMemo method. https://www.supermemo.com/en/archives1990-2015/english/ol/sm2

УДК 004.62

Эйзенах Д.С., Рамейков А.С. (2 курс магистратуры)
Никифоров И.В., к.н.т., доцент

ВЫСОКОПРОИЗВОДИТЕЛЬНАЯ СИСТЕМА ИЗВЛЕЧЕНИЯ ДАННЫХ С ВЕБ-РЕСУРСОВ

В современном цифровом мире данные приобретают все большую и большую ценность. Жаловаться на недостаток информации не приходится, ее объем неуклонно растет и о по оценкам компании “SEAGATE” к 2025 году составит 163 зеттабайта (10^{21} байт) [1]. Тем не менее, для эффективной работы с таким объемом информации необходим эффективный способ её извлечения, обработки и структурирования. Ручной сбор данных для дальнейшего анализа может занять много времени, а также привести различным ошибкам (дублированию, опечаткам в тексте и т.д.). Автоматический сбор данных с помощью API не всегда возможен, так как владелец ресурса может закрыть к нему доступ или ограничит скорость/объем скачиваемой информации. Веб-скрапер (англ., «web-scraping») – это технология, которая позволяет, имитируя поведение человека на веб-сайте, автоматически извлекать и преобразовывать данные без использования API. В данной работе поставлена задача повысить эффективность работы веб-скрапера на основе фреймворка Scrapy [2], посредствам организации распределённой системы вычислений с возможностью горизонтального масштабирования, а также предотвращения перегрузки системы.

На рисунке 1 представлен предложенный подход к реализации распределенной системы скрапинга данных.

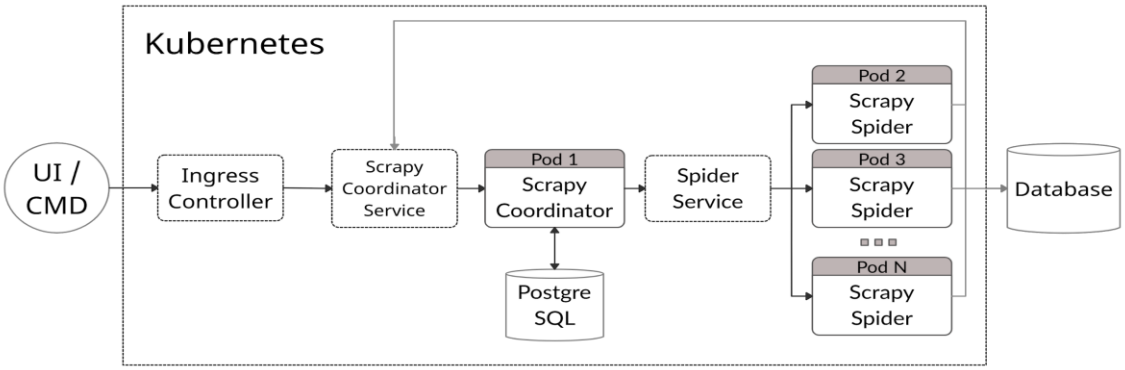


Рисунок 1 – Дизайн системы.

Система оркестрации: Система оркестрации — это система автоматического размещения, балансировки нагрузки и управления контейнерами приложений. Сегодня наиболее популярными инструментами являются Kubernetes [3], Docker Swarm, Apache Mesos.

На основании статьи [4] можно сделать вывод, что система оркестрации Kubernetes имеет преимущества в поддерживаемых технологиях.

Также Kubernetes поддерживает использование нескольких пространств имен, не имеющих прямого доступа к ресурсам друг друга. Это позволяет создать многопользовательскую среду, с возможностью выделения квот на процессорное время и объем оперативной памяти исходя из потребностей пользователя.

Ingress Controller: Контроллер входящего трафика - это элемент инфраструктуры оркестратора, основной задачей которого является проксирование внешнего трафика службам внутри кластера. Он также выполняет другие задачи, такие как: завершение SSL, балансировка [5] и маршрутизация трафика на основе имен и URL-адресов. Как видно из рисунка, входной контроллер является точкой входа в кластер для конечного пользователя и на основе имен хостов перенаправляет запросы в определенную среду.

Services: В нашей системе есть три сервиса: Scrapy Coordinator Service, Spider Service, Redis Service. Поды (приложения) внутри не являются постоянными и IP-адреса могут меняться, это может создать сложности для обеспечения доступа к каждому из них. Сервисы позволяют конечному пользователю или программе без IP-адреса конкретного пода, обращаться к нему по имени домена (например, crawler.company1, где поисковый робот - это общее имя модуля, а company1 - это имя пространства имен). Также сервис распределяет трафик с использованием метода Round robin, что позволяет предотвратить перегрузку отдельных узлов системы.

Scrapy Coordinator: Координатор используется для хранения очереди из входящих ссылок, генерации заданий для пауков, а также хранения обработанных URL-адресов. Сохранение отчищенных ссылок позволяет исключить повторные обходы. В качестве хранилища координатор использует базу данных PostgresSQL, которая позволяет полностью или частично перезапустить задание в случае возникновения внештатной ситуации в системе.

Scrapy Spider: Это автономное программное обеспечение, реализованное с использование фреймворка Scrapy. Оно поддерживает расширенный набор возможностей как по поиску нужной информации, так и по сбору данных с конкретных источников [6].

Тестирование системы проводилось на трех виртуальных машинах, имеющих следующие характеристики: 8 ядер и 16 Гб оперативной памяти. Производительность системы определяется по количеству обработанных уникальный URL-адресов, а также количеству собранных элементов страницы. Тестирование проводилось на протяжение 60 минут.

Таблица 1 – Результаты теста производительности.

	Single Node	**Two Node**	**Three Node**
Страниц за 60 мин.	124000	174000	196000
Элементов за 60 мин.	1650000	2380000	257000

При анализе работы приложенный подход показал свою жизнеспособность. При подключении дополнительных рабочих узлов, система демонстрирует лучшую производительность, а за счет использования системы оркестрации Kubernetes, появилась возможность автоматически отслеживать состояние каждого узла, автоматически перезапускать приложения, а также эффективно распределять нагрузку на систему.

ЛИТЕРАТУРА

1. «The Digitization of the World From Edge to Core,» November 2018. [В Интернете]. Available: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf.

2. Deepak Kumar Mahto, Lisha Singh, «A dive into Web Scraper world,» в *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, 2016.

3. Shemyakinskaya A.S., Nikiforov I.V., «Hard drives monitoring automation approach for Kubernetes container orchestration system,» *Proceedings of the Institute for System Programming of the RAS,* т. 32, № 2, pp. 99-106, 2020.

4. Isam Mashhour Al Jawarneh, Paolo Bellavista, Filippo Bosi, Luca Foschini, Giuseppe Martuscelli, Rebecca Montanari, Amedeo Palopoli, «Container Orchestration Engines: A Thorough Functional and Performance Comparison,» в *ICC 2019 - 2019 IEEE International Conference on Communications (ICC),* Shanghai, China, 2019.

5. Сафронов Д., Стоноженко К.М., Никифоров И.В., «Автоматическая балансировка нагрузки между потоковой обработкой данных и внутренними задачами кластера с использованием Kubernetes,» *Современные технологии в теории и практике программирования. Dell Technologies; EPAM Systems.,* pp. 165-167, 2020.

6. «Документация Scrapy,» [В Интернете]. Available: https://docs.scrapy.org/en/latest/.

УДК 004.415

Акназарова А. И. (4 курс бакалавриата)
Никифоров И. В., к.т.н., доцент
Прокофьев О. В., ст. преподаватель

ВЫСОКОПРОИЗВОДИТЕЛЬНАЯ ГИБКОМАСШТАБИРУЕМАЯ АРХИТЕКТУРА ПРОГРАММНОГО СРЕДСТВА ДЛЯ ОБРАБОТКИ ДАННЫХ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

На сегодняшний день одной из самых популярных и востребованных тематик в области разработки программного обеспечения является тематика хранения и обработки больших объемов данных [1], производимых различными источниками, такими как: мобильные телефоны, сенсоры, датчики и кассовые аппараты.

Целью работы является повышение эффективности оценки использования контрольно-кассовых машин города Москва за счет программного средства, реализующего алгоритмы генерации отчетов по разнотипным источникам данных.

Задачами, необходимыми для достижения цели являются:

1. Провести обзор существующих методов хранения и обработки больших данных.
2. Предложить архитектуру программного средства, обладающую свойствами гибкого масштабирования, высокой производительностью и возможностью обработки данных в режиме реального времени.
3. Разработать алгоритмы генерации отчетов по данным контрольно-кассовых машин города Москва.
4. Реализовать программную архитектуру и предложенные алгоритмы в программном средстве.
5. Продемонстрировать эффективность применения программного средства на тестовых данных.

В данной статье рассматривается архитектура программного средства, обладающая свойствами гибкого масштабирования, высокой производительностью и возможностью обработки данных в режиме реального времени.

Данная архитектура программного средства имеет гибридный метод хранения и обработки информации, а именно используется как хранилище данных – Oracle Exadata, Oracle Database, так и озеро данных [2], что дает все вышеописанные свойства.

Рассмотрим подробнее на примере декомпозиционной диаграммы [3] процесс загрузки исходных данных из источника в озеро данных (см. рисунок 1).

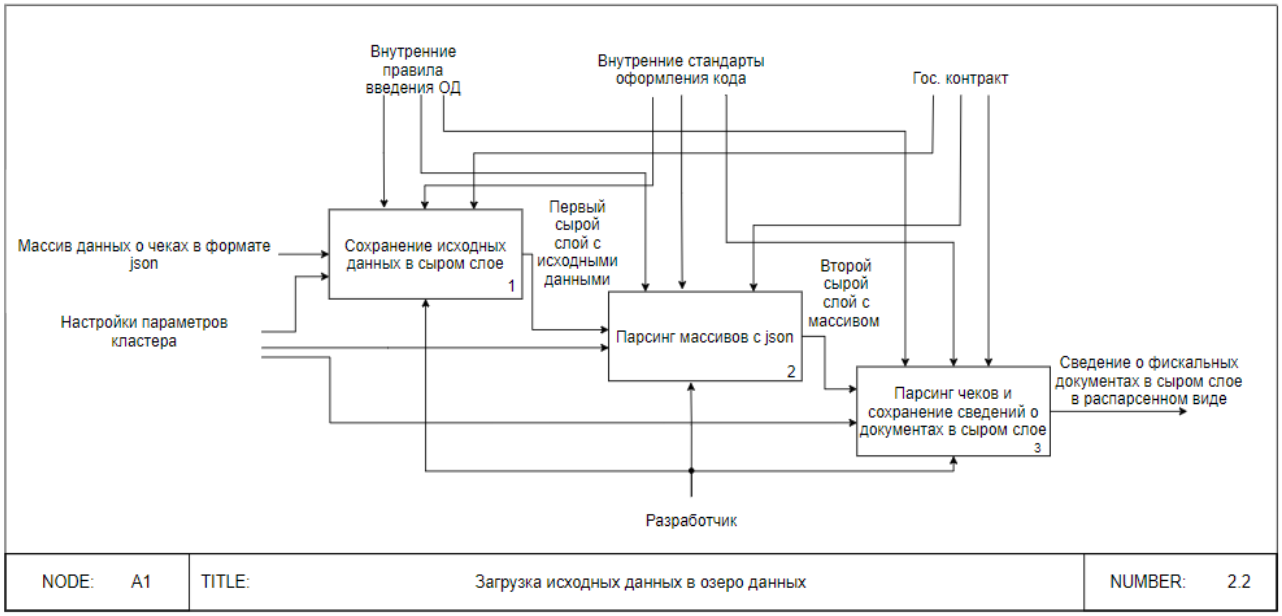


Рисунок 1 – Декомпозиционная диаграмма по процессу «Загрузка исходных данных в озеро данных»

Весь процесс «Загрузка исходных данных в озеро данных» состоит из 3 блоков:

1. Сохранение исходных данных в сыром слое.
2. Парсинг массивов с json.
3. Парсинг чеков и сохранение сведений о документах в сыром слое.

Для каждого из процессов на данной декомпозиционной диаграмме в качестве механизма, лица, производящего работу, выступает разработчик, а в качестве механизмов управления используются: внутренние правила введения ОД, внутренние стандарты оформления кода, государственный контракт.

В качестве входящих в первом процессе «Сохранение исходных данных в сыром слое» принимается массив данных о чеках в формате json, загруженный из Apache Kafka [4] и настройки параметров кластера.

Под настройками параметров кластера подразумевается выделение памяти, количества ядер, исполнителей, тип загрузки данных, временной интервал выполнения и т.д. Стоит отметить, что настройки параметров кластера для каждого процесса индивидуальны и прописываются в конфигурационных. yml файлах [5].

На выходе из первого блока-процесса результатом деятельности будет первый сырой слой с исходными данными и техническими обозначениями.

В качестве входа в процессы 2 и 3 используются также настройки параметров кластера, а также выходы из прошлого процесса: 1 и 2 соответственно.

На выходе из 3 процесса получаем сведения о фискальных документах в сыром слое в распарсенном виде, что в свою очередь позволит в любой момент времени пересоздать ту или иную витрину данных с необходимыми полями из распрасенной таблицы в сыром слое, что позволит повысить эффективность оценки использования контрольно-кассовых машин города Москва.

Предложенная архитектура позволяет обеспечить высокопроизводительную гибкомасштабируемую архитектуру программного средства для обработки данных в режиме реального времени.

ЛИТЕРАТУРА

1. Ивлев В.А., Никифоров И.В., Леонтьева Т.В., Обработка данных в геоинформационных системах для выбора местоположения рекламы, в книге: Современные технологии в теории и практике программирования. сборник материалов конференции. 2019. С. 27-30.

2. В.А. Смирнов, А.Р. Андреева, А.В. Севаянян, Особенности концепции хранения данных - озеро данных, Проблемы эффективного использования научного потенциала общества, Сборник статей по итогам Международной научно-практической конференции, 2019. – с. 55-57
3. Т.Б. Новикова, Л.В. Курзаева, В.Е. Петеляк, О.Е. Масленникова, И.Д. Белоусова, Описание управления бизнес-процессами предприятия на основе методологии Idef0: трудности разработки, рекомендации по совершенствованию построения диаграмм, Фундаментальные исследования, Общество с ограниченной ответственностью "Издательский Дом "Академия Естествознания" (Москва), 2015. – с. 318-322
4. В.Е. Павлов, Н.А. Тюкачев, Разработка графического интерфейса для представления сообщений из Apache Kafka в плоском виде, Молодежь и системная модернизация страны, Сборник научных статей 3-й Международной научной конференции студентов и молодых ученых. В 4-х томах. Ответственный редактор А.А. Горохов, 2018. – с. 129-131
5. Сысоев И.М., Никифоров И.В., Каплан Е.В., Создание подхода автоматизации обновления конфигурационных файлов программного обеспечения, В книге: Современные технологии в теории и практике программирования. сборник материалов конференции. 2019. С. 126-127.

УДК 004.415.2

Аникин Н.В. (4 курс бакалавриата)
Медведев Б.М., к.т.н., доцент

СОЗДАНИЕ ПРОГРАММНОГО КОМПЛЕКСА СБОРА И ХРАНЕНИЯ СТАТИСТИКИ СЕТЕВОГО ТРАФИКА ПРОМЫШЛЕННОГО НАЗНАЧЕНИЯ

Предприятия используют комплексы статистической обработки сетевого трафика для мониторинга загрузки сетевого оборудования и каналов связи, а также для анализа взаимодействия между узлами сети для обеспечения информационной безопасности.

Известные программные средства анализаторов сетевого трафика, такие, как Cisco Stealth Watch, nTop и др., представляют собой мощные и дорогостоящие системы мониторинга, бесплатные версии которых имеют ограничения по производительности [1, 2]. Кроме того, некоторые предприятия, имеющие стратегическое значение в экономике, могут иметь ограничение на использование импортного программного обеспечения. В связи с этим актуальной является задача разработки систем мониторинга и аналитики безопасности сетевой инфраструктуры для отечественных предприятий. В работе рассматривается проект Efros CI, разрабатываемый ООО "Газинформсервис", для которого необходимо разработать программные средства сбора и хранения статистики сетевого трафика [3].

Программные средства сбора используют данные телеметрии от маршрутизаторов, коммутаторов, межсетевых экранов и других сетевых инфраструктурных устройств разных производителей. Информация представлена в виде потоковых данных, сформированных по правилам различных протоколов: NetFlow, IPFIX, sFlow, NetStream [4 - 7].

Целью работы является разработка программного комплекса сбора и хранения статистики сетевого трафика промышленного назначения, обеспечивающего декодирование и нормализацию пакетов NetFlow, IPFIX, sFlow и NetStream.

Для достижения поставленной цели решены следующие задачи:

1. Разработка программных средств обработки пакетов.

2. Разработка средств тестирования программного комплекса.

Разработанные программные средства обработки пакетов состоят из следующих компонентов:

– Приложение Flow-Collector, выполняющее декодирование приходящих пакетов;

– База данных GeoIP, которая используется для сопоставления географического местоположения оборудования и IP-адресов, приведённых в каждом пакете;

–Распределённый программный брокер сообщений Apache Kafka [8];

– Программный комплекс Elastic Stack, обеспечивающий хранение и дедупликацию собранных данных.

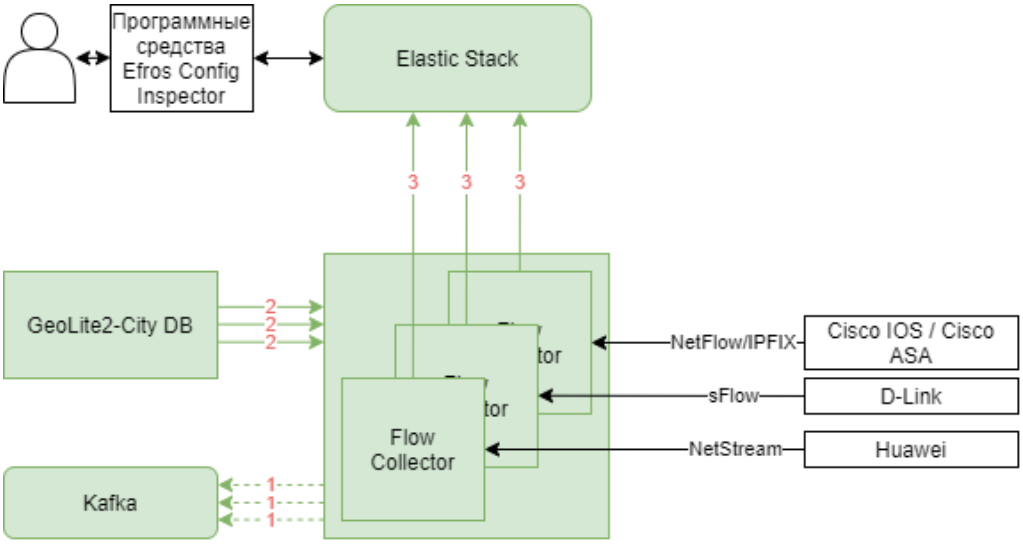


Рисунок 1– Функциональная схема сбора и хранения статистики сетевого трафика

На рисунке 1 представлена функциональная схема разрабатываемого комплекса. Маршрутизаторы различных производителей (Cisco, D-Link, Huawei) формируют пакеты, содержащие информацию о сетевом трафике, и отправляют их на обработку в Flow-Collector, используя сетевой стек протоколов UDP/IP. Каждый из трех процессов Flow-Collector обрабатывает пакеты только определенного потока (NetFlow/IPFIX, sFlow, NetStream) за счет установки разных UDP портов в настройках маршрутизаторов. Каждый пакет проверяется на соответствие ожидаемому протоколу, данные пакета декодируются и преобразуются к единому формату, а также к данным добавляется информация о местоположении сетевого оборудования, извлеченная из БД GeoLite2-City по IP-адресам. Собранная информация передается на хранение и последующую обработку в Elastic Stack. Программные средства Efros Config Inspector обеспечивают интерфейс с пользователем. Распределённый программный брокер сообщений Apache Kafka используется для тестирования и отладки программного обеспечения, а также для последующего расширения функциональности комплекса.

Параметрами запуска Flow-Collector являются:

– ожидаемый формат сообщений (название протокола) и UDP порт для прослушивания;

– количество создаваемых потоков выполнения для обработки поступающих на UDP порт сообщений;

– IP-адреса и UDP порты компьютеров, где развёрнуты Elastic Stack и Apache Kafka. База данных GeoLite2-City DB представлена в виде файла формата .mmdb, который содержит диапазоны IP-адресов и соответствующие им географические расположения.

Для тестирования был разработан программный генератор тестовых пакетов, отправляющий тестовые пакеты NetFlow на UDP порт по заданному IP адресу. Генератор позволяет задавать задержку между пакетами в диапазоне от 0,4 мс до 5 мс и длительность сеанса тестирования (количество тестовых пакетов).

Результаты тестирования показали, что пакеты с информацией о сетевом трафике обрабатываются в соответствии с требованиями. При большой нагрузке возникают потери пакетов, что связано со ограниченной производительностью компьютера, используемого для тестирования.

ЛИТЕРАТУРА

1. Cisco Secure Network Analytics (Stealthwatch) // Cisco. – URL: https://www.cisco.com/c/ru_ru/products/security/stealthwatch/index.html (дата обращения: 01.03.2021).

2. ntop – High Performance Network Monitoring Solutions based on Open Source and Commodity Hardware. // ntop. – URL: https://www.ntop.org/ (дата обращения: 01.03.2021).
3. Efros Config Inspector // Газинфорсервис. – URL: https://www.gaz-is.ru/produkty/zashchita-it-infrastrukturi/efros-config-inspector.html (дата обращения: 01.03.2021).
4. Introduction to Cisco IOS NetFlow - A Technical Overview // Cisco. – URL: https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html (дата обращения: 01.03.2021).
5. IP Flow Information Export (IPFIX) Entities // Internet Assigned Numbers Authority – URL: https://www.iana.org/assignments/ipfix/ipfix.xhtml (дата обращения: 01.03.2021).
6. sFlow Version 5 // sFlow – URL: https://sflow.org/sflow_version_5.txt (дата обращения: 01.03.2021).
7. Overview of NetStream // Huawei. – URL: https://support.huawei.com/enterprise/en/doc/EDOC1000178174/986bf11e/overview-of-netstream (дата обращения: 01.03.2021).
8. Documentation // Apache Kafka https://kafka.apache.org/documentation/ (дата обращения: 01.03.2021).

УДК 004.457

Забелин К.В., Максимчук В.А., Шемякинская А.С. (1 курс магистратуры)
Никифоров И.В., к.т.н., доцент

АВТОМАТИЗАЦИЯ СБОРКИ И РАЗВЕРТЫВАНИЯ СИСТЕМЫ АНАЛИЗА ДАННЫХ ЭЛЕКТРОННОЙ БИБЛИОТЕКИ СПБПУ

Важным этапом разработки программного обеспечения является сборка и развертывание конечного продукта. Эта задача усложняется, когда в системе используется большое количество компонентов, которые взаимодействуют между собой. Так как пользователям требуется простой и интуитивно понятный способ взаимодействия с реализованной системой анализа данных [1], задача сборки и развертывания конечного дистрибутива является актуальной.

Целью работы можно считать упрощение использования разработанной системы анализа данных конечным заказчиком. Для достижения этой цели, необходимо решить следующие задачи:

1) Проанализировать существующие подходы к сборке и развертыванию приложений.
2) Предложить структуру и процесс развертывания дистрибутива с системой.
3) Разработка необходимых для сборки конфигурационных файлов и скриптов.
4) Сборка конечного дистрибутива и отправка конечному пользователю.

Система анализа данных содержит сервис загрузки данных из библиотеки и локальное хранилище [2], непосредственно ElasticSearch и Kibana. Необходимо автоматизировать сборку сервиса и запуск всех ресурсов локально на платформе пользователя. В нашем случае это – Windows 10. Для этого было предложено использовать Docker [3], чтобы создать образ сервиса, и Docker Compose [4] для совместного развертывания ElasticSearch, Kibana и загрузчика данных. Docker позволяет создавать и запускать контейнеры, а Docker Compose используется для управления развертыванием нескольких контейнеров и упрощает конфигурацию сети и томов, необходимых для их взаимодействия. Чтобы автоматически инициализировать образ контейнера был написан специальный Dockerfile [5], который содержит необходимые для сборки приложения команды. Пример Dockerfile для сервиса загрузки данных представлен в листинге 1.

```
FROM golang:1.14 AS builder
RUN apt-get update -y && apt-get upgrade -y
COPY ./ /go/src/
WORKDIR /go/src/connector/
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o /go/bin/connector.app ./src/*.go

FROM alpine:latest AS runner
RUN apk -U upgrade && apk add python3 && apk add py3-pip
WORKDIR /go/connector/app/
COPY ./requirements.txt .
RUN pip3 install -r requirements.txt
COPY --from=builder /go/bin/connector.app .
COPY --from=builder /go/src/lib/ .
ENTRYPOINT [ "./utils/startup.sh" ]
```

Листинг 1 – Dockerfile для сервиса загрузки данных

Изначально для развертывания был предложен подход использования bat-скриптов, которые устанавливали необходимые зависимости с помощью chocolatey [6], такие как VirtualBox [7], Docker Compose, Docker Machine [8]. Далее вызывался скрипт по созданию Docker Machine, которая в свою очередь создавала виртуальную машину с Linux. Внутри этой виртуальной машины запускался Docker Compose, который с помощью заданного конфигурационного файла поднимал сервис загрузки данных, ElasticSearch и Kibana. Такой подход к автоматизации сборки и развертывания оказался неуниверсальным при развертывании на различных окружениях и сложен в отладке ошибок.

Другой метод - использование Docker Desktop [9] оказалось также проблематичным, так как установка инструмента требовала нескольких перезагрузок машины, включенную опцию Hyper-V в BIOS и ручное конфигурирование настроек, что усложняло процесс использования дистрибутива конечным пользователем.

В итоге было принято решение использовать совместно Vagrant [10] и bat-скрипты. Vagrant используется для упрощения конфигурирования виртуальной среды. Bat-скрипты запускают и останавливают работу скрипта Vagrant. Vagrant в свою очередь конфигурирует виртуальную машину VirtualBox, устанавливает на ней необходимые для развертывания зависимости и запускает Docker Compose с конфигурационным файлом контейнеров. После этого пользователь может увидеть URL Kibana в консольном выводе скрипта и использовать его в браузере для доступа к данным. Таким образом пользователю необходимо запустить инсталяционный скрипт, который установит на машины Vagrant и VirtualBox, перезагрузить компьютер, запустить стартовый скрипт и открыть браузер с полученным URL Kibana. Такой способ оказался наиболее отказоустойчивым, потребовал минимальное количество зависимостей на машине и меньшее количество написанных скриптов.

В результате работы был получен дистрибутив со структурой, продемонстрированной на рисунке 1. В папке дистрибутива содержатся bat-скрипты для запуска пользователем.

```
|   clean.bat
|   install.bat
|   README.txt
|   start.bat
|   Vagrantfile
|
+---config
|       config.yaml
|       docker-compose.yml
|       elasticsearch.yml
|       kibana.yml
|
\---scripts
        cleanup.ps1
        cleanup_docker.ps1
        install_docker.ps1
        setup_docker.ps1
```

Рисунок 1 – Дистрибутив системы анализа данных библиотеки

Таким образом был собран рабочий дистрибутив системы анализа данных библиотеки на платформе Windows 10. В дальнейшем планируется улучшить производительность развертывания и исправить недостатки текущей реализации, например, необходимость перезагрузки системы после установки Vagrant.

ЛИТЕРАТУРА

1. Исходный код проекта [Электронный ресурс]. Режим доступа: https://github.com/safronovD/spbstu-smart-librry
2. Петров А.А., Селин И.А., РАЗРАБОТКА ПРОГРАММЫ ПО ОБЪЕДИНЕНИЮ ХРАНИЛИЩ ДАННЫХ, НЕДЕЛЯ НАУКИ СПбПУ. материалы научной конференции с международным участием. Санкт-Петербургский политехнический университет Петра Великого. 2019. С. 85-87.
3. Максимчук В.А., Никифоров И.В., Разработка системы масштабирования реплик приложений с целью увеличения отказоустойчивости узлов кластера Kubernetes, Современные технологии в теории и практике программирования. сборник материалов конференции. Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. Санкт-Петербург, 2020. С. 160-161.
4. Shemyakinskaya A.S., Nikiforov I.V., Hard drives monitoring automation approach for Kubernetes container orchestration system, Proceedings of the Institute for System Programming of the RAS. 2020. Т. 32. № 2. С. 99-106.
5. Васильченко А.В., Воинов Н.В., Метод автоматической проверки лимитов ресурсов облачного провайдера AWS, Современные технологии в теории и практике программирования. сборник материалов конференции. Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. Санкт-Петербург, 2020. С. 146-147.
6. Chocolatey [Электронный ресурс]. Режим доступа: https://chocolatey.org/install
7. VirtualBox [Электронный ресурс]. Режим доступа: https://www.virtualbox.org/
8. Документация к Docker Machine [Электронный ресурс]. Режим доступа: https://docs.docker.com/machine/
9. Документация к Docker Desktop [Электронный ресурс]. Режим доступа: https://www.docker.com/products/docker-desktop
10. Документация к Vagrant [Электронный ресурс]. Режим доступа: https://www.vagrantup.com/docs

УДК 004.04

Иванов Г.В., Власов Д.В., Околович Л.Д. (4 курс бакалавриата)
Никифоров И.В., к.т.н., доцент

ПРОЕКТИРОВАНИЕ REST-ИНТЕРФЕЙСА СИСТЕМЫ ТРАНСКОДИРОВАНИЯ ВИДЕОПОТОКОВ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

Транскодирование видеопотоков является необходимой частью всех видеосервисов, стриминговых сервисов, которые стремительно набирают популярность [1]. Существующие системы транскодирования обладают рядом недостатков, которые накладывают ограничения на спектр решаемых задач.

К таким недостаткам относятся:

- Посредственная горизонтальная масштабируемость
- Высокая стоимость
- Недостаточная скорость работы

Поэтому актуальна задача разработки собственной системы транскодирования видеопотоков, свободной от перечисленных недостатков, проектирование ее REST интерфейса.

Целью работы является снижение трудоемкости взаимодействия с системой транскодирования, за счет разработки REST интерфейса, предоставляющего возможности

создания подключаемых клиентов, например, веб-приложения, и предоставляющего возможности горизонтального масштабирования.

Рассмотрим архитектуру разрабатываемой системы, представленную на Рисунке 1.

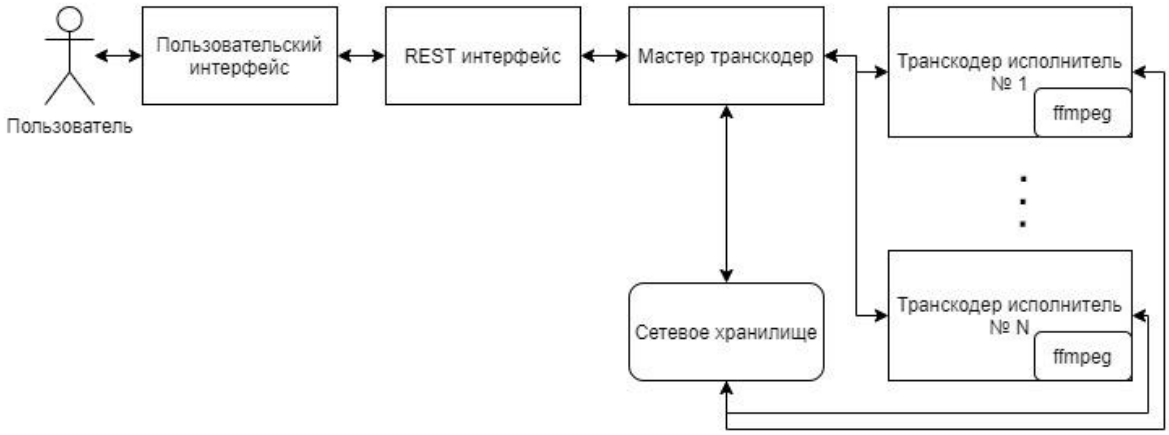


Рисунок 1 – Архитектура разрабатываемой системы.

Всю систему можно разделить на серверную часть и пользовательский интерфейс. Серверная часть, в свою очередь, разделена на мастер транскодер и транскодеры-исполнители. Каждый из них имеет доступ к сетевому хранилищу, в котором хранятся входные, промежуточные и итоговые видеофайлы, полученные в результате транскодирования. Сам процесс транскодирования осуществляется с помощью инструмента ffmpeg [2].

Таким образом, REST интерфейс должен обеспечить взаимодействие веб-интерфейса и мастер транскодера, а также обеспечить взаимодействие между мастер транскодером и транскодерами-исполнителями.

Опишем сценарии использования и требования, представленные на Рисунке 2 в виде MSC-диаграмм [3].

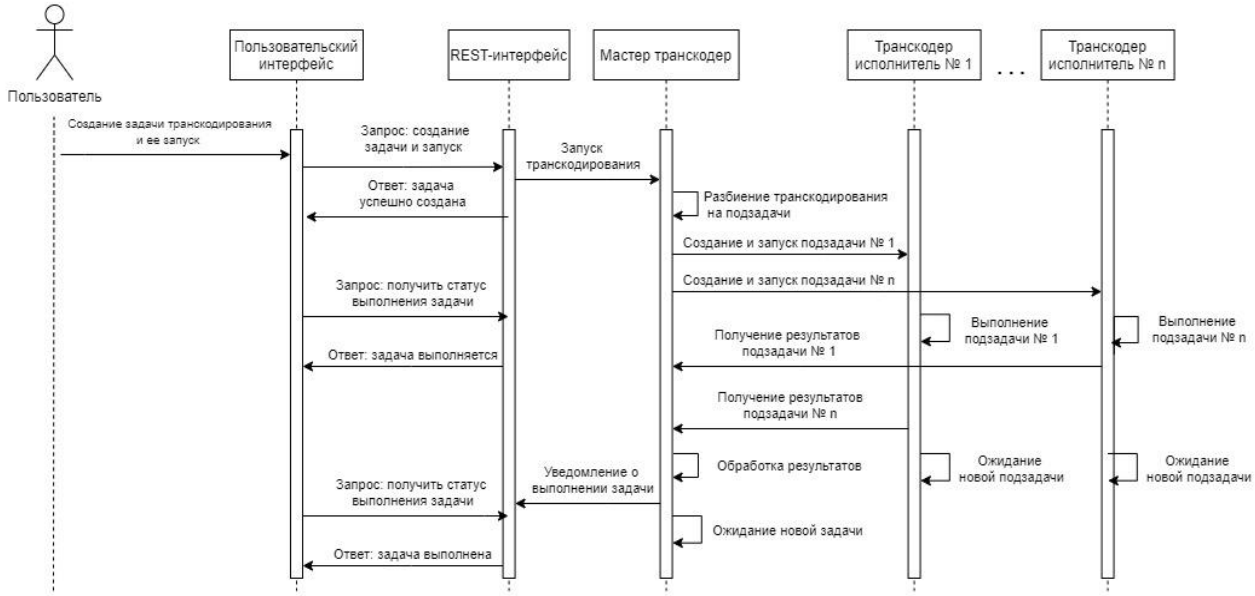


Рисунок 2 – Сценарий использования системы в виде MSC-диаграммы.

Пользователю необходимо дать возможность с помощью интерфейса:

– Видеть список задач с возможностью его редактирования - создать новые задачи, удалить существующие
– Запускать задачи на транскодирование, останавливать, видеть их статус и прогресс выполнения
– Видеть загрузку процессора, количество используемой и свободной оперативной памяти

Это значит, что соответствующие конечные точки должны быть также реализованы в REST интерфейсе.

Также при взаимодействии между мастер транскодером и транскодерами исполнителями необходимо:

- Иметь возможность создавать и удалять транскодеры исполнители при необходимости расширения или сужения кластера.
- Предоставить возможность транскодерам исполнителям узнавать наличие задач для исполнения и их параметры
- Предоставить возможность мастер транскодеру узнавать статус выполнения задач на каждом транскодере исполнителе

Интерфейс, удовлетворяющий вышеперечисленным условиям, позволит снизить трудоемкость использования системы транскодирования для конечных пользователей.

Также такой REST интерфейс позволит реализовать возможность горизонтального масштабирования, что в свою очередь позволит повысить производительность системы [4].

Работа выполнена в рамках совместного проекта СПБПУ и компании Okko.

ЛИТЕРАТУРА

1. Сычев В.К., Ленькова Ю.В., Цветкова Е.А., Никифоров И.В., анализ данных трендов youtube, Современные технологии в теории и практике программирования. сборник материалов конференции. Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. Санкт-Петербург, 2020. С. 131-132.
2. FFmpeg [Электронный ресурс]. Режим доступа: https://ffmpeg.org/
3. Воинов Н.В., Дробинцев П.Д., Котляров В.П., Никифоров И.В., Селин И.А., Генерация тестового набора на основе потока управления, Системы и средства информатики. 2015. Т. 25. № 1. С. 54-73.
4. Никифоров И.В., Курсовое проектирование по учебной дисциплине "Наука о данных и аналитика больших объемов информации", Учебное пособие / Санкт-Петербург, 2017.

УДК 004.51

Кольцов А.А. (2 курс магистратуры)
Никифоров И.В., к.т.н., доцент

АВТОМАТИЗАЦИЯ ОПРЕДЕЛЕНИЯ РЕГРЕССИИ ПРОИЗВОДИТЕЛЬНОСТИ СХД

Интернет вещей – одно из наиболее перспективных направлений в компьютерных науках. Множество устройств – камеры, сенсоры, датчики – генерируют множество непрерывных данных, которые возможно анализировать для получения статистики или предоставления какого-либо сервиса.

Анализ непрерывных данных – комплексная задача. В зависимости от ситуации, наибольшую ценность могут представлять, как данные, получаемые в реальном времени, так и исторические данные. Оба случая требуют наиболее быстрой и качественной их обработки. Одна из систем, которая позволяет упростить обработку потоковых данных – *Pravega*.

Pravega – система хранения потоковых данных[1] (далее – СХД) с открытым исходным кодом, которая позволяет хранить данные в виде бесконечного потока - *stream*. В многоуровневую архитектуру системы входят 2 вида хранилища: краткосрочное и долгосрочное, что позволяет оптимально работать с описанными типами данных. Ключевые метрики работы системы – задержка и пропускная способность, они играют важную роль при измерении производительности.

По мере разработки *Pravega* или аналогичных СХД, внедряется новая функциональность, а также добавляются улучшения и исправления. Потенциально, каждое из таких изменений может повлечь за собой регрессию в производительности. Один из способов минимизации вероятности возникновения данной ситуации – введение формального требования к изменениям продукта – они не должны провоцировать не ожидаемую регрессию

производительности. В качестве формализации процедуры проверки требования, возможно определить набор автоматических нагрузочных тестов для новых сборок продукта[4]. Анализ результата нагрузочных тестов позволит утверждать, выполняется требование или нет.

Целью работы является снижение трудоемкости нагрузочного тестирования СХД, а также предотвращение регрессии ее производительности.

Для достижения целей, в работе были решены следующие задачи: 1) реализована автоматизация развертывания системы *Pravega*; 2) реализована автоматизация сборки и развертывания системы нагрузочного тестирования *Mongoose*; 3) реализована автоматизация обработки результатов нагрузочного тестирования.

Задача автоматизации развертывания СХД заключается в разработке решения, которое, на основе определенных входных параметров, развернет тестируемую систему в определенной среде, например: *Kubernetes* кластер.

Автоматизация сборки и развертывания системы нагрузочного тестирования заключается в том, чтобы клиент – приложение, которое генерирует нагрузку и измеряет производительность – инкапсулировал зависимости, соответствующие тестируемой системе.

Реализация обработки результатов нагрузочного тестирования позволит предоставить информацию пользователю в наиболее удобном для чтения формате. Реализовав сравнение с результатами тестирования предыдущих сборок, появится возможность автоматически определять регрессию в случае, если процентная разница между заданными метриками превышает пороговое значение.

Рассмотрим пример. *Pravega* предоставляет *API* для работы с потоковыми данными: запись, чтение, обработка. По мере разработки системы и обновления интерфейсов, существует вероятность возникновения регрессии производительности, измерить которую можно с помощью инструментов для нагрузочного тестирования. Автоматизация процесса тестирования позволит снизить его трудоемкость, а обработка его результатов, их сравнение и отображение в удобном для пользователя виде позволит сократить время на их анализ.

В качестве инструмента для нагрузочного тестирования был выбран *Mongoose*[2]. Его основные плюсы: открытый исходный код, расширение для тестирования системы *Pravega*, а также возможность описания сценария нагрузки в виде *JavaScript*. Открытый исходный код позволит пересобирать инструмент с клиентскими библиотеками, которые соответствуют тестируемой сборке *Pravega*. Возможность описания сценария нагрузки позволит снизить трудоемкость автоматизации – вместо множества запусков *Mongoose*, достаточно будет в качестве аргумента передать описание различных профилей нагрузки, которые проведет инструмент (рисунок 1).

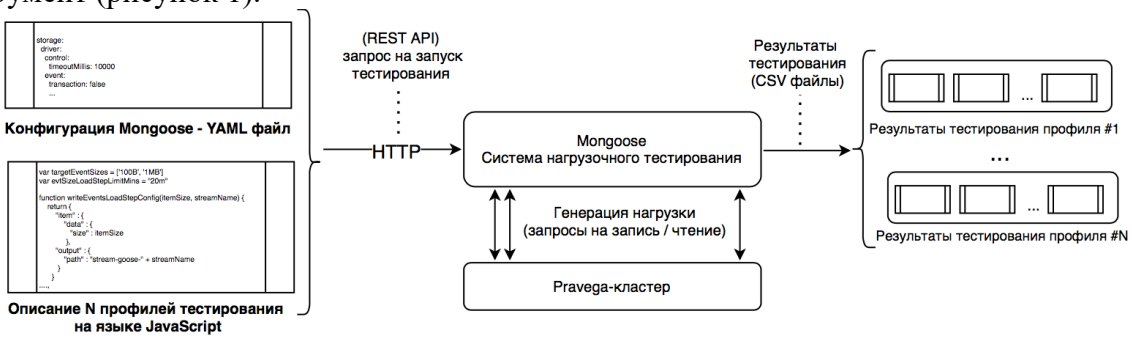


Рисунок 1 – Описание процесса нагрузочного тестирования СХД с помощью *Mongoose*

В качестве инструмента для автоматизации и обработки результатов был выбран *Jenkins* - система с открытым исходным кодом, которая позволяет реализовать непрерывную интеграцию ПО в виде различных задач, например: *Jenkins Pipeline Job* – описанные на языке *Groovy DSL* шаги автоматизации. Утилиты *Jenkins* для выполнения шагов автоматизации в контексте различных *docker*-контейнеров, обработки *CSV* файлов и вызова *shell* команд значительно упростят реализацию поставленной задачи.

Развертывание системы *Pravega* будет проводиться на *Kubernetes* кластере. Минимальная абстракция, позволяющая работать с программой в *Kubernetes* – *pod*, который

состоит из *docker*-контейнеров, развернутых на узлах кластера[5]. Следовательно, для тестирования сборки корневого компонента системы – *Pravega Core* - потребуются обернуть ее в *docker*-контейнер, идентификатор которого будет использоваться для создания *Pravega*-кластера.

Сборка и развертывание системы для нагрузочного тестирования *Mongoose* будет также проводиться в контексте реализуемой автоматизации. Это позволит обнаружить регрессию производительности не только в серверной, но и в клиентской части *Pravega*. Для этого в реализацию было добавлено 2 аргумента: ссылка на *git*-репозиторий, а также *commit hash,* который соответствует тестируемой сборке *Pravega*

Запуск нагрузочного тестирования возможен, как только объект и субъект тестирования развернуты на *Kubernetes*-кластере. Mongoose поддерживает *HTTP*-запросы для контроля и мониторинга - *REST API* [6], что позволяет запустить его с помощью *HTTP POST*-запроса (рис. 1). Автоматизация будет покрывать 2 крайних варианта использования *Pravega*: хранение событий малого и большого размера – 100 Б и 1 МБ соответственно. Генерируя события, *Mongoose* будет измерять различные метрики, в том числе: пропускную способность и задержку обработки событий. Высокие квантили распределения значений задержки – особо важные метрики для систем, которые работают с непрерывными данными в режиме реального времени[3]. После окончания теста, результаты будут занесены в *CSV* файл.

Обработка результатов - последняя стадия автоматизации. Она заключается в отображении основных метрик теста, а также их сравнением с результатами тестирования предыдущих сборок. Результаты *Mongoose* переводятся в *Groovy*-объект, который используется для отображения и сохранения метрик тестирования.

Регрессию производительности можно определить, сравнив результаты тестирования различных сборок. Сравнение возможно реализовать, если данные будут где-либо храниться, например: в базе данных (далее – БД). Используя раннее полученный *Groovy*-объект, возможно сформировать *SQL*-запросы на добавление данных в БД, а, затем, подключаясь по *ssh* к системе управления БД, выполнить их. Аналогично, можно получать данные о результатах тестирования предыдущих сборок.

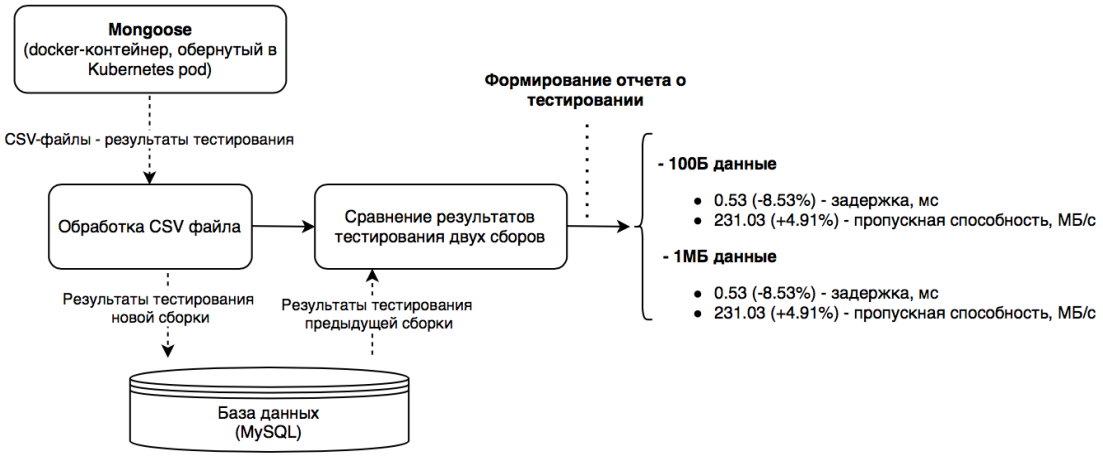


Рисунок 2 – Описание процесса создания отчета о тестировании

На данный момент, описанный подход к автоматизация нагрузочного тестирования полностью реализован: *Jenkins Job* разворачивает заданную сборку *Pravega* на *Kubernetes* кластере, собирает и разворачивает систему тестирования *Mongoose*, проводит нагрузочное тестирование и выводит результаты в виде *HTML* таблицы. В дальнейшем планируется реализация веб-интерфейса, который визуализирует результаты тестирования всех сборок, а также будет содержать отдельные графики для каждой из метрик.

ЛИТЕРАТУРА

1. Pravega [Электронный ресурс] URL: https://www.pravega.io (дата обращения: 13.03.2021)
2. Mongoose Storage Driver Pravega – документация [Электронный ресурс] URL: https://github.com/emc-mongoose/mongoose-storage-driver-pravega/ (дата обращения: 14.03.2021)

3. Yunqi Zhang, David Meisner, Jason Mars, Lingjia Tang, “Treadmill: Attributing the Source of Tail Latency through Precise Load Testing and Statistical Inference”, ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), June 2016 DOI: 10.1109/ISCA.2016.47
4. И. В. Никифоров, П. Д. Дробинцев, В.П. Котляров, “Интегральные критерии проверки требований к программному обеспечению”, Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. Информатика, телекоммуникации и управление. 2013. №3 (174)
5. Шемякинская А.С., Никифоров И.В., “Подход автоматизации мониторинга дисковых носителей для системы оркестрации контейнеров Kubernetes”. Труды Института системного программирования РАН. 2020;32(2), стр. 99-106.
6. В.И. Кочугова, А.А. Кольцов, А.А. Перескокова, И. В. Никифоров, “Проектирование REST-интерфейса для мониторинга и контроля инструмента распределенного тестирования СХД”.Современные технологии в теории и практике программирования: сборник материалов конференции 19 апреля 2019 г. – СПБ. : ПОЛИТЕХ-ПРЕСС, 2019, стр. 224

УДК 004.415.532

Петраченко Я. И. (4 курс бакалавриата)
Котлярова Л. П., ст. преподаватель

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ИНТЕРФЕЙСОВ КОМАНДНОЙ СТРОКИ ДЛЯ УПРАВЛЕНИЯ СИСТЕМАМИ ХРАНЕНИЯ ДАННЫХ КОМПАНИИ DELL TECHNOLOGIES

С ростом объема данных появляется множество новых решений для их хранения. На сегодняшний день активно разрабатываются системы хранения данных (СХД), предлагая большое многообразие систем в зависимости от нужд клиентов. Кластерные СХД представляют собой систему, в которой рабочая нагрузка распределяется равномерно между всеми узлами хранения. Высокая производительность, надёжное хранение данных и возможность оперативного доступа к ним обуславливает применение кластерных систем в самых разных сферах деятельности.

Dell EMC PowerStore [1] – это кластерное устройство компании Dell Technologies, которое обеспечивает многомерное масштабирование, постоянную оптимизацию объёма хранимых данных и поддержку носителей нового поколения [2]. Программный контроль управления кластером осуществляется с помощью нескольких интерфейсов – графического интерфейса пользователя (The PowerStore Manager), RESTful API [3] и интерфейса командной строки (Command Line Interface, сокр. CLI) [4].

The PowerStore Manager [5] – это web-интерфейс, запускающийся на контроллерах системы PowerStore. The PowerStore Manager предоставляет администраторам СХД простой и интуитивно понятный интерфейс для настройки и управления кластером. Он не требует установки дополнительного программного обеспечения на клиенте и помогает выполнить следующие задачи:

– начальную настройку нового узла PowerStore,
– добавление или удаление узлов существующего кластера,
– управление ресурсами кластера.

Несмотря на разнообразные возможности, предоставляемые The Powerstore Manager, некоторые пользователи предпочитают использовать другие доступные интерфейсы для более гибкого управления системой PowerStore. REST API [6] представляет собой набор объектов, операций и атрибутов, которые обеспечивают интерактивное и программное управление СХД PowerStore. Благодаря REST API пользователь может выделять новые компоненты кластера, такие как устройства и тома, выполнять аварийное переключение при отказе оборудования, создавать снимки для резервного копирования, а также собирать метрики для анализа статистических данных. Для того, чтобы управлять системой в своём пользовательском

окружении через REST API, можно воспользоваться полнофункциональным инструментом командной строки, который поддерживает PowerStore.

CLI [7] – это клиентское приложение, которое позволяет выполнять команды для управления СХД PowerStore с помощью интерфейса командной строки. Для взаимодействия с СХД CLI отправляет команды через защищенный протокол HTTPS с помощью REST API. Модель взаимодействия пользователя с СХД через интерфейс командной строки представлена ниже:

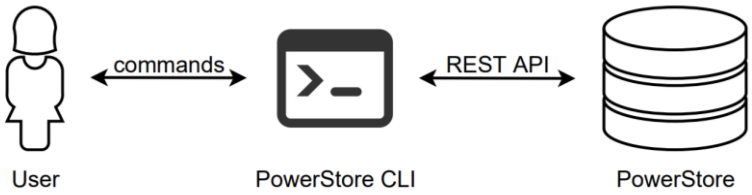


Рисунок 1 – Схема взаимодействия пользователя с СХД PowerStore через интерфейс командной строки.

Интерфейс командной строки PowerStore предназначен для пользователей, которые хотят интерактивно управлять системой PowerStore или использовать команды в сценариях для автоматизации рутинных задач. Например, таких как:

– конфигурирование и мониторинг системы (например, конфигурирование management или storage сетей, в случае если какой-либо адрес в сети становится недоступным, пользователь может самостоятельно переконфигурировать сеть, предоставив новые параметры);
– управление пользователями (в системе есть несколько ролей, в зависимости от которых созданные пользователи имеют разрешение на выполнение определенных действий. Пользователь, являющийся администратором, имеет расширенные возможности и может изменять права других пользователей, а также создавать и удалять пользователей);
– предоставление хранилища (резервация или предоставление хранилища для виртуальных машин);
– защита данных (управление правилами репликации и защиты данных, политику защиты можно назначать томам или группам томов);
– контроль доступа хоста к хранилищу (разделение хостов с правами только на чтение, на чтение-запись или запрет доступа).

Проверка функционирования интерфейса командной строки для СХД PowerStore является важной нерешённой задачей на данный момент. Для того, чтобы провести тестирование CLI, необходимо в существующий тестовой фреймворк добавить модуль, описывающий CLI, и автоматизировать тесты с использованием обновленного тестового фреймворка. При разработке модуля для тестирования CLI следует обратить внимание на то, чтобы данный модуль мог легко адаптироваться для тестирования других систем хранения данных. Поэтому в рамках данной работы предлагается провести сравнительный анализ между интерфейсами командной строки, представленными для управления другими СХД компании Dell Technologies. В рамках исследования рассматриваются принципы работы интерфейсов командной строки для разных видов СХД и задачи, которые они решают, для выявления сходств и отличий, которые будут учитываться при разработке модуля.

ЛИТЕРАТУРА

1. Dell EMC PowerStore Scalable All-Flash Storage [Электронный ресурс]. URL: https://www.delltechnologies.com/ru-ru/storage/powerstore-storage-appliance.htm (дата обращения 06.03.2021)
2. NVMe: the next revolution in data storage [Электронный ресурс]. URL: https://www.delltechnologies.com/ru-ru/learn/data-storage/nvme.htm (дата обращения 06.03.2021)
3. Ричардсон Л., Руби С. RESTful Web Services. O'Reilly Media, 2008.
4. Интерфейс командной строки [Электронный ресурс] [2015]. URL: https://teccxx.neocities.org/mx1/cli.html (дата обращения 10.03.2021)

5. Dell EMC PowerStore: PowerStore Manager Overview [Электронный ресурс]. URL: https://www.delltechnologies.com/en-us/collaterals/unauth/white-papers/products/storage/h18150-dell-emc-powerstore-manager-overview.pdf (дата обращения 11.03.2021)
6. Dell EMC PowerStore REST API Developers Guide Overview [Электронный ресурс]. URL: https://dl.dell.com/topicspdf/pwrstr-apidevg_en-us.pdf (дата обращения 11.03.2021)
7. Dell EMC PowerStore CLI User Guide [Электронный ресурс]. URL: https://dl.dell.com/topicspdf/pwrstr-cliug_en-us.pdf (дата обращения 11.03.2021)

УДК 004.42

Петров А. А. (1 курс магистратуры)
Никифоров И. В., к.т.н, доцент
Зайцев И. В., ст. преподаватель

БАЛАНСИРОВКА ДАННЫХ ДЛЯ ПРОГРАММЫ ОБЪЕДИНЕНИЯ СИСТЕМ ХРАНЕНИЯ ДАННЫХ

Ежегодно количество обрабатываемых и хранимых данных растёт [1]. Для целей хранения и обработки активно применяются облачные вычисления и распределенные системы хранения данных (СХД) [2]. Стоит отметить, что в по историческим и экономическим причинам компании разворачивают несколько различных СХД, но в целях максимально эффективного использования имеющегося пространства для хранения информации их можно виртуально объединить. Это возможно сделать при помощи программно-определенных систем хранения данных. При этом пользователи могут иметь разное качество подключения к имеющимся у них СХД и узлам, а последние, скорее всего, взаимодействуют с клиентом с разной скоростью и могут находиться в разных регионах (странах). Нахождение хранилищ в разных странах может приводить к тому, что для данных, например, персональных данных граждан, важны места хранения (страны, регионы). Так, например, федеральный закон «О персональных данных» требует хранить данные о гражданах России только на территории Российской Федерации [3], также ряд ограничений на трансфер данных налагает и General Data Protection Regulation (GDPR).

В программно-определенных системах СХД используется контроллер и балансировщик данных [4] для обеспечения распределенного хранения, обработки и передачи данных. В то же время, балансировщик и контроллер обладают рядом недостатков, связанных с нестабильной скоростью подключений и разной степенью доступности данных.

Исходя из вышесказанного, в работе предлагается доработать контроллер и балансировщик данных для программно-определенных СХД следующим образом:

- внедрить возможность размещения данных только на хранилищах, находящихся в заданном списке регионов;
- динамически анализировать скорость подключения к хранилищу и переопределять его при необходимости из быстрых хранилищ в медленные (менять числовой уровень скорости взаимодействия с хранилищем).

Существующие решения, такие как DellEMC ScaleIO (Dell EMC PowerFlex) [5], CEPH [6], SANsymphony-V[7], позволяют построить СХД над разным набором хранилищ (узлов), учитывая скорость взаимодействия хранилища с конкретным узлом, которая задаётся пользователем при помощи числового значения из конкретного диапазона. Большинство СХД поддерживают эффективную репликацию данных, для создания реплики в общем случае требуется меньше 50% доступного места для хранения [5][8], 50% требуется для простого зерколирования (обычно RAID 1+0). Также все существующие программно-определенные СХД автоматически перераспределяют данные для размещения более часто используемых (горячих) данных на наиболее быстрых хранилищах [7].

Для решения поставленной задачи для каждого хранилища следует добавить метаинформацию о регионе, где он расположен, тем самым добавив его в определенный регион. При этом стоит учесть, что виртуально страны и регионы могут быть объединены в группы по какому-либо признаку, например, странны члены Европейского Союза. В свою очередь к загружаемым на СХД данным также должна быть добавлена информация о том на узлах в каких регионах, группах загружаемые данные могут быть размещены. Отсутствие такой информации означает, что данные можно размещать на любом хранилище. При этом все куски данных, полученные из исходного файла, должны иметь такие же ограничения, как и весь файл. Далее при работе с блоками, имеющими ограничения по месту хранения, балансировщик выбирает хранилища только из соответствующих регионов и в соответствии с распределением данных на «горячие» и «холодные» распределяет их по хранилищам из выборки.

При этом все хранилища, подключенные к СХД раз в определенный промежуток времени, проходят проверку на доступность и скорость доступа. Проверка на доступность проводится для своевременной автоматической репликации данных, утративших свои реплики (может проводиться путем запроса контрольного блока данных или при помощи ping). Проверка на скорость доступа проводиться для корректного размещения «холодных» и «горячих» данных между хранилищами. Данные полученные при помощи этой проверки, а также данные, введённые пользователем для каждого хранилища, учитываются при назначении числового маркера скорости доступа к хранилищу. Проверка по скорости доступа также может быть проведена при помощи записи и скачивания контрольного блока, или как минимум при помощи утилиты ping. Из чего следует, что обе проверки могут проходить одновременно, но проверки доступности хранилища должны проводиться чаще, чем проверки скорости доступа к хранилищу, так как потеря доступа к данным для клиентов является большей проблемой, чем замедление доступа к ним.

Полученное решение позволило:

- эффективно объединить имеющиеся системы хранения данных, находящиеся в разных странах. В случаях, когда к месту хранения информации накладываются определенные требования;
- динамически оценивать скорость доступа к имеющимся хранилищам, что увеличит скорость доступа к хранимой информации. Недостатком динамической переоценки скорости доступа к хранилищам является то, что в случае частого изменения скорости доступа к какому-либо из хранилищ балансировщик ресурсов осуществляет большое количество операций записи и чтения, что в свою очередь приведет к снижению производительности решения целиком.

ЛИТЕРАТУРА:

1. Воинов Н.В., К. Родригес Гарсон, Никифоров И.В., Дробинцев П.Д., Система обработки больших данных для анализа событий репозитория Github, Международная конференция по мягким вычислениям и измерениям. 2019. Т. 1. С. 283-286.
2. Никифоров И.В., Курсовое проектирование по учебной дисциплине "Наука о данных и аналитика больших объемов информации", Учебное пособие / Санкт-Петербург, 2017.
3. О внесении изменений в отдельные законодательные акты Российской Федерации в части уточнения порядка обработки персональных данных в информационно-телекоммуникационных сетях. Федеральный закон № 242 от 21 июля 2014.
4. Сафронов Д., Стоноженко К.М., Никифоров И.В., Автоматическая балансировка нагрузки между потоковой обработкой данных и внутренними задачами кластера с использованием Kubernetes, В сборнике: Современные технологии в теории и практике программирования. сборник материалов конференции. Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. Санкт-Петербург, 2020. С. 165-167.
5. Dell Tecnologies // PowerFlex Spec Sheet, 2021. https://www.delltechnologies.com/ru-ru/storage/powerflex.htm#scroll=off&pdf-overlay=/ru-ru/collaterals/unauth/briefs-handouts/products/storage/powerflex-specification-sheet.pdf . Дата обращения: 12.03.2021

6. CEPH // Intro to CEPH, 2021. https://docs.ceph.com/en/latest/start/intro/. Дата обращения: 13.03.2021
7. DataCore // DataCore SANsymphony, 2021. https://www.datacore.com/products/sansymphony/. Дата обращения: 13.03.2021
8. Dell Technologies // Dell EMC PowerScale F600 All-Flash NAS Node. https://www.delltechnologies.com/ru-ru/storage/powerscale/powerscale-f600-all-flash.html. Дата обращения: 13.03.2021

УДК 004.051

Ревтова Н.Д. (4 курс бакалавриата)
Котлярова Л.П., ст. преподаватель

РАЗРАБОТКА ФРЕЙМВОРКА ДЛЯ АДАПТИВНОЙ КОНФИГУРАЦИИ ВНУТРЕННИХ ПАРАМЕТРОВ СХД С ЦЕЛЬЮ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ

На сегодняшний день существует множество систем хранения данных, демонстрирующих высокую производительность, однако с ростом объёма хранимых данных возникает необходимость улучшать её показатели. Иногда для повышения производительности СХД оптимизируется код, перестраивается архитектура продукта или вводится в эксплуатацию более новое и быстрое оборудование. Целью моей работы является исследование возможности повышения производительности системы хранения данных путём оптимизации её внутренних параметров.

СХД поставляется с большим количеством настраиваемых параметров, которые управляют её поведением. Настройка таких параметров может обеспечить значительное увеличение производительности, но является сложной задачей из-за нелинейного поведения системы и в некоторых случаях отсутствия числовых значений (например, тип файловой системы). Поэтому традиционно разработчики выбирают конфигурацию системы на основе своих знаний и опыта [1].В итоге заданный набор параметров почти никогда не меняется. Кроме того, одна и та же конфигурация может использоваться, даже когда продукт подвергается различным рабочим нагрузкам.

Для поставленной цели в стадии разработки находится фреймворк.

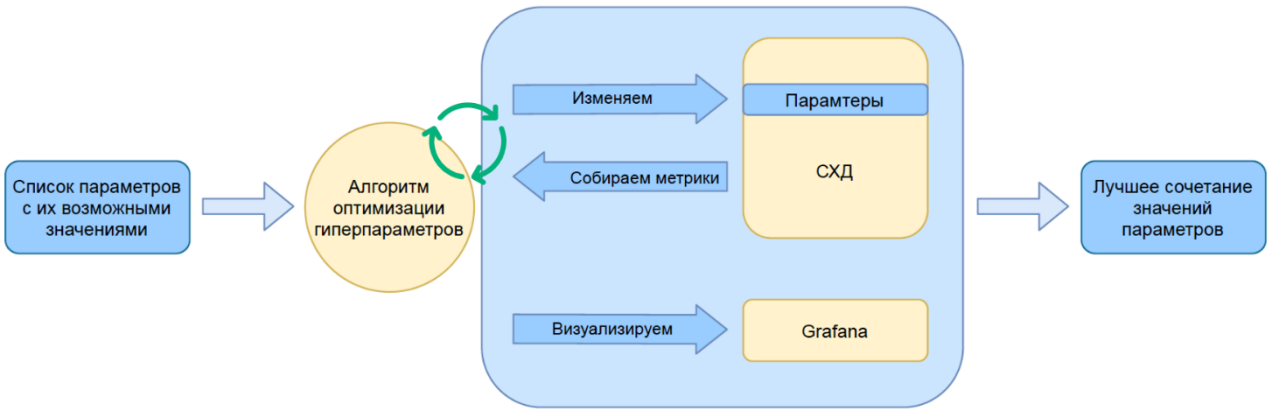


Рисунок 1 – Фреймворк для адаптивной конфигурации внутренних параметров СХД.

Составные части фреймворка: алгоритм оптимизации гиперпараметров на основе машинного обучения, адаптер для изменения параметров СХД, адаптер для извлечения метрик производительности, адаптер для отправки результатов оптимизации в Grafana [2].

Особенности реализации фреймворка: на вход подаётся набор параметров для изменения с установленными пределами для каждого из них. Запускается алгоритм оптимизации гиперпараметров и выдаётся их соответствующий набор. Далее с помощью разработанного адаптера разворачивается СХД с предложенной конфигурацией, и подвергается заданной

нагрузке. Другим адаптером извлекаются метрики производительности, которые должны быть оптимизированы, и отправляются обратно алгоритму оптимизации гиперпараметров, который в свою очередь на их основе генерирует новую конфигурацию для СХД. Соответственно, данный процесс повторяется, пока система не примет решение об остановке всего процесса оптимизации. Чтобы иметь больше информации о результатах и визуализации собранных метрик производительности, используется Grafana. В результате фреймворк предлагает наилучшую конфигурацию параметров, установленных для заданной рабочей нагрузки.

Преимущество данного подхода увеличения производительности СХД в том, что не требуется дополнительных затрат на переорганизацию кода или закупки нового оборудования. Помимо этого, плюсом является возможность подбирать оптимальную конфигурацию системы исходя из нужд клиента.

ЛИТЕРАТУРА

1. Zhen Cao, Geoff Kuenning, Erez Zadok. Carver: Finding Important Parameters for Storage System Tuning. In Proceedings of the 18th USENIX Conference on File and Storage Technologies (FAST'20), 2020.
2. Grafana – многоплатформенное веб-приложение для аналитики и интерактивной визуализации метрик с открытым исходным кодом [Электронный ресурс]. URL:https://grafana.com/(Дата обращения 14.03.2021)

УДК 004.415.25

Саравайский М. А. (2 курс магистратуры)
Маятин А. В., к.п.н., доцент

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ МОДИФИКАЦИИ ПРОДУКТА DISK LIBRARY FOR MAINFRAME ДЛЯ ОБЕСПЕЧЕНИЯ ЭФФЕКТИВНОГО ОБМЕНА ДАННЫМИ МЕЖДУ IBM SYSTEM Z И ДРУГИМИ СИСТЕМАМИ

В наше время для различных целей применяется множество различных вычислительных систем. Разные системы имеют разную продолжительность использования, которая зависит от большого количества факторов. Выбор используемого ПО для конкретной задачи также обуславливается многими факторами, но для критических задач приоритетными являются надежность, качество и совместимость с уже существующей инфраструктурой. Именно поэтому в некоторых отраслях до сих пор используются системы, которые являются прямыми потомками IBM System/360, и поддерживают полную обратную совместимость с ней. В связи с популярностью таких систем отдельное место на рынке систем хранения данных занимают виртуальные библиотеки магнитных лент. Одним из таких продуктов является Dell EMC DLm.

Dell EMC DLm справляется со своими основными задачами, однако, с течением времени появляются новые требования, и необходимость в новой функциональности затрагивает и такие стабильные рынки, как рынок систем хранения для мейнфреймов IBM. Данные, которые обрабатывает мейнфрейм крайне интересны для всевозможных исследований, однако мейнфреймы не предназначены для таких типов нагрузок. Очевидным решением был бы перенос этих данных на другие системы и их исследование с использованием всех преимуществ соврменной инфраструктуры. Однако трудности переноса данных возникают, прежде всего, в связи со множеством особенностей архитектуры систем от IBM: такими как отсутствие привычных иерархических файловых систем, специализированные кодировки, в которых данные хранятся, и проприетарные протоколы передачи данных.

Также существует необходимость переносить определенные нагрузки с мейнфреймов на другие системы, которые в наши дни больше подходят для определенных целей.

В настоящей работе рассматриваются различные способы переноса данных, их особенности и эффективность их использования в промышленных масштабах, дается оценка

эффективности использования тех или иных решений и предлагается использование DLm для осуществления переноса данных с мейнфреймов IBM.

В рамках настоящей работы предполагается определить требования к новой функциональности переноса данных, оценить востребованность на рынке и потенциальные отличия от существующих решений. На основе этого – спроектировать оптимальную модернизацию продукта и реализовать новую функциональность, которая позволит получить доступ к данным используя системы других архитектур. (Рисунок)

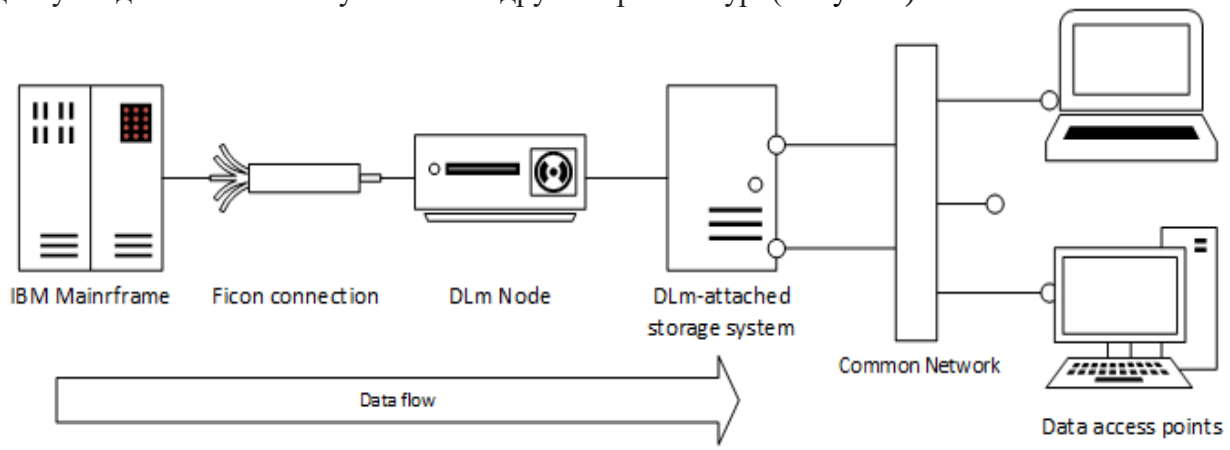


Рисунок 1 – Общая структура решения

В настоящей работе произведен обзор потенциальных решений, альтернатив использованию системы виртуальных магнитных лент и обоснована эффективность предлагаемого решения.

Также в рамках настоящей работы произведен обзор текущей архитектуры DLm, рассмотрены различных способы ее модификации и произведена их оценка. По результатам обзора предложен оптимальный с точки зрения эффективности и безопасности подход к реализации обмена данными между мейнфреймами IBM и системами других архитектур.

Вкратце решение можно описать следующим образом: предлагается ввести новый тип библиотек магнитных лент и новый тип приводов магнитных лент (Рисунок) для того, чтобы на всех этапах обработки данных все участники процесса обладали нужной информацией о данных, при этом сами данные будут оставлены нетронутыми и не будут иметь сложного формата и огромного количества мета-данных.

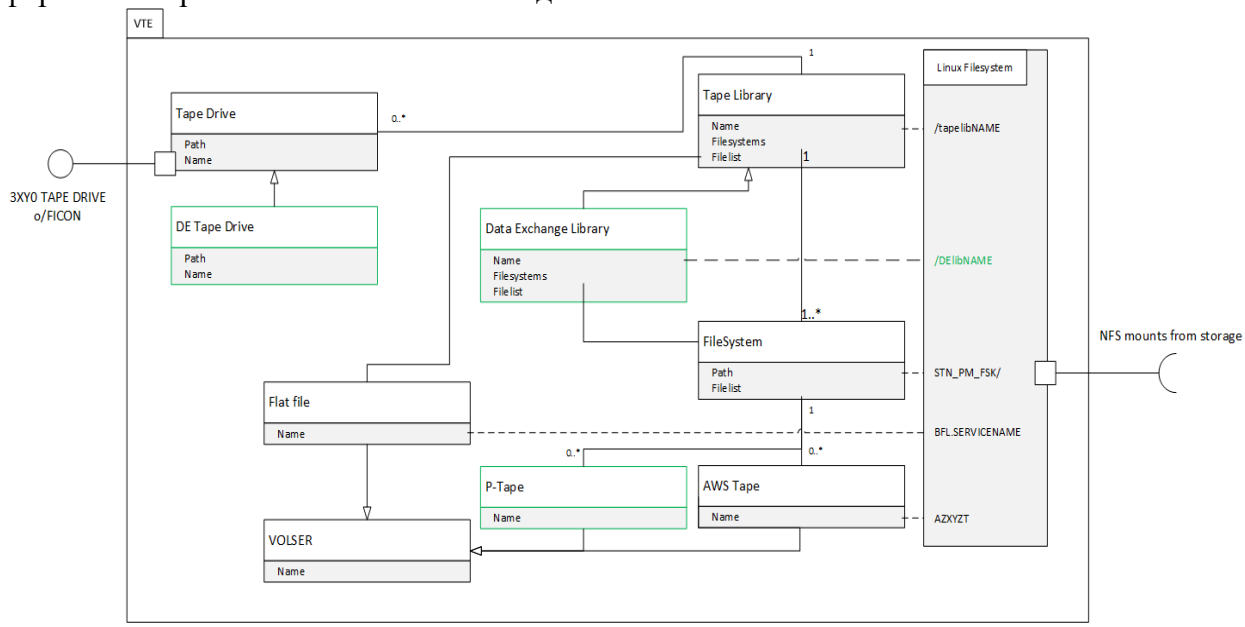


Рисунок 2 – Итоговая реализация

Спроектированное решение является оптимальным и позволяет эффективно переносить данные с мейнфреймов IBM на системы других архитектур для их использования или анализа.

ЛИТЕРАТУРА

1. Dell EMC. (2018). *Virtual Tape Library (VTL)*. Получено из Dell EMC Glossary: https://www.dellemc.com/en-us/glossary/virtual-tape-library.htm
2. Luminex. (2019). *MDI XPDS*. Получено из Luminex website: https://www.luminex.com/download.php?prod=Luminex_MDI_Cross-Platform_Data_Sharing
3. Mike Ebbers, J. K. (2012). *Introduction to the New Mainframe: z/OS Basics.* IBM.
4. Honma, N. T. (14 04 2003 г.). A method of open system file backup by mainframe. *Wiley Periodicals, Inc. Electron Comm Jpn Pt 3,* стр. 65-73.
5. Len Bass, P. C. (2013). *Software Architecture in Practice.* Addison-Wesley Professional.

УДК 004.415

Тимошенский Г. Д. (4 курс бакалавриата)
Котлярова Л. П., ст. преподаватель

ИССЛЕДОВАНИЕ ПОДХОДОВ И РЕАЛИЗАЦИЯ АВТОМАТИЧЕСКОЙ КОНФИГУРАЦИИ СИСТЕМЫ НЕПРЕРЫВНОЙ ИНТЕГРАЦИИ ДЛЯ ПОЛНОСТЬЮ ОБЛАЧНЫХ ПРИЛОЖЕНИЙ

С каждым годом популярность облачной инфраструктуры увеличивается. Если в 2019 году, согласно исследованию Flexera [1], 84% организаций в индустрии информационных технологий использовали в своей работе хотя бы одно публичное или частное облако, то в 2020 году уже 94% компаний использовали как минимум один облачный сервис, заявляет Cloud Industry Forum [2].

Компании осознали полезность облаков в развитии своего бизнеса, поэтому вполне ожидаемо, что по прогнозам аналитиков в 2021 продолжится рост популярности облачной инфраструктуры. В начале года около 88% организаций заявили Capgemini [3], что в перспективе 12 месяцев ожидают увеличения доли облачных технологий в своем бизнесе. Помимо этого, компании сосредоточатся на создания полностью облачных (cloud native) приложений.

Такие приложения были созданы для того, чтобы максимально эффективно использовать ресурсы облака. Зачастую они представляют собой набор слабо связанных между собой микросервисов, которые запущены в контейнерном окружении и динамически управляются с помощью гибких практик DevOps [4].

Чтобы эти приложения могли обрабатывать поступающие данные, можно воспользоваться несколькими методами. Во-первых, можно обрабатывать заявки в реальном времени. Такой способ эффективно использует вычислительные ресурсы, но при этом возможны отказы в обслуживании, если все имеющиеся обработчики заняты. Вторым способом является применение специальных платформ для потоковой обработки данных, которые позволяют хранить очередь заявок до тех пор, пока один или несколько обработчиков не освободятся. Одной из таких платформ является Dell EMC Streaming Data Platform (SDP) [5].

Чтобы продемонстрировать возможности платформы и помочь разработчикам в создании собственных приложений для нее, имеется специальный набор приложений-примеров, которые размещены в репозитории продукта с открытым исходным кодом Pravega, на котором базируется SDP.

В процессе ознакомления с приложениями у разработчиков неизбежно возникнет желание изменить что-то в исходном коде приложений, чтобы лучше разобраться в устройстве платформы и ее преимуществах. Чтобы избежать потенциальных ошибок из-за человеческого

фактора и ускорить процесс ознакомления с возможностями платформы и приложений для нее, предлагается реализовать систему непрерывной интеграции и доставки, которая будет автоматически производить сборку, тестирование и развертывание изменённого приложения.

Из-за популярности практик непрерывной интеграции и доставки существует довольно много инструментов, поэтому процесс выбора правильного инструмента встает наиболее остро. Чтобы сузить перечень обозреваемых систем, будем рассматривать 5 наиболее популярных сервисов по мнению пользователей портала DZone[6], а именно:

– Jenkins
– Buddy
– TeamCity
– Bamboo CI
– GitLab CI

После сравнения CI/CD систем и выбора одной из них для решения поставленной задачи нужно автоматизировать ее установку и настройку в окружении пользователя. Когда система развернута и приведена к необходимой конфигурации, разработчик может вносить изменения в исходный код приложений, демонстрирующих работу SDP, и оперативно получать результаты его сборки, тестирования и развертывания.

Таким образом общая схема работы с разрабатываемой системой выглядит так:

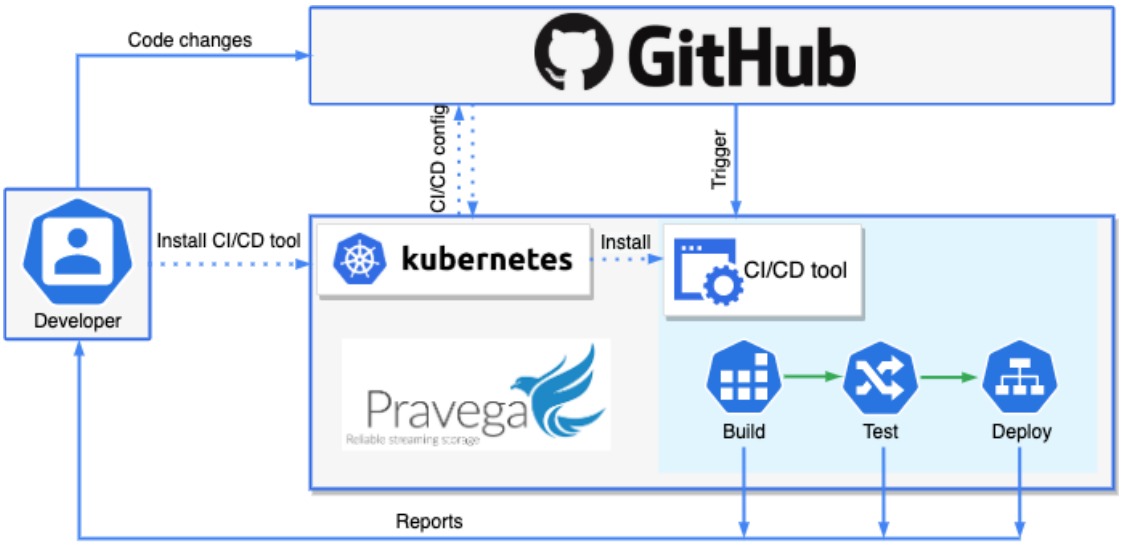


Рисунок 1 – Схема работы разрабатываемой системы.

В рамках данной работы решаются задачи автоматического конфигурирования системы непрерывной интеграции согласно принципу configuration as code [7]. Производится установка и настройка выбранного инструмента и необходимых для корректной работы расширений и плагинов. Также настраивается взаимодействие с имеющейся инфраструктурой и системой контроля версий. Проведение сборки приложения и его тестирование затрагивается только в качестве проверки работоспособности разрабатываемой системы.

В качестве приложения, иллюстрирующего работу SDP, используется приложение по анализу тональности текста с помощью алгоритмов машинного обучения.

ЛИТЕРАТУРА

1. State of the cloud report 2019 [Электронный ресурс]. URL: https://resources.flexera.com/web/media/documents/rightscale-2019-state-of-the-cloud-report-from-flexera.pdf?elqTrackId=372b6798c7294392833def6ec8f62c5c&elqaid=4588&elqat=2&_ga=2.75255952.381106268.1556868633-1445624021.1556868633 (Дата обращения 13.03.2021)
2. The cloud delivered for over 90 per cent of UK plc when tackling the challenge of COVID [Электронный ресурс]. URL: https://www.cloudindustryforum.org/content/cloud-delivered-over-90-cent-uk-plc-when-tackling-challenge-covid (Дата обращения 11.03.2021)

3. 2021 predictions from Capgemini: The rise of cloud economics [Электронный ресурс]. URL: https://cloudcomputing-news.net/news/2021/jan/05/2021-predictions-from-capgemini-the-rise-of-cloud-economics (Дата обращения 12.03.2021)
4. 10 key attributes of cloud-native applications [Электронный ресурс]. URL: https://thenewstack.io/10-key-attributes-of-cloud-native-applications/ (Дата обращения 13.03.2021)
5. Dell EMC Streaming Data Platform [Электронный ресурс]. URL: https://www.delltechnologies.com/en-us/storage/streaming-data-platform.htm (Дата обращения 13.03.2021)
6. 10 Best Continuous Integration Tools For 2020 [Электронный ресурс]. URL: https://dzone.com/articles/10-best-continuous-integration-tools-in-2020 (Дата обращения 13.03.2021)
7. Configuration as Code: What is it and how is it beneficial? [Электронный ресурс]. URL: https://octopus.com/blog/configuration-as-code-what-is-it-how-is-it-beneficial (Дата обращения 11.03.2021)

УДК 004.416.6

Титова А.В. (2 курс магистратуры)
Маятин А.В., к.п.н., доцент

МОДИФИКАЦИЯ ЯДРА ВИРТУАЛИЗАЦИИ МАГНИТНЫХ ЛЕНТ ДЛЯ НОВОЙ МОДЕЛИ ПРОДУКТА DISK LIBRARY FOR MAINFRAME

Disk Library for mainframe - виртуальная библиотека магнитных для мейнфрейма, которая является посредником между мейнфреймом и хранилищем данных. С его помощью мейнфрейм пишет данные и читает их с эмулированных магнитных лент, которые на самом деле являются фалами на современных СХД производства Dell Technologies. Продукт имеет долгую и богатую историю, постепенно росло количество дорогостоящих функциональностей, а вместе с ними росла и стоимость продукта. В связи с прекращением поддержки “базовой” модели, DLm начал терять сегмент пользователей, что сказалось на позиции продукта на рынке. Чтобы исправить ситуацию, было принято решение создать новую “базовую” модель – DLm2500. Эта модель будет обеспечивать самую минимальную функциональность. При этом, данная модель возьмет за основу современную кодовую базу, которая войдет в дальнейшие “полноценные” релизы и модели, чтобы код был максимально безопасным и легко поддерживаемым.

При разработке новой модели перед разработчиками ядра виртуализации магнитных лент были поставлены следующие задачи:

1. Отключить всю дополнительную функциональность продукта, удостовериться, что она не работает (миграция данных на “холодную” СХД, шифрование, поддержка работы с физическими магнитными лентами)
2. Удостовериться, что вся базовая функциональность продолжает работать (подразумеваются манипуляции с магнитными лентами: инициализация, чтение, запись, затирание, отмена затирания, удаление, поддержка различных параметров этих действий, конфигурирование системы в целом)
3. Реализовать определенный набор “ограничений” (удешевление стоимости продукта происходит не только за счет ограничения функциональностей, но и за счет удешевления аппаратного обеспечения, что порождает ограничения по памяти и максимальной нагрузке)
4. Реализовать новую функциональность (механизм контроля и регуляции ёмкости используемой системы хранения данных)
5. Обеспечить поддержку нескольких новых типов СХД (эмпирическое выявление оптимальных параметров работы с СХД и возможность их изменения)

Так как продукт дорогостоящий и напрямую работает с данными клиентов, среди которых и банки, и авиаперевозчики, и медицинские учреждения, и многие другие, при всех

этих манипуляциях необходимо в первую очередь обеспечивать безопасность данных и надежность кода. Кроме того, продукт существует уже много лет, поэтому необходимо также модернизировать код с целью повышения его читаемости и использования более современных и надежных технологий, то есть производить рефакторинг.

Так как было принято решение использовать одну и ту же кодовую базу и для “базовой” модели, и для обычной, первые две задачи были решены добавлением модуля определения доступности различных функциональностей в зависимости от модели продукта.

Третья задача вылилась в исследование механизмов использования памяти ядром виртуализации, и оказалось, что оба ограничения можно реализовать с помощью одного и того же кода – необходимо лишь ограничить максимальное количество виртуальных устройств работы с виртуальными же магнитными лентами. Это уменьшит количество статически выделяемых структур, уменьшит потенциальную “распараллеленность” программы, а значит и ресурсов, выделяемых на это как с точки зрения памяти, так и с точки зрения нагрузки на систему.

Четвертая задача была наиболее интересной с точки зрения разработки – необходимо было спроектировать механизмы подсчета текущей ёмкости СХД, периодической проверки текущей ёмкости, принятия решения о начале мероприятий по восстановлению ёмкости, оповещения пользователя о достижении критической ёмкости, прекращения возможности записи новых данных, если ёмкость превышает второе критическое значение и, собственно сами мероприятия по восстановлению ёмкости. Также необходимо было обеспечить конфигурирование таких параметров, как периодичность проверок ёмкости, оба критические значения ёмкости, максимальная ёмкость, восстанавливаемая за одну попытку.

Пятая задача потребовала глубокого погружения в код и рефакторинга методов хранения параметров взаимодействия с СДХ, что привело, в первую очередь, к повышению читабельности кода. В результате рефакторинга добавление поддержки новых типов СХД стало занимать почти в 9 раз меньше строк кода по сравнению с теми же манипуляциями до рефакторинга.

В процессе тестирования модели на виртуальных СХД (так случилось из-за опоздания поставки настоящих СХД в лабораторию) было найдено слабое место ядра виртуализации, а именно – наличие большого количества “долгих” блокировок. Из-за несовершенства реализации системных вызовов в виртуальных СХД, долгие блокировки над системными вызовами приводили к большому количеству ошибок из-за превышения временных лимитов на те или иные действия. Выяснилось, что из-за привычки разработчиков взаимодействовать с быстрыми СХД, появилась тенденция расставлять блокировки на слишком большое количество операций. Эта проблема потребовала пересмотра использования блокировок, и в результате даже на “долгих” виртуальных СХД продукт стал работать без ошибок. В перспективе эта проблема могла проявиться гораздо позже и могла потребовать гораздо большего анализа и количества изменений в коде.

В результате была спроектирована и реализована новая “базовая” модель продукта Disk Library for mainframe, которая реализует базовую функциональность виртуальной библиотеки магнитных лент, предоставляет механизмы автоматического восстановления ёмкости СХД, является менее требовательной к аппаратному обеспечению, и решает бизнес-задачу по возвращению утерянного сегмента пользователей.

ЛИТЕРАТУРА

1. Dell EMC/ Disk Library for mainframe release 5.1 spec sheet, 2019. URL: https://www.dellemc.com/en-us/collaterals/unauth/data-sheets/products/storage/h5937-emc-disk-lbry-mf-ss.pdf
2. Dell EMC/ Disk Library for mainframe DLm8500 release 5.1 spec sheet, 2019. URL: https://www.dellemc.com/en-us/collaterals/unauth/data-sheets/products/storage/h4207-disk-library-mainframe-ds.pdf

3. HG Insights/ Dell EMC Disk Library for mainframe. URL: https://discovery.hgdata.com/product/dell-emc-disk-library-for-mainframe
4. Фаулер, Мартин. Рефакторинг: улучшение существующего кода [текст] / М. Фаулер. – СПб: Символ-Плюс, 2003. – 432 с.
5. Williams, Anthony D. C++ Concurrency in Action [текст] / Anthony D. Williams, 2011 – 325 с.

УДК 004.4

Хвецкович А. (4 курс бакалавриата)
Маслаков А. П., ст. преподаватель

НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ ГРАФОВЫХ БАЗ ДАННЫХ

Графовые базы данных (БД) являются новой разновидностью баз данных, реализующую сетевую модель в виде графа и его обобщения [1]. Как правило, они используются в биоинформатике, социальных сетях, поисковых роботах. Поскольку в данных областях крайне важна производительность используемых решений, перед разработчиками остро стоит вопрос о быстродействии графовых БД. Поэтому актуальной задачей, решенной в данном исследовании, является оценка производительности различных реализаций графовых БД и стэков для работы с ними.

Для исследования производительности были использованы данные футбольного турнира под названием «Английская Премьер-Лига» за 2020 год [2]. Они включают в себя турнир, состоящий из 38 туров, в каждом из которых по 10 сыгранных игр между 20 командами. Всего в БД содержится 439 записей, суммарно данные весят 153 КБ. Данные представлены в виде графа, краткая схема которого может быть изображена на рисунке 1.

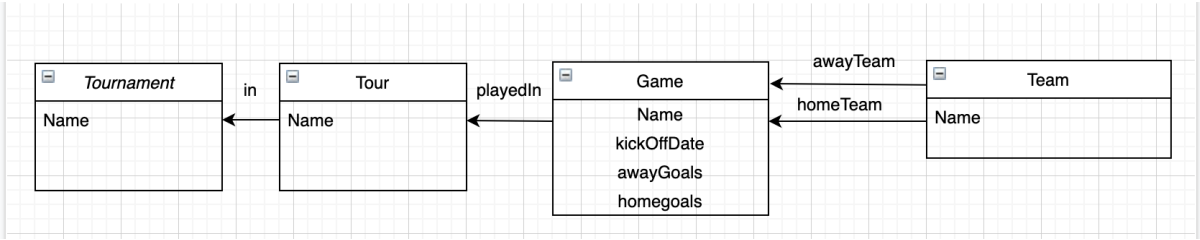


Рисунок 1 - Схема графа БД.

Для каждой команды создаётся отдельная вершина, то есть 20 вершин с типом «Team». Аналогично, для каждой из 380 сыгранной в турнире игры создаётся своя вершина с типом «Game». Также создаётся 38 вершин с типом «Tour» для сыгранных туров, и одна вершина с типом «Tournament», обозначающая сам турнир. Все вершины характеризуются свойством «Name». Вершины с типом «Game» дополнительно содержит свойства «homeGoals», «awayGoals» и «kickOffDate», хранящими сведения о количестве забитых голов домашней командой, гостевой командой и о дате игры соответственно.

Таким образом, вершины с типом «Team» соединены с вершинами с типом «Game» рёбрами с возможными типами «AwayTeam» и «HomeTeam» (в зависимости от того, играла ли команда данную игру «дома» или «в гостях»), при этом вид связи один-ко-многим, то есть одна команда играет во многих играх. В свою очередь вершины с типом «Game» соединены рёбрами с вершинами с типом «Tour», связь один-к-одному, так как все игры уникальные и принадлежат только одному конкретному туру. Вершины с типом «Tour» соединены с вершиной «Tournament» (в нашем проекте такая вершина единственная) связью один-к-одному.

В качестве движка графовой БД были выбраны следующие: JanusGraph[3], Neo4j[4], AWS Neptune[5] и Tigergraph[6]. Для JanusGraph дополнительно были протестированы разные хранилища бэкенда (Backend Storage).

Тестирование осуществлялось на окружениях, продемонстрированных на рисунке 2

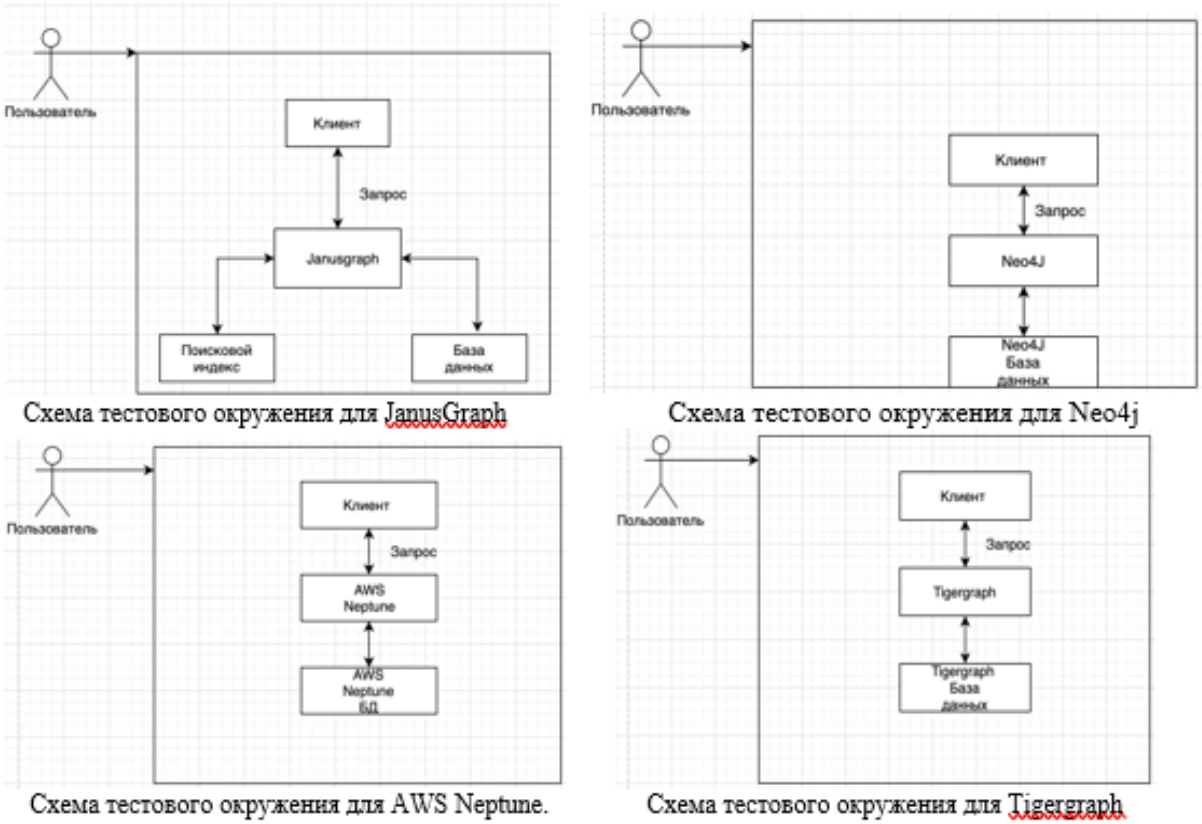


Рисунок 2 - Схемы тестовых окружений БД.

Запросы, которые позволили измерить время исполнения запросов для БД Tigergraph были написаны на языке GSQL [7]. Для всех остальных представленных БД использовался язык обхода графов Gremlin [8]. Глядя на то, как быстро система обрабатывает самый сложный запрос — вывод всего турнира, можно судить о быстродействии всей системы, т.к. этот запрос учитывает в себе все возможные связи. Также тестировалось время выполнения запроса по выборке одной вершины по её ID. Но так как вывод одной вершины во всех БД отличается тривиальностью, то обращаем внимание на более сложный запрос.

Запрос на языке Gremlin:

g.V().hasLabel("tournament").in("in").in("played").in("homeTeam").valueMap().path()

В данном запросе выводится весь турнир.

В Таблице 1 представлены таблица с результатами тестирования.

Таблица 1 - Результаты тестирования.

Запрос	**Neo4j, сек**	**Neptune, сек**	**JanusGraph (Cassandra, ES в рамках одной транзакции), сек**	**JanusGraph in-memory, сек**	**JanusGraph Berkley (Docker), сек**	**JanusGraph (Cassandra, ES), сек**	**Tigergraph, сек**
Вывод всего турнира	5,465	6,657	7,779	5,318	5,473	**5,155**	14,243
Вывод всех домашних команд турнира	6,112	6,479	1,371	5,319	5,461	**5,414**	13,569

Запрос	Neo4j, сек	Neptune, сек	JanusGraph (Cassandra, ES в рамках одной транзакции), сек	JanusGraph in-memory, сек	JanusGraph Berkley (Docker), сек	JanusGraph (Cassandra, ES), сек	Tigergraph, сек
Вывод всех выездных команд турнира	6,007	7,054	0,051	5,287	5,429	**5,447**	14,216
Счет общего количества всех команд турнира	5,294	6,402	0,034	5,348	5,389	**5,306**	15,899
Вывод всех игр определенного тура	5,288	5,115	0,164	5,296	5,38	**5,331**	14,378
Вывод домашних команд определенного тура	5,317	5,106	0,038	5,315	5,389	**5,561**	14,789
Вывод времени выездных игр клуба	5,766	5,085	1,291	5,292	5,584	**5,742**	14,678
Вывод времени выездных игр клуба	5,701	5,084	0,041	5,305	5,657	**5,437**	14,899
Вывод вершины по ID	0,01	0,005	0,01	0,01	0,011	**0,01**	14,754

По результатам проведённого исследования, на основе Таблице 1, в качестве оптимальной конфигурации была выбрана связка графовой БД JanusGraph вместе с хранилищем бэкенда Cassandra и поисковой системой Elasticsearch.

ЛИТЕРАТУРА

1. Графовая база данных. Электронный ресурс: https://ru.wikipedia.org/wiki/ Графовая_база_данных (дата обращения: 21.12.2020)
2. Компания “ОККО”. Электронный ресурс: https://okko.sport/tournament/epl_2020/calendar/tour (дата обращения: 21.12.2020)
3. База данных JanusGraph. Электронный ресурс: https://janusgraph.org/ (дата обращения: 21.12.2020)
4. База данных Neo4j. Электронный ресурс: https://neo4j.com/ (дата обращения: 21.12.2020)
5. База данных AWS Neptune. Электронный ресурс: https://pages.awscloud.com/AWS-Learning-Path-Getting-Started-with-Amazon-Neptune_2020_LP_0009-DAT.html (дата обращения: 21.12.2020)
6. База данных TigerGraph. Электронный ресурс: https://www.tigergraph.com/ (дата обращения: 21.12.2020)
7. Язык запросов GSQL. Электронный ресурс: https://www.tigergraph.com/gsql/ (дата обращения: 21.12.2020)
8. Язык запросов Gremlin. Электронный ресурс: https://tinkerpop.apache.org/ (дата обращения: 21.12.2020)

УДК 004.087.2

Юрков А.А. (2 курс магистратуры)
Петров А.В., ст. преподаватель

МОНИТОРИНГ ЗДОРОВЬЯ ЖЕСТКИХ ДИСКОВ

Целью данной работы является разработка модели мониторинга здоровья жестких дисков. Данная модель должна оценивать отклонение или деградацию устройства от его нормального состояния путем сопоставления данных датчиков со статистическими данными. Такой подход дает возможность обнаруживать аномальное поведение устройств в онлайн режиме. Это используется для предупреждения пользователя о надвигающихся сбоях, а также для предоставления информации экспертам для диагностики.

Подход машинного обучения (ML) к проблеме прогнозирования выхода из строя диск заключается в изучении вероятностных характеристик рабочих и приводных дисков, которые скоро выйдут из строя. Учитывая выборoку, производится прогноз, столкнется ли данный накопитель с неизбежным отказом. Рассматриваемая проблема может формулироваться как задача классификации, математически сформулированная следующим образом: с учетом последовательности ввода *{xi,1;xi,2;...,xi,T}* предсказывают одну метку yi, которая применяется ко всей последовательности. Обучение модели производиться на примере пар *(XI, YI)*, где *xi* – последовательность вектор атрибутов SMART, собранных с диска i в разное время t, а yi - логическое значение, обозначающее, вышел ли диск из строя.

Подготовка данных:

Набор данных BACKBLAZE, используемый для этого исследования, содержит SMART-измерения, собираемые ежедневно в течение 1 года с примерно 50 000 накопителей в центре обработки данных. Поскольку каждый производитель накопителей собирает различные атрибуты SMART, разработанная модель применялась только к накопителям Seagate, которые составляют почти 50% от общего числа накопителей, но ее можно легко изменить для применения к накопителям других типов.

Особенности данных и распределения:

Данные содержат ~ 100 отдельных необработанных и нормализованных атрибутов SMART с добавлением еще нескольких основных сведений о накопителе, таких как модель накопителя, серийный номер, температура, кол-во ошибок чтения и т.д. Как только исходное значение одного из этих атрибутов SMART отличается от нуля, BACKBLAZE выдает предупреждение для проверки диска. Для этого исследования используются только исходные значения SMART (которые оказались более информативными, чем нормализованные значения), большинство вышеупомянутых атрибутов тесно связаны с отказами дисков.

Подготовка данных для алгоритмов машинного обучения:

Перед применением ML важно правильно подготовить данные. В данном случае нельзя просто задать (x1,x2,…,xk, y), где вектор x это значения всех SMART атрибутов, а y - кол-во дней до выхода из строя. В таком виде кол-во дней никак не зависит от динамики изменения каунтеров, т.к. вектор x не содержит в себе такой информации.

Метод выявления динамических характеристик:

В данной работе предлагается подход к ML где обучение происходит на динамических характеристиках за определенные период времени. К числу агрегатных функций, выбранных для представления поведения накопителей, относятся характеристики линейной регрессии, подходящей для каждого атрибута в непрерывном временном окне выбранного количества дней. Простое соответствие линейной регрессии - это уравнение прямой линии: $y = \alpha + \beta x$, которое наилучшим образом соответствует значениям атрибутов как функции времени. В качестве признаков для модели мы используем рассчитанные параметры β и α (наклон регрессии и пересечение соответственно) и СКО, которые являются нормализованной мерой того, насколько хорошо модель соответствует данным. Это выполняется для каждого

временного окна. Кроме того, дисперсии, максимальное значение, разница между максимальным и минимальным значением и максимальным значением между последовательными выборками используются в качестве составных элементов для модели классификации для обучения.

Для демонстрации был выбран атрибут smart_5 снятый с двух дисков.

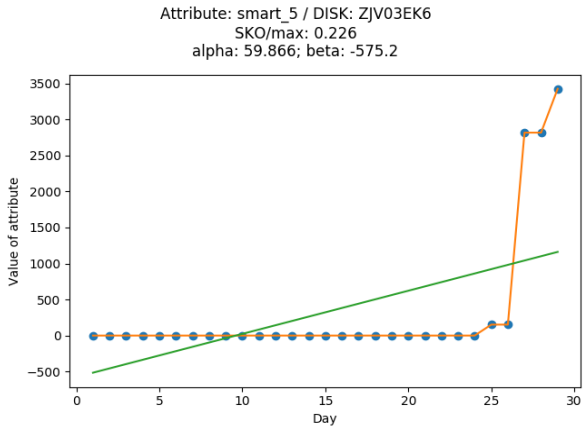


Рисунок 1 - Пример несоответствия кривой и тренда.

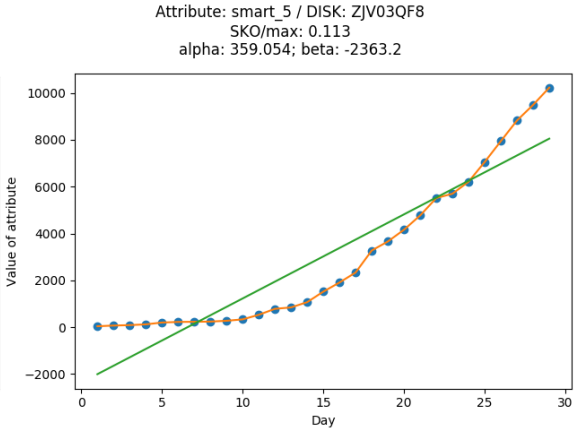


Рисунок 2 - Пример соответствия кривой и тренда.

Можно заметить, что при переходе от статистических данных к данным, отражающим динамику, не достаточно только значений линии тренда, поэтому вводится третий параметр - СКО деленный на максимальное значение в данном окне, для нормализации, т.к. абсолютное значение не представляет практического интереса.

Одной из основных проблем при проведении прогнозного анализа является низкая доля неисправных. Это сложная задача в двух аспектах: во-первых, вероятностная модель имеет только несколько положительных примеров, и не все из них представляют аномальное количество атрибутов ошибок, например, разные диски могут выйти из строя по разным причинам. Во-вторых, низкое количество неисправных накопителей повышает эффективность ложных срабатываний, что требует чрезвычайно точных результатов от реализованной модели.

Для этой работы использовался набор данных с открытым исходным кодом от BACKBLAZE, включающий обширную SMART-информацию, ежедневно собираемую из большого числа пользователей. Это позволило использовать не только последние значения SMART, но и включать новые функции, полученные за разные промежутки времени последних дней работы накопителей. Эти составные функции отражают динамику атрибутов накопителей, например, скорость их роста, что было крайне важно для прогнозирования отказов накопителей самим BACKBLAZE в 2016 году.

ЛИТЕРАТУРА

1. Описание значений SMART атрибутов [Электронный ресурс]. - Режим доступа: https://www.ixbt.com/storage/hdd-smart testing.shtml#n5
2. Официальный сайт BACKBLAZE [Электронный ресурс]. - Режим доступа: https://www.backblaze.com
3. Официальный сайт HDDScan [Электронный ресурс]. - Режим доступа: http://hddscan.com/
4. Статистические данные BACBLAZE [Электронный ресурс]. - Режим доступа: https://www.backblaze.com/b2/hard-drive-test-data.html
5. Технология SMART [Электронный ресурс]. - Режим доступа: https://rlab.ru/doc/s.m.a.r.t..html

СОДЕРЖАНИЕ

Приветствие от Санкт-Петербургского Центра Разработок Dell Technologies 3

Приветствие от компании EPAM 4

Информационное сообщение 5

Тезисы докладов конкурса-конференции 10

Секция Программная инженерия: приложения, продукты и системы 10

Алейников П.И., Рамейков А.С., Фёдоров С.А. Система прогнозирования выработки солнечной электростанции 10

Алексеева М.В., Тимофеев Д.А., Самочадин А.В. Построение статистического парсера CCG для русского языка 11

Булатов Д.Е., Леонтьева Т.В. Экосистема умного дома на базе ESP8266 13

Власенко Н. А., Дусаева А. И., Эйзенах Д. С., Маслаков А. П. Реализация системы измерения размеров объектов с использованием технологий компьютерного зрения 15

Власов Д.В., Иванов Г.В., Околович Л.Д., Никифоров И.В. Реализация отзывчивого web-интерфейса для системы транскодирования видеофайлов и видеопотоков 17

Губенко Н.О., Коликова Т. В. Федеративное обучение и частное масштабирование 19

Власов М.О., Молодяков С.А. Электронная стабилизация видео на смартфонах 21

Думаев Р.И., Молодяков С.А. Автоматическое обнаружение COVID-19 по рентгеновским снимкам 22

Загороднев А.А., Приступ М.В., Русков М.О., Маслаков А.П. Разработка онлайн платформы для менеджмента гонок 25

Зайцев Д.И., Пиккио П.Ф., Графеева Н.Г. Расширение функциональности Android приложения 27

Иванов Д.А., Петров А.В., Каплан Е.В. Разработка системы управления умным домом с помощью мобильного приложения и системы автоматизации Home Assistant 29

Иванов В.О., Малеев О.Г. Глубокое обучение нейронных сетей для распознания изображений.. 31

Имхасина И.Х., Самочадин А.В. Разработка приложения для любительского наблюдения за птицами 33

Катричко У.Д., Литвинов Ю.В. Реализация генератора распоряжений о распределении студентов на дисциплины по выбору 34

Кауана Хил Л. А., Коликова Т. В. Приложение для прогнозирования поведения фондовых рынков с использованием нейронных сетей 36

Клименко М.А., Коликова Т.В. Мобильное приложение под операционную систему Android для поддержания здорового образа жизни 37

Ладыгина П.А., Самочадин А.В. Разработка программного средства для прогнозирования прочности бетона с применением методов машинного обучения 37

Коновалов И.И., Воинов Н.В. Разработка интерактивного помощника для консультирования учащихся в задачах лингводидактики 39

Лебедев Д.М., Устинов А.Г., Эйзенах Д.С., Маслаков А.П. Мобильное приложение под операционную систему IOS направленное на развитие когнитивных способностей 41

Лукин А.А., Мироненков Г.В., Воинов Н.В., Каплан Е.В. Программно-аппаратный комплекс контроля качества пищевой продукции 42

Лелюхин Д.О., Федоров С.А. Разработка ПО в виде сервиса для управления камерами машинного зрения Ximea xiX 44

Маликов А. Т., Филиповский В. М. Разработка программы построения генеалогических деревьев на языке C++ в среде Microsoft Visual Studio 45

Небогатиков И.Ю., Соловьев И.П. Система контроля продолжительности действий обитателя в среде умного дома 47

Околович Л.Д., Власов Д.В., Иванов Г.В., Никифоров И.В. Разработка модуля транскодирования видеопотоков в режиме реального времени с использованием библиотеки ffmpeg 48

Орлов В. В., Коликова Т. В. Мобильное приложение под операционную систему Android для планирования и организации путешествий 49

Неверовский Н. А., Никифоров И. В. Метод построения обогащенного метаданными изображения с использованием REST-микросервисов 50

Письмерова В.А., Черноруцкий И.Г. Разработка системы поддержки принятия решения по торговле ценными бумагами 53

Ромицына М. С., Петров О. Н. Применение технологий искусственного интеллекта в задачах классификации сигналов 55

Полетова Н.В., Дробинцев Д.Ф. Рекомендательная система рецептов блюд на основе пищевого контекста 56

Скоков Н.С., Пшеничная К.В. Выбор средств нейтрализации атак вида 58

Стариков А.О., Хмыров И.В., Маслаков А.П. Разработка веб-приложения для тестирования торговых стратегий на архивных данных 60

Суслов К. А., Шаповал П. В., Коликова Т. В. Разработка мобильного приложения для страхования транспортных средств под Android ОС и iOS 62

Сясько М.В., Соловьев И. П. Расчёт и визуализация траекторий движения заряженных частиц в низкочастотных электромагнитных полях 65

Сиверин Д.А., Самочадин А.В. Приложение для классификации и оценки объема материала 66

Титов Ф.М., Самочадин А.В. Разработка программного средства протоколирования совещаний 67

Шаповалова И.А., Коликова Т.В. GraphQL - альтернатива REST в разработке web API 69

Томилин И. С., Фёдоров С. А. Метеостанция с возможностью удаленного управления и мониторинга 72

Шевченко А.Г., Тышкевич А.И. Распределенные системы технологического обслуживания и управления конфигурациями специализированных радиостанций 74

Шульц С. В., Петров О. Н. Исследование применения технологий смешанной реальности в задачах исследовательского проектирования 75

Секция Программная инженерия: инструментальные средства и технологии проектирования и разработки 78

Александров А. В., Сениченков Ю. Б. Модель оптимизации светофорных фаз 78

Афанасов А.К., Литвинов Ю.В. Извлечение данных Android-приложения WhatsApp 79

Афанасов А.К., Богданов Е.А., Баклановский М.В. Реализация асимметричного маркерного процессинга на C66X DSP и ARM Cortex-A15 81

Барсуков Н.Д., Посметныйс Д., Никифоров И.В. Распределенная микросервисная архитектура системы анализа логов платформы 83

Бетуганов А.А., Хомяков А.А., Сараджишвили С.Э. Инструментальные средства мобильной разработки 86

Васеха И. А., Петров А. В. Разработка инструмента для генерации mock-сервисов на основе AsyncAPI спецификации 88

Васильев Д. С., Маслаков А. П. Разработка RESTful API на основе языка Python и фреймворка Flask 90

Воскресенский А.С., Семёнова-Тян-Шанская В.А. Использование алгоритмов машинного обучения при защите от скликивания 92

Голубева К.А., Самочадин А.В. Разработка приложения для контроля посещаемости с использованием модуля Bluetooth 93

Доленко Д.В., Шарапов А.А., Коликова Т.В. Разработка программы для синхронизации контента между устройствами в рамках локальной сети 95

Егоров М.Э., Леонтьева Т.В. Разработка приложения для звонков под iOS на основе WebRTC . 97

Гусев И. С., Коликова Т. В. Разработка мессенджера с возможностью обмена геопозицией на базе микросервисной архитектуры 98

Ерохин Я.Д., Дробинцев П.Д. Сравнительный анализ баз данных для ведения журнала устройств.. 99

Засядько В.В., Воинов Н.В., Прокофьев О.В. Веб-сервис для автоматизации процессов в отделе маркетинговых исследований 101

Козельский Е.И, Шаляпин В.В. Генерация информативных вопросов в информационно-асимметричных беседах 102

Кущенко А. Е., Самочадин А. В. Разработка инструмента для извлечения моделей процесса разработки программного обеспечения 103

Лебедев И.В., Медведев Б.М. Разработка демонстрационных проектов для устройства интернета вещей на базе Arduino 105

Лысенко В.В., Сташевский Я. В., Петров А.В. Прототип Telegram-Bot PolySchedule 107

Марченко В.А., Хватов С.А., Воскобойников С.П. Разработка и оптимизация алгоритма сравнения геномов 109

Мухин Д.А., Горавнева Т.С. Исследование уязвимостей информационных систем предприятий и методов обеспечения их безопасности 110

Новоселова Е.И., Самочадин А.В. Разработка средств оценки трудозатрат на реализацию программных продуктов на базе различных методик 112

Пиккио П. Ф., Зайцев Д. И., Графеева Н. Г. Проектирование консольной игры жанра RogueLike 114

Поликарпова А. И., Самочадин А. В. Разработка системы по расчету корреляции между уровнями свойств внимания и частотой морганий 116

Пости И. М., Танков В. Д. Инкрементальный анализ и представление Markdown документов в Markdown plugin для IntelliJ IDEA 118

Станкевичуте Е., Воинов Н.В., Леонтьева Т.В. Разработка Telegram-бота для Спортивного клуба 120

Стягов А. Ю., Маслаков А. П. Перспективы декларативного подхода к созданию пользовательских интерфейсов в Android 122

Талалова Е.В., Тышкевич А.И. Постобработка радиолокационных данных для улучшения качества распознавания в поле обзора радиолокационного детектора 124

Усов М.А., Воинов Н.В., Зайцев И.В. Разработка Android-приложений на основе декларативного подхода с применением Jetpack Compose 126

Фигловский К. С., Никифоров И. В., Юсупова О.А. Использование Gradle build cache для оптимизации времени сборки 127

Фролова А.Ю., Никифоров И.В. Автоматизация сбора и расчета рекламных метрик в программном средстве на основе данных внешних источников 129

Ченский Т. С., Семенова-Тян-Шанская В.А. Сравнение инструментальных средств и алгоритмов кластерного анализа данных 131

Шапиев Т.К., Дробинцев Д.Ф Telegram Bot для поиска фильмов 133

Секция Программная инженерия: методы и алгоритмы теории программирования 135

Алексеева Д.В., Никифоров И.В. Оптимизация нагрузки мультифрактального трафика между гетерогенными узлами распределенной вычислительной системы 135

Абрамова Т.О., Тышкевич А.И. Совершенствование методов цифровой обработки сигналов для радионавигационных систем, применяемых для задач позиционирования 137

Болтунов А.А., Черноруцкий И.Г. Исследование алгоритма стрекозы для решения задач оптимизации..... 138

Варламов Д.А., Шошмина И.В. Моделирование и формальная верификация распределенной транзакционной памяти CUDA-DTM 140

Виноградов П.А., Дробинцев Д.Ф. Прогнозирование изменений цен акций фондового рынка с помощью рекуррентной нейронной сети и анализа тональности текста 142

Ворошилов М.К., Дробинцев П.Д. Прогнозирование тренда котировок финансовых инструментов с помощью машинного обучения 144

Григорьев А.А., Черноруцкий И.Г. Проблема жёсткости на примере алгоритма пчелиной колонии 146

Дроздов С.В., Уткин Л.В. Метод интеллектуальной классификации гистологических изображений с использованием пространственного положения отдельных элементов 148

Друц О.В., Клещёнок Д.И., Семенча В.Е., Муравьев Е.А. Автоматизация синтеза семантических моделей в топологической семантике 150

Дудина А. Н., Черноруцкий И. Г. Сжатие изображений с помощью генеративно-состязательной сети..... 152

Жукова К.А., Черноруцкий И.Г. Исследование работоспособности адаптивного градиентного метода AMSGrad в условиях жесткости целевых функционалов..... 153

Забродина Е.С., Черноруцкий И.Г. Преобразование изображений в сверхразрешение при помощи нейронной сети 155

Загородний Г.Р., Стахиев А.В., Абрамов Н.А. Удаление артефактов видеопленки методом двойной интерполяции кадров 157

Каргалов Л.А., Черноруцкий И.Г. Оптимизация маршрутизации в серверно-ориентрированных сетях 158

Крамаров Е. В., Черноруцкий И. Г. Применение гибридного алгоритма WOAGWO для задачи кластеризации 160

Лаппо С.С., Петров А.В. Адаптивная система планирования времени 161

Иванов Г.А., Королев К.И., Семенча В.Е., Муравьев Е.А. Сравнение спецификаций и программ в топологической модели семантики..... 163

Масленникова А.А., Черноруцкий И.Г. Исследование и сравнительный анализ алгоритма ADAM в моделях глубокого обучения 164

Мордвинцева А.И., Черноруцкий И.Г. Применение алгоритма социального паука для решения задач оптимизации..... 166

Русаков Е. С., Черноруцкий И. Г. Анализ спутниковых снимков городов с помощью свёрточных нейронных сетей 169

Сабуткевич А.М., Самочадин А.В. Программные средства организации контрольных тестирований на основе методики адаптивного обучения 170

Сатюков Е. Д., Уткин Л. В. Разработка и исследование метода объяснения предсказаний модели машинного обучения с цензурированными данными с использованием нейронных сетей 172

Семакина Д.Н., Черноруцкий И.Г. Исследование эффективности метода поиска гармонии для нахождения экстремумов жестких целевых функционалов 174

Соколова А. А., Черноруцкий И. Г. Исследование алгоритма светлячков для решения задач оптимизации 176

Стояноски К., Дробинцев П.Д. Готовые решения для использования машинного обучения в мобильных приложениях с использованием Firebase ML Kit 178

Султанова Д. В., Черноруцкий И. Г. Исследование алгоритма летучей мыши для решения задач оптимизации 179

Шалгуева С. Л., Воскобойников С. П., Попов Е.Н. Разработка программы для моделирования взаимодействия электромагнитных рассеивателей 181

Шумковская Э.О., Черноруцкий И.Г. Реализация улучшенного китового алгоритма оптимизации и исследование его эффективности на жестких целевых функционалах 182

Юсупова О. А. Программная реализация и анализ метода определения доминирующей ЦБ для управления портфелем ЦБ 183

Секция Технологии обработки и анализа больших массивов данных на основе работ EPAM 185

Елисеев П.В., Петров О.Н. Интегрированная система контроля сервиса 185

Златоустов В.А., Горавнева Т.С., Пшеничная К.В. Разработка приложения для функционирования 187

Конкин В.В., Субцельная О.Д. Разработка системы умной парковки 189

Парахин К. А., Никифоров И. В. Разработка гибкой и высокопроизводительной платформы для анализа машинных данных 191

Романов К.М., Никифоров И.В., Леонтьева Т. В. Программный сервис автоматического определения ошибок в пользовательских сессиях кроссплатформенного приложения 193

Сафронов Д., Соломонов М.С., Стоноженко К.М., Никифоров И.В. Расширение функционала электронной библиотеки СПбПУ за счет внедрения модуля анализа данных 196

Шевцов Д. Ю., Никифоров И. В., Метод автоматизации процесса изучения информации за счет применения подхода инкрементального чтения 198

Эйзенах Д.С., Рамейков А.С., Никифоров И.В. Высокопроизводительная система извлечения данных с веб-ресурсов 200

Секция Подходы к разработке программного обеспечения на основе технологий Dell Technologies 203

Акназарова А. И., Никифоров И. В., Прокофьев О. В. Высокопроизводительная гибкомасштабируемая архитектура программного средства для обработки данных в режиме реального времени 203

Аникин Н.В., Медведев Б.М. Создание программного комплекса сбора и хранения статистики сетевого трафика промышленного назначения 205

Забелин К.В., Максимчук В.А., Шемякинская А.С., Никифоров И.В. Автоматизация сборки и развертывания системы анализа данных электронной библиотеки СПбПУ 207

Иванов Г.В., Власов Д.В., Околович Л.Д, Никифоров И.В. Проектирование REST-интерфейса системы транскодирования видеопотоков в режиме реального времени 209

Кольцов А.А., Никифоров И.В. Автоматизация определения регрессии производительности СХД 211

Петраченко Я. И., Котлярова Л. П. Сравнительный анализ интерфейсов командной строки для управления системами хранения данных компании Dell Technologies 214

Петров А. А., Никифоров И. В., Зайцев И. В. Балансировка данных для программы объединения систем хранения данных 216

Ревтова Н.Д., Котлярова Л.П. Разработка фреймворка для адаптивной конфигурации внутренних параметров СХД с целью повышения производительности 218

Саравайский М. А., Маятин А. В. Проектирование и реализация модификации продукта Disk Library for mainframe для обеспечения эффективного обмена данными между IBM System Z и другими системами 219

Тимошенский Г. Д., Котлярова Л. П. Исследование подходов и реализация автоматической конфигурации системы непрерывной интеграции для полностью облачных приложений..... 221

Титова А.В., Маятин А.В. Модификация ядра виртуализации магнитных лент для новой модели продукта Disk Library for mainframe 223

Хвецкович А., Маслаков А. П. Нагрузочное тестирование графовых баз данных 225

Юрков А.А., Петров А.В. Мониторинг здоровья жестких дисков 228

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ В ТЕОРИИ И ПРАКТИКЕ ПРОГРАММИРОВАНИЯ

Сборник материалов конференции

22 апреля 2021 года

Налоговая льгота – Общероссийский классификатор продукции
ОК 005-93, т. 2; 95 3004 – научная и производственная литература

Подписано в печать 19.04.2020. Формат 60×84/16. Печать цифровая.
Усл. печ. л. 14,75. Тираж 200. Заказ 1784.

Отпечатано с готового оригинал-макета, предоставленного редколлегией,
в Издательско-полиграфическом центре Политехнического университета.
195251, Санкт-Петербург, Политехническая ул., 29.
Тел.: (812) 552-77-17; 550-40-14.