

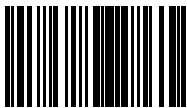
Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
GRADUATION THESIS

Разработка сервиса умного поиска бизнес-процессов с использованием методов
машинного обучения для 1С "Центр Управление Предприятием"

Обучающийся / Student Юрпалов Сергей Николаевич
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет информационных технологий и программирования
Группа/Group М34051
Направление подготовки/ Subject area 09.03.02 Информационные системы и технологии
Образовательная программа / Educational program Программирование и интернет-технологии 2020
Язык реализации ОП / Language of the educational program Русский
Квалификация/ Degree level Бакалавр
Руководитель ВКР/ Thesis supervisor Кравченко Дарья Андреевна, Университет ИТМО, факультет информационных технологий и программирования, преподаватель (квалификационная категория "преподаватель практики")

Обучающийся/Student

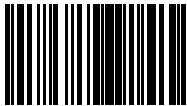
Документ подписан	
Юрпалов Сергей Николаевич	
17.05.2024	

(эл. подпись/ signature)

Юрпалов
Сергей
Николаевич

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Кравченко Дарья Андреевна	
17.05.2024	

(эл. подпись/ signature)

Кравченко
Дарья
Андреевна

(Фамилия И.О./ name
and surname)

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ /
OBJECTIVES FOR A GRADUATION THESIS**

Обучающийся / Student Юрпалов Сергей Николаевич
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет информационных технологий и программирования
Группа/Group М34051
Направление подготовки/ Subject area 09.03.02 Информационные системы и технологии
Образовательная программа / Educational program Программирование и интернет-технологии 2020
Язык реализации ОП / Language of the educational program Русский
Квалификация/ Degree level Бакалавр
Тема ВКР/ Thesis topic Разработка сервиса умного поиска бизнес-процессов с использованием методов машинного обучения для 1С "Центр Управление Предприятием"
Руководитель ВКР/ Thesis supervisor Кравченко Дарья Андреевна, Университет ИТМО, факультет информационных технологий и программирования, преподаватель (квалификационная категория "преподаватель практики")

Характеристика темы ВКР / Description of thesis subject (topic)

Название организации-партнера / Name of partner organization: ООО "БиАйЭй-Технолоджиз"

Тема в области фундаментальных исследований / Subject of fundamental research: нет / not

Тема в области прикладных исследований / Subject of applied research: да / yes

Основные вопросы, подлежащие разработке / Key issues to be analyzed

Основное задание: разработка сервиса умного поиска бизнес-процессов с использованием методов машинного обучения для 1С "Центр Управление Предприятием".

Перечень подлежащих разработке вопросов:

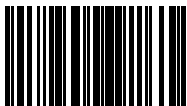
1. Проведение исследования данных
 - 1.1) Исследование исторических данных
 - 1.2) Извлечение пользовательского ввода из шаблонов
 - 1.3) Аугментация данных
2. Проектирование и разработка универсального модуля обработки текстов
3. Проведение исследования моделей машинного обучения
4. Проектирование и разработка программного комплекса

Дата выдачи задания / Assignment issued on: 30.11.2023

Срок представления готовой ВКР / Deadline for final edition of the thesis 17.05.2024

СОГЛАСОВАНО / AGREED:

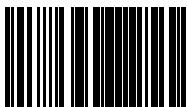
Руководитель ВКР/
Thesis supervisor

Документ подписан	
Кравченко Дарья Андреевна	
11.05.2024	

(эл. подпись)

Кравченко
Дарья
Андреевна


Задание принял к
исполнению/ Objectives
assumed BY

Документ подписан	
Юрпалов Сергей Николаевич	
16.05.2024	

(эл. подпись)

Юрпалов
Сергей
Николаевич

Руководитель ОП/ Head
of educational program

Документ подписан	
Маятин Александр Владимирович	
17.05.2024	

(эл. подпись)

Маятин
Александр
Владимирович

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
SUMMARY OF A GRADUATION THESIS**

Обучающийся / Student Юрпалов Сергей Николаевич
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет информационных технологий и программирования
Группа/Group М34051
Направление подготовки/ Subject area 09.03.02 Информационные системы и технологии
Образовательная программа / Educational program Программирование и интернет-технологии 2020
Язык реализации ОП / Language of the educational program Русский
Квалификация/ Degree level Бакалавр
Тема ВКР/ Thesis topic Разработка сервиса умного поиска бизнес-процессов с использованием методов машинного обучения для 1С "Центр Управление Предприятием"
Руководитель ВКР/ Thesis supervisor Кравченко Дарья Андреевна, Университет ИТМО, факультет информационных технологий и программирования, преподаватель (квалификационная категория "преподаватель практики")

**ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
DESCRIPTION OF THE GRADUATION THESIS**

Цель исследования / Research goal

Разработка сервиса умного поиска бизнес-процессов с использованием методов машинного обучения для 1С "Центр Управление Предприятием"

Задачи, решаемые в ВКР / Research tasks

Изучение предметной области; изучение исторических данных; исследование методов аугментации данных; реализация моделей машинного обучения; тестирование качества моделей машинного обучения; проектирование архитектуры; написание кода модуля обработки текстовых данных; написание кода программного комплекса;

Краткая характеристика полученных результатов / Short summary of results/findings

Универсальный модуль обработки текстов используется во всех исследованиях и проектах команды; Разработанные модели машинного обучения значительно превосходят по качеству предыдущее решение; Разработанный программный комплекс используется продуктовой версией 1С "Центр Управления Предприятием" в "Деловые Линии";

Обучающийся/Student

Документ подписан	
Юрпалов Сергей Николаевич	

Юрпалов
Сергей

Руководитель ВКР/
Thesis supervisor

17.05.2024	
------------	--

(эл. подпись/ signature)

Николаевич

(Фамилия И.О./ name and surname)

Документ подписан	
Кравченко Дарья Андреевна	
17.05.2024	

(эл. подпись/ signature)

Кравченко
Дарья
Андреевна

(Фамилия И.О./ name and surname)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	8
ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ	11
ГЛАВА 1. ПРЕДМЕТНАЯ ОБЛАСТЬ.....	14
1.1 1С: “Центр Управления Предприятием”	14
1.1.1 История развития 1С.....	14
1.1.2 1С: “Центр Управления Предприятием” в Деловые Линии.	15
1.1.3 Интерфейс 1С: “Центр Управления Предприятием”	16
1.1.4 Поиск БП в 1С: “Центр Управления Предприятием”	17
1.2 Постановка задачи	19
1.2.1 Описание задачи от заказчика	19
1.2.2 Тема исследования	19
1.2.3 Цели исследования.....	19
1.2.4 Постановка задачи.....	19
ГЛАВА 2. ИССЛЕДОВАНИЕ ДАННЫХ	20
2.1 Постановка задачи на этап	20
2.1.1 Цель исследования	20
2.2 Исследование исторических данных	20
2.2.1 Описание данных	20
2.2.2 План исследования.....	21
2.2.3 Ход работы.....	22
2.2.4 Выводы и предложения	24
2.3 Извлечение пользовательского ввода из шаблонов 1С: ЦУП ...	26
2.3.1 Описание данных	26

2.3.2 План исследования.....	26
2.3.3 Ход работы.....	27
2.3.4 Выводы	28
2.4 Аугментация данных	29
2.4.1 Описание данных	29
2.4.2 План исследования.....	29
2.4.3 Ход работы.....	29
2.4.4 Выводы	36
ГЛАВА 3. УНИВЕРСАЛЬНЫЙ МОДУЛЬ ОБРАБОТКИ	
ТЕКСТОВЫХ ДАННЫХ	37
3.1 Постановка задачи на этап	37
3.1.1 Цель этапа	37
3.1.2 Функциональные требования	37
3.1.3 Требования к шагам обработки текста	38
3.2 Проектирование	39
3.2.1 Стек технологий	39
3.2.2 Программная архитектура.....	40
3.3 Разработка и тестирование.....	41
3.3.1 Реализация.....	41
3.3.2 Тестирование	41
3.3.3 Выводы	41
ГЛАВА 4. ИССЛЕДОВАНИЕ МОДЕЛЕЙ МАШИННОГО	
ОБУЧЕНИЯ	43
4.1 Постановка задачи на этап	43
4.1.1 Цель исследования	43

4.2 Проведение экспериментов с моделями машинного обучения.	43
4.2.1 Описание задачи.....	43
4.2.2 План исследования.....	44
4.2.3 Ход работы.....	44
4.2.3 Выводы	53
ГЛАВА 5. РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА	54
5.1 Постановка задачи на этап	54
5.1.1 Цель этапа	54
5.1.2 Требования к реализации	54
5.1.3 Функциональные требования	55
5.1.4 Нефункциональные требования	56
5.2 Проектирование	57
5.2.1 Стек технологий	57
5.2.2 Системная архитектура	58
5.2.3 Архитектура данных	59
5.2.4 Программная архитектура.....	60
5.3 Разработка и тестирование.....	61
5.3.1 Реализация.....	61
5.3.2 Тестирование	61
5.3.3 Выводы	62
ЗАКЛЮЧЕНИЕ	63
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	64
ПРИЛОЖЕНИЕ А	65
ПРИЛОЖЕНИЕ Б.....	66

ПРИЛОЖЕНИЕ В	68
ПРИЛОЖЕНИЕ Г	71
ПРИЛОЖЕНИЕ Д	72

ВВЕДЕНИЕ

Развитие области информационных технологий и конкуренция между крупными предприятиями способствуют развитию программных решений для автоматизации управления бизнес-процессами. Одним из ключевых инструментов, широко используемых в современных реалиях, в этой сфере является 1С: “Центр Управления Предприятием”. Эта система предоставляет компаниям широкие возможности по организации управления бизнес-процессам, позволяет автоматизировать все подразделения и контуры учета производственного предприятия в соответствии российскими и международными стандартами.

В последние годы особое внимание уделяется развитию "умных" функций в рамках системы 1С: “Центр Управления Предприятием”. Такие инструменты позволяют компаниям автоматизировать процессы принятия решений, опираясь на анализ данных и использование методов машинного обучения. Результатом внедрения функциональности на основе технологий глубокого обучения является оптимизация времени на выполнение разнородных задач, повышение качества обслуживания клиентов. Транспортная компания “Деловые Линии”, являющаяся заказчиком программного продукта, описываемого в данной выпускной квалификационной работе, также считает это направление приоритетным для развития.

В стандартной конфигурации 1С: “Центр Управления Предприятием”, которая на данный момент используется в “Деловые Линии”, текстовый поиск обычно реализуется через использование фильтров и запросов, которые, в основном, основаны на наивном поиске по вхождению части строки в название искомого объекта. Пользователи могут вводить ключевые слова или фразы, и система ищет соответствия в текстовых полях записей базы данных. Такой метод имеет слабые стороны: он не учитывает синонимы или смысл поискового запроса, что часто приводит к неполному или

неудовлетворительному результату. Таким образом, этот подход часто не является наиболее эффективным и требует доработок для повышения точности и скорости поиска, а также для улучшения пользовательского опыта.

Одной из ключевых задач для компании “BIA Technologies” является создание интеллектуальной системы поиска с использованием методов машинного обучения, способной предложить пользователям – работникам компании – наиболее релевантные бизнес-процессы по текстовому запросу. Это позволяет сократить время на поиск корректного бизнес-процесса для составления заявки, снизить вероятность выбора ошибочного варианта клиентом.

Для создания интеллектуальной системы поиска в рамках 1С: “Центр Управления Предприятием” необходимо выполнить ряд этапов. В первую очередь, требуется получить исторических данных о поисковых запросах пользователей в системе, провести их обработку, очистку, исследование. Затем необходимо создать, обучить и протестировать модели машинного обучения, способные предсказывать соответствующие текстовому запросу бизнес-процессы. Наконец, необходимо создать сервис, к которому будет обращаться 1С: “Центр Управления Предприятием” для получения ответа от искусственного интеллекта.

Таким образом, целью данной выпускной работы является создание первой версии сервиса, который организующий поиск заявок с использованием искусственного интеллекта. Минимально жизнеспособный продукт имеет существенное ограничение по срокам реализации, поэтому в него не включена функциональность по автоматическому сбору данных в реальном времени и обучении модели на них.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Исследование исторических данных.

2. Создание набора релевантных данных для обучения и тестирования моделей машинного обучения.
3. Обработка данных.
4. Выбор метрик для оценки качества полученного решения.
5. Обучение и тестирование моделей машинного обучения.
6. Создание сервиса, использующего модели машинного обучения.

В данной выпускной работе будет следующий список пунктов:

1. Вступление.
2. Список используемых определений и терминов.
3. Первая глава – в ней освещена предметная область и постановка задачи.
4. Вторая глава – состоит из методики исследования данных и выводов по ним, техник для создания набора релевантных данных.
5. Третья глава – она содержит программную и систему архитектуру универсального модуля обработки данных, разработанного в ходе работы над проектом.
6. Четвёртая глава – в ней отображён процесс выбора метрик для оценки качества моделей, а также ход исследования, включающий эксперименты с различными архитектурами машинного обучения и выводы по ним.
7. Пятая глава – она включает в себя архитектуру разработанного программного комплекса для предсказания БП по текстовому запросу пользователя.
8. Заключение.
9. Список использованных источников

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ЦУП — 1С: “Центр Управления Предприятием”

БП — Бизнес-процесс, информационные и документационные потоки которого полностью или частично автоматизированы в 1С: ЦУП.

Умный поиск — сервис по поиску БП в 1С: ЦУП с использованием алгоритмов машинного обучения, разрабатываемый мной в ходе выпускной работы.

Заявка – документ в 1С: ЦУП, который создаётся в рамках БП. Является целевым объектом для клиента умного поиска.

Модель (Model) — это программная сущность, используемая для воссоздания поведения процесса при помощи определенного статистического метода, применяемого к историческим данным этого процесса.

Признак (Feature) — это числовая характеристика объекта или процесса.

Пайплайн (Pipeline) — это множество взаимосвязанных программных компонентов, используемых последовательно для обработки данных.

Предсказание (Prediction) — это ожидаемое поведение процесса, вычисленное моделью.

Метрика (Metric) — это функция, определяющая численное значение качества предсказаний модели на основе сравнения предсказанных и истинных данных.

Точность (Accuracy) — это доля правильно классифицированных образцов относительно общего числа образцов.

Средний обратный ранг (MRR) — это метрика, используемая в задачах ранжирования, которая оценивает качество модели по месту корректного ответа в выдаче.

Датасет (Dataset) – набор данных о поведении процесса.

Аугментация данных (Data Augmentation) — это увеличение датасета для обучения модели путем модификации существующих данных.

Целевая переменная (Target Variable) — признак объекта, в предсказании которого заключается задача модели.

Глубокое обучение (Deep Learning) — совокупность методов машинного обучения, основанных на обучении представлений, а не специализированных алгоритмах под конкретные задачи.

Лемматизация (Lemmatization) — процесс приведения словоформы к лемме, то есть её нормальной форме.

Токенизация (Tokenization) — это процесс разбиения фразы, предложения, абзаца или всего текстового документа на более мелкие единицы, например, отдельные слова или термины.

Стоп-слова (Stop Words) — это часто используемые слова, которые не вносят никакой дополнительной информации в текст.

Именованная сущность (NER) — это объект реального мира, такой как человек, местоположение, организация, продукт и другие, который можно обозначить именем собственным.

Трансформер (Transformer) — архитектура глубоких нейронных сетей, основанная на механизме внимания.

Гиперпараметры (Hyperparameters) — параметр модели, значение которого используется для управления процессом обучения.

Ансамбль моделей (Model Ensemble) — метод машинного обучения, при котором несколько индивидуальных моделей комбинируются для получения более точного и устойчивого прогноза. Каждая модель в ансамбле

может быть обучена на различных подмножествах данных или с использованием разных алгоритмов обучения.

Задача классификации с многими метками (Multilabel Classification) — тип задач машинного обучения, где каждый образец данных может быть помечен несколькими категориями или метками.

Бекэнд (Backend) — часть программного обеспечения, отвечающая за обработку и хранение данных, а также выполнение бизнес-логики приложения.

Суммаризация (Summarization) — получение резюме на основе текста.

Стемминг (Stemming) — это процесс нахождения основы слова для заданного исходного слова

MVP (Minimal Viable Product) — продукт, обладающий минимальными, но достаточными для удовлетворения первых потребителей функциями.

API (Application Programming Interface) - набор способов и правил, по которым различные программы общаются между собой и обмениваются данными.

ClearML — платформа для управления машинным обучением, которая автоматизирует и упрощает процессы подготовки данных, обучения моделей и отслеживания экспериментов для исследователей и инженеров. Также включает в себя холодное хранилище артефактов исследований с поддержкой версионирования.

ГЛАВА 1. ПРЕДМЕТНАЯ ОБЛАСТЬ

1.1 1С: “Центр Управления Предприятием”

1.1.1 История развития 1С

История развития приложений 1С в российском бизнесе началась в конце 1990-х годов. В то время основным средством автоматизации бухгалтерии и управления предприятием были локальные клиент-серверные решения. С развитием интернета и компьютерных технологий появилась потребность в удаленном доступе к данным и автоматизации бизнес-процессов. В ответ на это, 1С начала разрабатывать web-приложения для учета и управления, такие как 1С: Бухгалтерия онлайн, 1С: ERP в облаке и другие.

К 2005 году уже многие предприятия использовали web-приложения 1С для ведения бухгалтерии, управления складом и производством. С развитием мобильных технологий, стало возможным создание мобильных приложений для работы с системами 1С. Например, мобильное приложение "1С: Предприятие Mobile" позволяет руководителям и сотрудникам предприятий получать доступ к важной информации о компании и контролировать бизнес-процессы прямо с мобильного устройства.

Сегодня приложения 1С предоставляют широкий спектр услуг для бизнеса: от учета и анализа финансовой деятельности до автоматизации управления производственными процессами. С их помощью пользователи могут вести учет товаров и услуг, контролировать финансовое состояние предприятия, управлять персоналом и многое другое. Это значительно упрощает работу предприятия и повышает его эффективность в условиях современного делового мира.

1.1.2 1С: “Центр Управления Предприятием” в Деловые Линии

Центр Управления Предприятием в рамках компании Деловые Линии представляет собой комплексное программное решение для автоматизации управленческих процессов и ведения бизнеса. Развитие этой платформы началось в начале 2000-х годов, когда компания столкнулась с необходимостью улучшения операционной эффективности и контроля над бизнес-процессами.

На данный момент, по официальным данным [1], компания насчитывает порядка 31000 сотрудников, и практически каждый из них использует 1С: “Центр Управления Предприятием” для взаимодействия с клиентами и поставщиками, составления заявок и отчётов.

1.1.3 Интерфейс 1С: “Центр Управления Предприятием”

Интерфейс 1С: “Центр Управления Предприятием” для рядового пользователя может показаться очень комплексным и запутанным. В рамках данной выпускной работы мы будем рассматривать только ту его часть, которая непосредственно относится к внедряемому сервису – окно поиска бизнес-процессов на рабочем столе. На рисунке оно выделено красным прямоугольником.

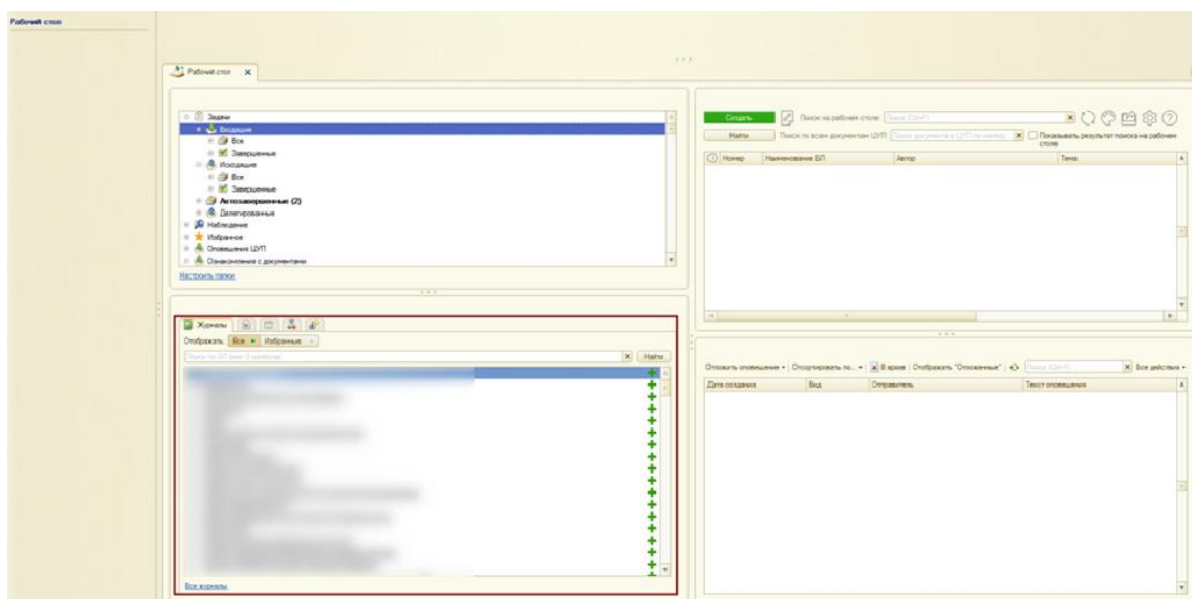


Рисунок 1 – Рабочий стол 1С: “Центр Управления Предприятием”

В этом списке пользователь может выбрать интересующий его бизнес-процесс и составить заявку.

1.1.4 Поиск БП в 1С: “Центр Управления Предприятием”

Однако, поиск необходимой заявки в списке из более чем 100 вариантов требует значительное количество времени. Именно для этого в данное поле был встроен инструмент “Поиск”. На рисунке 1 можно пронаблюдать текстовое поле и кнопку “Найти”, которые отвечают за его реализацию на клиентской части приложения. Логика поиска в решении 1С: “Центр Управления Предприятием” “из коробки” является простой — это текстовый поиск, который выдаёт результаты, в названиях которых есть указанный запрос.

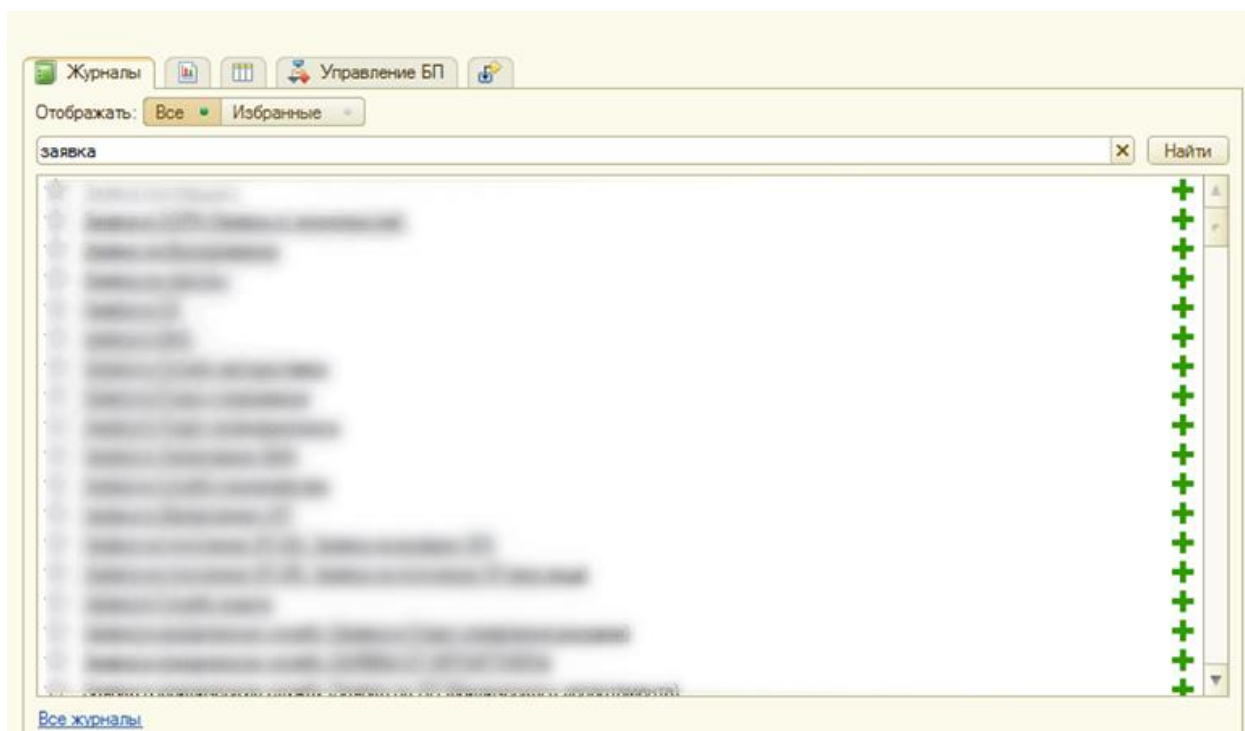


Рисунок 2 – Пример поискового запроса “заявка” в оригинальной реализации поиска

Текущая организация поиска не удовлетворила заказчика по следующим критериям:

- 1) Сотрудники затрачивают значительное количество времени на поиск необходимых бизнес-процессов.
- 2) Сотрудники совершают ошибки при составлении заявок, так как не могут корректно определить релевантный бизнес-процесс.
- 3) Для поиска корректного бизнес-процесса сотруднику необходимо знать точное название или его часть, что вызывает сложности у нового персонала.

Таким образом, он обратился к компании “BIA Technologies” с запросом на системное решение.

1.2 Постановка задачи

1.2.1 Описание задачи от заказчика

Пользователь системы ЦУП желает, чтобы при вводе в строку поиска БП ЦУП текста или части текста, система подбирала подходящий по названию или по смыслу БП ЦУП.

1.2.2 Тема исследования

Разработка модели машинного обучения для автоматического определения бизнес-процесса, который наиболее подходит под запрос пользователя.

1.2.3 Цели исследования

Снижение затрачиваемого клиентом времени на оформление заявки.

Снижение количества ошибочно оформленных заявок по причине неверного выбора.

1.2.4 Постановка задачи

Постановка задачи машинного обучения: разработка модели или ансамбля моделей для решения задачи классификации с многими метками на основе языковых данных.

Постановка задачи бекэнд: разработка сервиса с REST API для организации доступа к модели.

ГЛАВА 2. ИССЛЕДОВАНИЕ ДАННЫХ

2.1 Постановка задачи на этап

2.1.1 Цель исследования

Сформировать набор данных, пригодный для использования в модели машинного обучения для решения задачи.

2.2 Исследование исторических данных

2.2.1 Описание данных

От заказчика были получены следующие файлы - выгрузка исторических данных. Период выгрузки 26.01.2023 – 14.01.2024

service_requests_documents.csv — файл с произвольно составленными документами из базы "Заявка в службу (общая)"

individual_documents.csv — файл с произвольно составленными документами из базы "База бизнес-процессов"

service_requests_documents_templates.csv — файл с документами, сформированными на основе шаблонов, из базы "Заявка в службу (общая)"

individual_documents_templates.csv — файл с документами, сформированными на основе шаблонов, из базы "База бизнес-процессов"

Исследуемые документы имеют 2 источника: база "Заявка в службу (общая)" и база "База бизнес-процессов". Также документ может быть либо составлен человеком вручную, либо с использованием шаблона.

Каждый из документов имеет следующую структуру:

date — дата создания документа

number — регистрационный номер документа

topic — заголовок документа

text — текст документа

bp — наименование бизнес-процесса

bp_id — идентификационный номер бизнес-процесса

date	number	topic	text	bp	bp_id
2023-01-26 10:39:23	030044586	01. Предоставление оборудования/01.2. Отсутствие в горячем резерве	... Проблемы с: Принтер HP LaserJet M110we \r\n Укажите вид и модель устройства\r\nПодключенный: \r\nUSB кабелем, к этому же компьютеру, По сети к другому компьютеру	Заявка на компьютерный сервис	ea77c126-24b3-11e2-92ee-e41f13b975c8
2023-06-22 15:54:45	064152476	Ежегодный		Заявка на согласование отпуска	4725237-05d9-11e1-847d-001a643625fe

Таблица 1 – Примеры исследуемых документов. Первый составлен по вручную, второй – по шаблону.

2.2.2 План исследования

1. Объединить исходные данные.
2. Провести исследование стандартных характеристик каждого признака.

3. Отфильтровать неудовлетворительные образцы.
4. Сделать выводы, сформировать вопросы к заказчику.

2.2.3 Ход работы

Количество образцов в объединённой выборке: 1.216.984

Даты выгрузки: 26.01.2023 – 14.01.2024

Количество уникальных номеров: 1.021.754 (дубликаты удаляем)

Объединим topic и text в колонку full_text для удобства.

Имеем 0.5% пустых значений full_text в выборке, можно пренебречь.

Процент пустых значений в данных

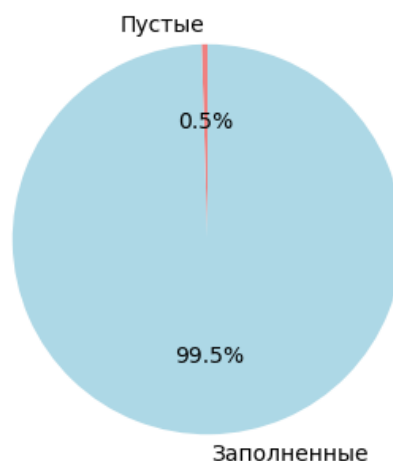


Рисунок 3 – Круговая диаграмма значений full_text

Рассмотрим график “Ящик с усами” для нахождения аномальных значений длин текстов заявок. В разрезе данной визуализации можно обнаружить тексты длиной в несколько тысяч и десятков тысяч, что аномально много для данной предметной области.

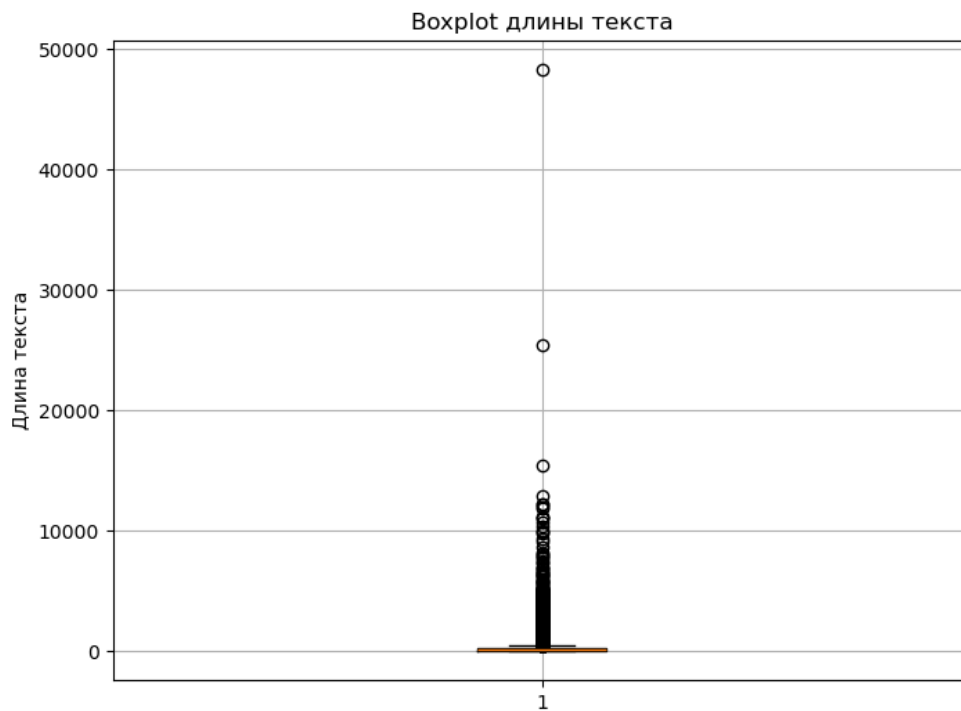


Рисунок 4 – График “Ящик с усами” для оценки распределения длин сообщений.

Устраним аномальные данные, установив нижнюю и верхнюю границы по формулам.

$$IQR = Q_3 - Q_1$$

$$X_1 = Q_1 - 1.5 * IQR$$

$$X_2 = Q_3 + 1.5 * IQR$$

Где Q_3 – значение 75-ого квантиля исходных данных, Q_1 – значение 25-ого квантиля исходных данных, IQR – межквантильное расстояние, X_1 – нижняя граница уса, X_2 – верхняя граница уса.

Полученные значения границ: $X_1 = -178$, $X_2 = 462$.

Количество удовлетворяющих этим критериям значений: 968.589

Имеем 6.5% пустых значений br , можно временно пренебречь.

Процент пустых значений в данных

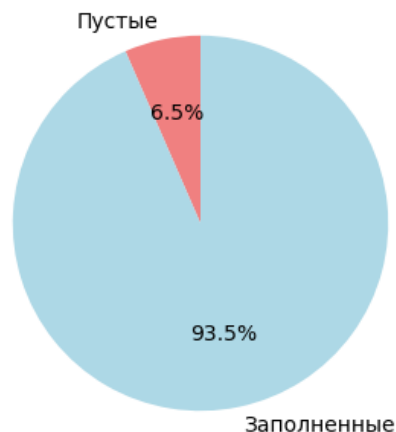


Рисунок 5 – Круговая диаграмма значений bp

Количество уникальных БП в данных: 45

Распределение данных по классам по классам БП – Приложение А, рисунок 1

Количество текстов, составленных по шаблону: 191.195

2.2.4 Выводы и предложения

В выборке присутствует всего 45 БП из 126 целевых для заказчика. Если поиск будет основан на 1 алгоритме, обученном на данной выборке, будет невозможно найти 64.3% БП, что негативно скажется на качестве пользовательского опыта. Нужно использовать ансамбль с алгоритмами, работающими по другому принципу: текстовый поиск, сематический поиск, основанные на названиях БП, а не на текстах заявок.

Тексты в выборке, являющиеся текстами составленных заявок, не соответствуют целевым для поиска БП. Например, заявке в Административно-Хозяйственный Отдел с текстом “Добрый день! Меня зовут Иванов Иван Иванович, я работник склада С. города Н. На рабочем месте №Х перегорела лампочка. Прошу её заменить!” будет соответствовать один из поисковых запросов: “замена лампочки”, “заявка на замену лампочки” и т.п. Таким образом, приходим к выводу, что необходима аугментация данных для

моделирования поведения пользователя и, следовательно, составления наиболее релевантного набора данных для обучения.

19.7% заявок в выборке составлены по шаблонам 1С: ЦУП, что предполагает наличие в них общих фраз и нетекстовых символов, из-за чего аугментация этих данных, а также обучение на них будет затруднено. Необходимо получить у заказчика список шаблонов и выработать решение по извлечению пользовательского ввода из них.

6.5% записей имеют пустое поле БП. Возможно, необходимо запросить разметку образцов у заказчика или решить эту задачу методами кластеризации. Задача кластеризации была перенесена в этап боевого решения, она не будет рассматриваться в данной работе.

2.3 Извлечение пользовательского ввода из шаблонов 1С:

ЦУП

2.3.1 Описание данных

От заказчика был получен файл с используемыми шаблонами:

service_requests_templates.csv — файл с шаблонами заявок как базы “Заявка в службу (общая)”, так и базы “База бизнес-процессов”.

Каждый из документов имеет следующую структуру:

template_code — уникальный код шаблона.

name — название шаблона.

template_text — текст документа, написанного на основании шаблона.

template_code	name	template_text
000001349	Шаблон: Проблемы с устройством ввода-вывода (Принтер)	... Проблемы с: Принтер HP LaserJet M110we \r\n Укажите вид и модель устройства\r\nПодключенный: \r\nUSB кабелем, к этому же компьютеру, По сети к другому компьютеру

Таблица 2 – Пример данных о шаблоне.

2.3.2 План исследования

1. Рассмотреть данные из файла с шаблонами.
2. Составить класс для описания каждого шаблона, его ключевых полей.
3. Составить класс для извлечения пользовательского ввода из текстов с помощью описания шаблонов.

4. Составить статистику по успешности извлечения данных.

2.3.3 Ход работы

Количество данных о шаблонах: 391

Разработан класс `Template` для описания шаблона, включающий в себя ключевые поля шаблона, их длины и идентификаторы. Он принимает такие входные параметры, как код, имя, текст, разделитель и пользовательский ввод. Он обрабатывает входной текст для определения тегов, индексов тегов и ключей. Теги идентифицируются с помощью регулярных выражений, а ключи извлекаются путем разбиения текста на основе идентифицированных тегов. Класс предоставляет методы для получения ключей, длины ключей и индексов тегов. Код класса – приложение Б, рисунок 1

Класс `Extractor` для извлечения из текста, составленного на основе шаблона, пользовательского ввода. Он инициализируется с данными об имеющихся шаблонах, затем извлекает шаблоны и сопоставляет ключи в документах. Он определяет индекс шаблона, соответствующего документу, извлекает значения, основанные на ключах, и возвращает их в виде строки. Класс предоставляет методы для сопоставления элементов в тексте, поиска индексов шаблонов, получения начальных ключей и извлечения значений из документов на основе шаблонов. Код класса – приложение Б, рисунок 2

Имеем 191.195 образцов, составленных на основе шаблонов. Применим созданные классы на данных и рассмотрим результаты.

	Всего	Извлечено	Не извлечено	Доля извлеченных
Заявка в службу (общая)	58906	45478	13428	0.77
База бизнес-процессов	132289	80918	51371	0.61
Всего	191195	126396	64799	0.66

Таблица 3 – Сводная таблица результатов извлечения текстов пользователей из шаблонов

2.3.4 Выводы

Удалось извлечь пользовательский ввод из 66% запросов, составленных по шаблонам 1С: ЦУП. 126.396 образцов дополняют обучающую выборку, что увеличит её объём на 16.3%, что может значительно улучшить обобщающую способность модели, а это скажется на качестве генерируемых ответов.

В остальных 34% случаев возникла ошибка при извлечении. Одна из основных причин ошибки – заказчик выгрузил нам актуальные версии шаблонов, которые нерелевантны для части исторических данных.

На этапе MVP продолжать исследование нецелесообразно, так как на этапе постановки были зафиксированы жёсткие сроки разработки. Получение исторических версий шаблонов или формирование их вручную увеличит трудозатраты, запланированные на данный этап, в 1.5 – 2 раза по оценке аналитика, что не удовлетворяет установленным срокам.

2.4 Аугментация данных

2.4.1 Описание данных

На данном этапе будем использовать данные из шагов 2.2 – образцы без шаблонов и 2.3 – заявки, пользовательский ввод которых удалось извлечь. Очистим полученные данные от лишних колонок.

В итоге, каждый из документов имеет следующую структуру:

bp_id — идентификатор БП.

bp — название БП.

text — тема и текст документа.

bp_id	bp	text
e31effd2-23fc-11e3-8194-6eae8b603f99	Заявка в Департамент безопасности ЛК (БП Заявка в службу контроля и режима)	Фото движения груза.\nДобрый день.\nПросьба предоставить фото движения груза №Х.

2.4.2 План исследования

1. Провести анализ доступного инструментария для получения поискового запроса по полному тексту образца.
2. Провести тестирование инструментария, доступного для локального развёртывания в корпоративном контуре.
3. Создать скрипт для автоматического создания поискового запроса по текстам выборки.

2.4.3 Ход работы

Исследование было сфокусировано на инструментах, позволяющих сократить, суммаризовать исходный текст. В основном, подобранные подходы

основываются на подходах глубокого обучения, в частности моделях архитектуры трансформер и больших языковых моделях.

Сбер Суммаризатор: использует технологии машинного обучения для автоматического краткого извлечения ключевой информации из текста, помогая сократить объем и улучшить понимание текстовых данных.

bert-extractive-summarizer: реализация суммаризатора на основе BERT, который извлекает наиболее информативные предложения из текста для создания краткого обзора или сводки.

IlyaGusev/rugpt3medium_sum_gazeta: модель, основанная на RuGPT-3.0, предназначенная для автоматического создания кратких сводок или резюме новостных статей из российских газет.

UrukHan/t5-russian-summarization: модель, основанная на T5, которая применяется для русскоязычной суммаризации текста, выделяя ключевые аспекты и сведения для краткого обзора.

UrukHan/t5-russian-summarization: аналогично предыдущей модели, она также использует архитектуру T5 для суммаризации текста на русском языке, помогая создавать сжатые обзоры информации.

Llama-2-70B-chat: большая языковая модели с открытым исходным кодом, предназначенная для чат-ботов, использует 70 миллиардов параметров для обучения и генерации текста.

Llama-2-13B-chat: большая языковая модели с открытым исходным кодом, предназначенная для чат-ботов, уменьшенная относительно версии на 70 миллиардов параметров до 13.

ai_forever/ruGPT-3.5: модель, основанная на RuGPT-3.5, обученная на русскоязычных данных для генерации текста и решения различных задач NLP.

ai_forever/FRED-T5-1.7B: использует архитектуру T5 и 1.7 миллиарда параметров для обработки текста, создания суммаризаций, ответов на вопросы и других задач.

h2oai/h2ogpt-gm-oasst1-en-2048-falcon-40b-v2: Мощная модель, использующая архитектуру GPT, обученная на 40 миллиардах параметров для генерации текста на английском языке.

Saiga2-Lora: адаптер для русского языка для моделей архитектуры Llama2, созданный российским разработчиком PyaGusev.

DeepL: сервис машинного перевода, обладающий высокой точностью и натуральностью перевода на различные языки. В его основе лежат модели глубокого обучения.

Yandex Translate: сервис перевода от Яндекса с широким набором поддерживаемых языков и возможностей интеграции.

Google Translate: Известный сервис машинного перевода от Google с широким спектром поддерживаемых языков и возможностями синтеза речи.

Argos: сервис машинного перевода, написанный на python, с открытым исходным кодом.

Проведём тестирование. Для него было подобрано несколько разноплановых текстов заявок, на которых мы будем проверять качество извлечения сути. Объективной метрики, определяющей сохранность точного смысла предложения обнаружено не было, так что оценка будет проводиться умозрительно. Дальнейшие результаты будут представлены в виде таблицы с легендой для удобства восприятия.

Значение	Цвет
Неудовлетворительный результат, не стоит использовать.	
Неудовлетворительный результат, но есть важные детали, не стоит использовать.	
Удовлетворительный результат, использование осуществимо, но нежелательно.	
Хороший результат, есть замечания, использование возможно.	
Отличный результат, рекомендовано к использованию.	
Эталонный результат.	

Таблица 4 – Обозначения для цветов в таблице с результатами тестирования.

Инструмент	Вариант использования	Резюме
Llama-2-70B-chat	Запрос на русском языке	Эталонное качество с учётом невозможности регулировать параметры модели. Есть возможность подобрать более релевантный запрос для улучшения качества. Используется сторонний сервис, локальное развёртывание невозможно.
Llama-2-13B-chat	Запрос на русском языке	Работает в режиме чат бота, не следует запросу. Неудовлетворительно.
Llama-2-13B-chat	Запрос на русском языке с внутренним переводом	Неплохо справляется с простыми запросами, на сложных полностью теряется смысл, добавляются новые

		детали. Необходимо 3 запроса для обработки 1 текста.
Llama-2-13B-chat с Saiga2-Lora	Запрос на русском языке	На простых примерах качество неплохое, на сложных не очень удовлетворительно. Можно отредактировать запрос, чтобы при невозможности краткого ответа выдавалось: "Не знаю."
Llama-2-13B-chat с Saiga2-Lora	Запрос на русском языке с внутренним переводом	Качество перевода полного предложения с английского на русский неплохое, суммаризация качественная. Перевод сущности сообщения с английского на русский среднего качества. Вероятно, с помощью добавления контекста и подбора запроса это можно улучшить. Необходимо 3 запроса для обработки 1 текста.
Llama-2-7B-chat с Saiga2-Lora	Запрос на русском языке	На части запросов результаты такие же или несколько хуже, чем 13B, с другими не справляется в целом. Также нестабильные результаты даже для одного списка параметров. Скорость в 2.5 раза выше, чем у 13B.
Llama-2-7B-chat с Saiga2-Lora	Запрос на русском языке с внутренним переводом	На простых запросах показывает хорошее качество и производительность, но на сложных проявляется ошибка - начинает отвечать на сообщение вместо

		перевода, запрос и настройки не помогают исправить ситуацию. Необходимо 3 запроса для обработки 1 текста.
ai_forever/ruGPT-3.5	Запрос на русском языке	Неудовлетворительно, т.к. ведут себя как чат-помощники, продолжают указанный текст вместо решения задачи. Подобрать запрос и настройки для решения поставленной задачи не удалось.
ai_forever/FRED-T5-1.7B	Запрос на русском языке	Аналогично предыдущему.
h2oai/h2ogpt-gm-oasst1-en-2048-falcon-40b-v2	Запрос на русском языке	Аналогично предыдущему.
Сбер Суммаризатор	Запрос на русском языке	Неудовлетворительно, т.к. добавляет новые детали, неправильно расставляет акценты, теряя смысл.
bert-extractive-summarizer	Запрос на русском языке	Аналогично предыдущему.
bert-extractive-summarizer	Запрос на русском языке	Аналогично предыдущему.
UrukHan/t5-russian-summarization	Запрос на русском языке	Аналогично предыдущему.
Llama-2-13B-chat с DeepL	Запрос на русском языке с переводом с помощью утилиты	Эталонное качество сокращения и перевода. Результат похож на запрос пользователя к поисковой системе. Нет

		возможности использовать DeepL локально, невозможно использовать.
Llama-2-13B-chat с Argos	Запрос на русском языке с переводом с помощью утилиты	Качество суммаризации хорошее. Улавливает больше деталей из сообщения, чем эталон, но длина запроса становится больше. Менее схожа с ожидаемым вводом пользователя. Качество обратного перевода удовлетворительное, теряет детали и контекст слова.
Llama-2-13B-chat с Yandex Translate Offline	Запрос на русском языке с переводом с помощью утилиты	Качество суммаризации хорошее. Улавливает больше деталей из сообщения, чем эталон, но длина запроса становится больше. Менее схожа с ожидаемым вводом пользователя. Качество перевода хорошее, но присутствуют ошибки.
Llama-2-13B-chat с Google Translate Offline	Запрос на русском языке с переводом с помощью утилиты	Эталонное качество перевода с сохранением контекста, но необходимо использовать эмулятор android. Можно использовать без интернета. Недостатки: низкая производительность. Качество суммаризации сравнимо с эталоном.

Таблица 5 – Результаты тестирования инструментов для суммаризации.

Для реализации был выбран подход Llama-2-13B-chat с Google Translate Offline.

Взаимодействие с большой языковой моделью осуществлялось при помощи готового приложения [2]

2.4.4 Выводы

Выбранный подход позволил суммаризовать 455.912 образцов с хорошим качеством за время разработки. Пример: полный текст “Прошу принять документы по накладной от компании Х. Номер счета фактуры исправлен. Адрес не является существенным недочетом”, суть “Принять документы по накладной.”. Полученные тексты умозрительно напоминают предполагаемый шаблон действия пользователя в рамках системы поиска.

Таким образом, получен набор релевантных данных для обучения и тестирования моделей машинного обучения. Для проведения исследования моделей машинного обучения необходимо обработать полученные тексты для улучшения показателей обобщающей способности моделей.

Далее рассмотрим процесс разработки универсального модуля для обработки текстов, описанный в Главе 3.

ГЛАВА 3. УНИВЕРСАЛЬНЫЙ МОДУЛЬ ОБРАБОТКИ ТЕКСТОВЫХ ДАННЫХ

3.1 Постановка задачи на этап

3.1.1 Цель этапа

Необходимо реализовать универсальный модуль обработки текста. На вход модуля подается датасет с указанием полей, которые нужно обработать. В конфигурации модуля выполняется настройка алгоритма выполнения предобработки. В результате обработки возвращается датасет с новым атрибутом - обработанный текст. Модуль должен быть реализован как для использования в ходе исследования, так и для подключения в сервис.

3.1.2 Функциональные требования

Аналитиком были выдвинуты следующие функциональные требования:

- 1) Основной модуль необходимо реализовать в виде универсального интерфейса с возможностью гибкого подключения различных реализаций этапов предобработки, расширения функциональности.
- 2) Настройки для модуля хранятся в файле конфигурации, необходимо реализовать возможность использования файлов разных расширений (json, yaml).
- 3) Конкретный алгоритм отдельной функции подготовки определяется файлом настроек.
- 4) Состав и порядок выполнения этапов обработки текстов определяется файлом настроек. Порядок выполнения должен быть проверен на корректность при чтении файла настроек.
- 5) Справочники (такие как словарь расшифровок, регулярные выражения для замены определенных паттернов) должны храниться вне кода реализации модуля для удобного дополнения.

- 6) Класс должен предусматривать возможность обработки как локальных файлов, так и загрузку из ClearML.

3.1.3 Требования к шагам обработки текста

Необходимо реализовать основные шаги обработки текста, которые используется в каждом проекте команды. Их список:

- 1) Замена аббревиатур (многозначные аббревиатуры оставляем в исходном виде)
- 2) Лемматизация
- 3) Стемминг
- 4) Удаление цифр
- 5) Нижний регистр
- 6) Замена всех вариационных пробельных символов на одинарный пробел
- 7) Замена именованных сущностей на соответствующие теги
- 8) Замена шаблонных сущностей на соответствующие теги (номер накладной, номер паспорта, номер машины, номер эл. почты и т.п.), либо удаление этих последовательностей из текста

3.2 Проектирование

3.2.1 Стек технологий

В нашей компании используются следующие средства для разработки программных решений.

Среда разработки: PyCharm

Язык программирования: Python 3.10

Для реализации данного модуля были выбраны следующие технологии. Это наиболее популярные и эффективные решения для работы с русскоязычными данными на языке Python.

Библиотеки и фреймворки: pandas, clearml, datasets, natasha, nltk, pyyaml, spacy, re

3.2.2 Программная архитектура

На рисунке показано отношение компонентов для универсального модуля обработки текстов. Он состоит из 5 частей:

- 1) Configuration – модуль, отвечающий за чтение файлов конфигурации, составление пайплайна загрузки и обработки текстовых данных.
- 2) Data Collection – модуль, реализующий функциональность сбора данных как из локальных, так и из облачных ресурсов.
- 3) Data Preprocessing – модуль, включающий инструменты для обработки полного набора данных, например, очистки от пропущенных значений.
- 4) Text Processing – модуль, содержащий набор классов для трансформации текстовых данных.
- 5) Data Processing – основной компонент универсального модуля обработки текстов, в который подключаются вышеописанные модули.

Рабочее название модуля – ML Data Engine

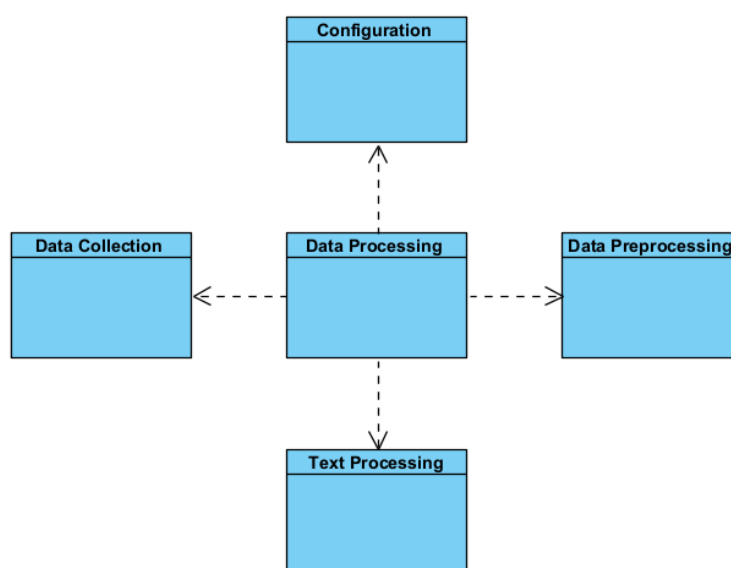


Рисунок 6 – Диаграмма модулей ML Data Engine

3.3 Разработка и тестирование

3.3.1 Реализация

Для соответствия функциональным требованиям был разработан набор классов, который можно рассмотреть на: диаграмме классов универсального модуля обработки текста – Приложение В, рисунок 1

Была создана гибкая архитектура с возможностью расширения любого из компонентов модуля при помощи парадигмы объектно-ориентированного программирования и принципов SOLID.

Для демонстрации реализации компонентов были выбраны классы из каждого компонента модуля:

Реализация класса для сбора данных формата csv из ClearML - Приложение В, рисунок 2

Реализация абстрактного класса для работы с неопределёнными значениями - Приложение В, рисунок 3

Реализация класса для лемматизации текстовых данных - Приложение В, рисунок 4

3.3.2 Тестирование

Мной был разработан ряд модульных тестов с использованием техник “разбиение на классы эквивалентности” и “анализ граничных значений” по методике “белый ящик”. Общее покрытие строк кода тестами составило 94%, отчёт о покрытии – Приложение В, рисунок 2

3.3.3 Выводы

В ходе данного этапа был разработан универсальный модуль для обработки текстовых данных, который соответствует всем заявленным функциональным требованиям и требованиям к реализации. Он будет

использован как в ходе исследований, так и в составе боевого решения в качестве подключаемой библиотеки. С его помощью данные, полученные в Главе 2, были обработаны. Таким образом, получили релевантный набор данных для проведения экспериментов. Перейдём к этапу исследованию моделей машинного обучения, описанному в Главе 4.

ГЛАВА 4. ИССЛЕДОВАНИЕ МОДЕЛЕЙ МАШИННОГО ОБУЧЕНИЯ

4.1 Постановка задачи на этап

4.1.1 Цель исследования

Создание ансамбля моделей машинного обучения для определения БП по поисковому запросу.

4.2 Проведение экспериментов с моделями машинного обучения

4.2.1 Описание задачи

На этапе исследования данных мы столкнулись с проблемой, которая заключается в том, что в имеющихся данных отсутствуют образцы, принадлежащие к существенной доле актуальных бизнес-процессов. Таким образом, алгоритм не сможет предсказывать эти классы, что негативно скажется на пользовательском опыте. В связи с этим было принято решение о создании ансамбля алгоритмов для решения данной задачи.

Задача была декомпозирована на 3 аспекта:

1. Умный поиск должен учитывать текстовое сходство запроса и поисковых категорий. С учётом встроенной в 1С: ЦУП системы, на данный момент это основной пользовательский шаблон действий, поэтому он обязан присутствовать в ансамбле с самым высоким приоритетом. Пример: на запрос “отпуск заявка” выдаётся “Заявка на отпуск”.

2. Сервис не может упускать из вида семантическое сходство поискового запроса и названий бизнес-процессов. Алгоритм или модель такого формата должна иметь 2 приоритет в ансамбле. Пример: на запрос “кадровые операции” выдаётся “Кадровая операция”.

3. Последним запросом заказчика был учёт логической связи введённого текста и искомого БП. Этот аспект будем решать с помощью модели машинного обучения на основании, полученного на прошлых этапах датасета. Пример: на запрос “заменить лампочку” выдаётся “Заявка в Административно-хозяйственный отдел”.

4.2.2 План исследования

1. Выбрать ряд метрик для оценки эффективности алгоритмов.
2. Провести исследования алгоритмов текстового поиска, составить первый слой ансамбля.
3. Провести исследования алгоритмов семантического поиска, составить второй слой ансамбля.
4. Провести исследования алгоритмов моделей, составить третий слой ансамбля.
5. Провести тестирование ансамбля, сделать выводы.

4.2.3 Ход работы

Для корректной оценки работы как алгоритмов, так и ансамбля, необходимо учитывать 2 аспекта: корректность выдачи – нахождение правильного ответа в результатах – и ранжирование – насколько близко к первым позициям находится верный БП.

Таким образом, минимальный набор метрик включает:

TopN Accuracy:

$$TopN Accuracy = \frac{1}{|Q|} * \sum_{i=1}^{|Q|} K_i$$

Где Q – общее количество выдач алгоритма,

$$K_i = \begin{cases} 1, & \text{если корректный ответ находится в первых } N \text{ результатах} \\ 0, & \text{в противном случае} \end{cases}$$

Accuracy:

$$Accuracy = \frac{1}{|Q|} * \sum_{i=1}^{|Q|} K_i$$

Где Q – общее количество выдач алгоритма,

$$K_i = \begin{cases} 1, & \text{если корректный ответ первый в результатах} \\ 0, & \text{в противном случае} \end{cases}$$

Mean Reciprocal Rank:

$$MRR = \frac{1}{|Q|} * \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

Где Q – общее количество выдач алгоритма, rank_i – позиция первого корректного ответа в выдаче.

Для всех перечисленных метрик наилучшим значением является 1, наихудшим - 0.

Для удовлетворения критерия текстового поиска была выбрана библиотека Whoosh [3]. Это быстрая, многофункциональная библиотека поисковой системы на языке Python. Она предоставляет набор инструментов для индексирования текста и последующего поиска в этом индексе, подобно тому, как работают поисковые системы вроде Google в меньшем масштабе. Whoosh предназначен для использования в приложениях, которым нужно искать в больших объемах текста и извлекать документы, соответствующие запросу, а также выполнять их ранжирование.

Поиск в Whoosh использует следующие шаги при работе:


Токенизация: текст документов разбивается на слова или лексемы.

Индексирование: библиотека создает из лексем инвертированный индекс - структуру данных, которая позволяет быстро находить документы, содержащие те или иные лексемы. Этот индекс хранится на диске, и по нему можно осуществлять эффективный поиск.

Обработка запросов: когда поступает запрос, Whoosh разбирает его на составляющие лексемы, аналогично тому, как обрабатываются документы. Затем он ищет эти лексемы в инвертированном индексе, чтобы найти подходящие документы.

Оценка и ранжирование: Соответствующие документы оцениваются в зависимости от того, насколько они соответствуют критериям запроса. Стандартный алгоритм – BM25 [4].

Создадим поисковый индекс на основании полученных от заказчика данных о релевантных названиях БП, предварительно обработав их при помощи универсального модуля, разработанного в предыдущих этапах.



```
1 def create_search_index(df: pd.DataFrame, indexname: str = "bp_search") -> None:
2     storage = FileStorage(SEARCH_INDEX_DIR)
3
4     ix = storage.create_index(BpSchema, indexname=indexname)
5     writer = ix.writer()
6
7     for _, row in df.iterrows():
8         writer.add_document(bp_id=row['bp_id'], name=row['name'], content=row['processed_text'])
9
10    writer.commit()
```

Рисунок 7 – Метод создания индекса для Whoosh


```

1 def search_in_index(text: str, indexname: str="bp_search") -> Dict[str, Dict[str, float]]:
2     storage = FileStorage(SEARCH_INDEX_DIR)
3     ix = storage.open_index(indexname=indexname)
4
5     text = DataProcessor.process_single_text(text)
6     with ix.searcher() as searcher:
7         query = QueryParser("content", ix.schema).parse(text)
8
9         results = searcher.search(query, terms=True, limit=6)
10        pred = {}
11        for r in results:
12            pred.setdefault(r['bp_id'], {'name': r['name'], 'score': r.score})
13
14    return pred

```

Рисунок 8 – Метод для использования текстового поиска

Тестирование проводилось на том же наборе данных, на котором был построен индекс. В его ходе были проведена аугментация – исключение случайных слов, смена их порядка.

Результаты:

Метрика	Значение
TopN Accuracy	0.997
Accuracy	0.966
MRR	0.984

Таблица 6 – Результаты тестирования текстового поиска

В итоге, получили качественный алгоритм текстового поиска, который станет первым слоем умного поиска и будет обслуживать наиболее популярные запросы.

Для удовлетворения критерия текстового поиска была выбрана библиотека Faiss [5]. Это библиотека с открытым исходным кодом, разработанная командой Facebook AI Research. Она предназначена для эффективного поиска сходства и кластеризации плотных векторов. Faiss особенно полезна в ситуациях, когда имеются большие наборы векторов, и


задача состоит в том, чтобы быстро найти векторы, которые наиболее похожи на заданный вектор запроса.

Faiss использует алгоритмы, оптимизированные как для CPU, так и для GPU, для вычисления сходства между векторами, что делает его чрезвычайно быстрым при поиске в больших массивах данных. Он поддерживает различные типы метрик для вычисления сходства векторов.

В рамках этапа будем использовать Faiss как индекс для поиска, вектора, наиболее схожего со сформированным моделью `distiluse-base-multilingual-cased-v1`. Это расширенная версия универсального кодировщика предложений, оптимизированная для нескольких языков и обеспечивающая эффективную генерацию вкраплений для многоязычных задач понимания и уподобления текстов.

Для предобработки текстов используем следующую последовательность шагов в универсальном модуле. Данный ряд преобразований был выработан при совместном обсуждении с тимлидом. Эти шаги позволят добиться наибольшей универсальности текстов, но, при этом, не повредят их смысл.

- 1) Замена именованных сущностей на токены
- 2) Нижний регистр
- 3) Удаление цифр
- 4) Лемматизация
- 5) Удаление стоп-слов на русском языке
- 6) Замена аббревиатур на развёрнутые версии
- 7) Удаление несловесных символов
- 8) Удаление пробелов



```

1  def _predict(self, text: str) -> List[ModelPrediction]:
2      predictions: List[ModelPrediction] = []
3      vector = np.array([self._model.encode(text)]) # type: ignore
4      distances, neighbours = self._index.search( # type: ignore
5          vector, self._config.common_properties.output_count
6      )
7
8      for index, distance in zip(neighbours[0], distances[0]):
9          index_row = self._index_controller.get_index_row(index)
10         predictions.append(
11             ModelPrediction(
12                 bp_code=index_row[0],
13                 request_type_id=index_row[1],
14                 confidence=1 - distance,
15                 name=self._name,
16                 version=self._version,
17             )
18         )
19
20     return predictions

```

Рисунок 9 – Метод предсказания у семантического поиска

Полномасштабное тестирование эффективности данного слоя на этапе MVP не проводилось за неимением релевантных данных. Ручное тестирование продемонстрировало достойную производительность.

В итоге, получили качественный и производительный алгоритм семантического поиска, который учтёт нечёткий ввод пользователя. Он займёт место второго слоя умного поиска.

В рамках исследования последнего этапа исследования будут проведены эксперименты над различными архитектурами моделей машинного обучения, после чего будет выбрана лучшая по совокупности метрик.

Необходимо помнить, что мы решаем задачу классификации текстовых данных произвольной длины, поэтому нужно учитывать необходимость векторизации данных для приведения их к формату численных векторов

равной длины. Альтернативным вариантом решения данной проблемы является использование архитектур нейронных сетей, поддерживающих ввод произвольной размерности в одной из проекций.

Данные были разделены в пропорции 80:20 на обучающий датасет и тестовый для базовых алгоритмов, и 75:5:20 на обучающий, валидационный и тестовый для алгоритмов глубокого обучения. Это стандартные пропорции, используемые для исследований.

Подбор релевантных гиперпараметров для базовых алгоритмов проводился с помощью кросс-валидации, для моделей машинного обучения вручную.

Для предобработки текстов используем аналогичную шагу сематического поиска последовательность шагов в универсальном модуле.

Использованные векторизаторы:

Векторизатор TFIDF — преобразует текст во взвешенный векторный формат на основе частоты терминов и обратной частоты документов, выделяя важные слова, уникальные для документов.

Векторизатор Count — преобразует текст в матрицу количества лексем, представляя документы как векторы количества терминов, что подходит для базового анализа текста.

Word2Vec — Генерирует вкрапления слов путем обучения нейросетевой модели на корпусе текстов, фиксируя семантические связи между словами в виде плотного вектора.

Использованные алгоритмы:

Наивный Байесовский классификатор (MultinomialNB) — вероятностный классификатор, основанный на теореме Байеса с предположением о независимости между признаками.

Логистическая регрессия (Logistic Regression) — статистическая модель, оценивающая вероятности с помощью логистической функции.

Градиентный бустинг (Gradient Boosting) — ансамблевый метод, который последовательно строит модели, каждая новая модель исправляет ошибки, допущенные предыдущими.

Случайный лес (Random Forest) — метод ансамблевого обучения, который строит множество деревьев решений во время обучения и выдает класс, который является модой классов отдельных деревьев.

Использованные модели глубокого обучения:

Свёрточная нейронная сеть (CNN) — регуляризированный тип нейронной сети с прямой передачей данных, которая самостоятельно обучается построению признаков с помощью оптимизации фильтров.

DistillBERT — более компактная, быстрая, дешевая и легкая версия моделей архитектуры BERT. Требуется значительно меньше ресурсов для внедрения и использования с сохранением эффективности.

Результаты:

Модель	TopN Accuracy	Accuracy	MRR
MultinomialNB + TFIDF Vectorizer	0.703	0.6	0.642
MultinomialNB + Count Vectorizer	0.634	0.53	0.567
Logistic Regression + TFIDF Vectorizer	0.648	0.52	0.576
Gradient Boosting + TFIDF Vectorizer	0.507	0.41	0.447
Random Forest + TFIDF Vectorizer	0.611	0.5	0.547
CNN + Word2Vec	0.966	0.91	0.936
DistillBERT Transformer	0.916	0.85	0.876

Таблица 7 – Результаты исследования моделей

По итогам тестирования лучшей оказалась модель свёрточной нейронной сети в паре с векторизатором Word2Vec.

Ансамблевые методы показали негативную производительность. Это связано с тем, что они требуют большого количества ресурсов для построения ряда деревьев, которыми мы не располагали. Скромные же по размеру ансамбли показали слабую обобщающую способность.

Архитектура CNN: Приложение Г, рисунок 1

В итоге, получили мощную модель глубокого обучения, которая показала отличную производительность на тестовых данных. С её внедрением на 3 слой реализация поиска действительно становится “умной”.

4.2.3 Выводы

По итогам этапа мы получили ансамбль из 3 моделей, покрывающих весь спектр поведенческих шаблонов пользователей. Было проведено тестирование на валидационной выборке, составленной заказчиком. Её состав: 1588 реально сделанных заявок, которые были суммаризованы до поискового запроса и 1937 запросов для поиска по ключевым словам из БП.

Результаты:

Метрика	Значение
TopN Accuracy	0.903
Accuracy	0.78
Mean Reciprocal Rank	0.83

Таблица 8 – Результаты тестирования ансамбля

По таблице можно заметить, что умный поиск показывает отличную производительность, хотя и несколько уступающую тестовым показателям. Также можно заметить, что есть некоторые проблемы с ранжированием: ансамбль находит правильный БП в 90.3% случаев, но в выдаче он оказывается первым всего в 78%. Это будет являться одним из ключевых улучшений, предложенных заказчику при переходе на следующий этап.

Далее необходимо организовать доступ пользователя к разработанному ансамблю моделей.

Перейдём к разработке программного комплекса, описанной в Главе 5.

ГЛАВА 5. РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА

5.1 Постановка задачи на этап

5.1.1 Цель этапа

Реализация API, которое будет использовано системой 1С: ЦУП для определения БП по поисковому запросу в строке поиска на стороне 1С: ЦУП.

5.1.2 Требования к реализации

- 1) На этапе MVP программный комплекс должен реализовывать монолитную архитектуру. Однако при проектировании необходимо учесть возможность перехода к архитектуре модульный монолит – вынос логики предсказания БП в отдельный сервис на устройство, имеющее графический ускоритель.
- 2) Доступ к файлам моделей должен осуществляться через ClearML.
- 3) Для обработки текстов входящих запросов необходимо использовать универсальный модуль обработки текста.
- 4) Необходимо реализовать интерфейс для гибкого подключения любой модели к сервису. При этом разработчик алгоритма должен будет только предоставить архитектуру модели и её файлы, предобработка текста и фильтрация результатов является ответственностью сервиса.
- 5) В конфигурации сервиса должны быть доступны следующие параметры:
 - Использовать ли БД для сохранения отчётов? Да / Нет
 - Использовать ClearML для загрузки моделей? Да / Нет

- Название проекта ClearML
- Максимальное количество предсказаний, созданных сервисом
- Список моделей, входящих в ансамбль, и порядок их применения

5.1.3 Функциональные требования

API должен реализовывать следующие конечные точки:

1. GET /api/v1/ping – проверка работоспособности сервиса и его компонентов.

Заголовки запроса: нет.

Заголовки ответа: Приложение Д, рисунок 1

Тело ответа: нет.

2. GET /api/v1/business-processes – получение доступных для предсказания и оценки бизнес-процессов.

Заголовки запроса: нет.

Заголовки ответа: Приложение Д, рисунок 2

Тело ответа: Приложение Д, рисунок 3

3. POST /api/v1/predict – получение предсказаний бизнес-процессов по тексту запроса.

Заголовки запроса: нет.

Тело запроса: Приложение Д, рисунок 4

Заголовки ответа: Приложение Д, рисунок 5

Тело ответа: Приложение Д, рисунок 6

Диаграмма последовательности метода: Приложение Д, рисунок 7

4. POST /api/v1/save-mark – оценка имеющихся предсказаний модели и / или добавление новых бизнес-процессов для уточнения выдачи.

Заголовки запроса: нет.

Тело запроса: Приложение Д, рисунок 8

Заголовки ответа: Приложение Д, рисунок 9

Тело ответа: { }

Диаграмма последовательности метода: Приложение Д, рисунок 10

5.1.4 Нефункциональные требования

Среднее время отклика сервиса на конечную точку /api/v1/predict не должно превышать 0.5 секунд.

5.2 Проектирование

5.2.1 Стек технологий

В нашей компании используются следующие средства для разработки программных решений.

Среда разработки: PyCharm

СУБД: ClickHouse

Язык программирования: Python 3.10

Для реализации программного комплекса были выбраны следующие технологии. Это популярные фреймворки на языке Python для развёртывания API со сжатыми сроками. Также использованы технологии для интеграции моделей и алгоритмов, описанных в Главе 4.

Библиотеки и фреймворки: uvicorn, fastapi, pyyaml, sqlalchemy, sqlalchemy-clickhouse, pandas, clearml, pytorch, tensorflow, sentence-transformers, whoosh, faiss

5.2.2 Системная архитектура

Разработанный сервис размещён в Pod на одной из Node внутри кластера Kubernetes компании “BIA Technologies”, развёрнут внутри Docker Container. Он общается с базой данных ClickHouse, развёрнутой внутри Docker Container на сервере компании, для записи данных о запросах пользователей и ответах сервиса. Также сервис коммуницирует с ClearML, который также развёрнут в Kubernetes, для получения файлов моделей. У программного комплекса планируется 2 пользователя: 1С: “Центр Управления Предприятием” и WEB форма для разметки заказчиком ответов сервиса.

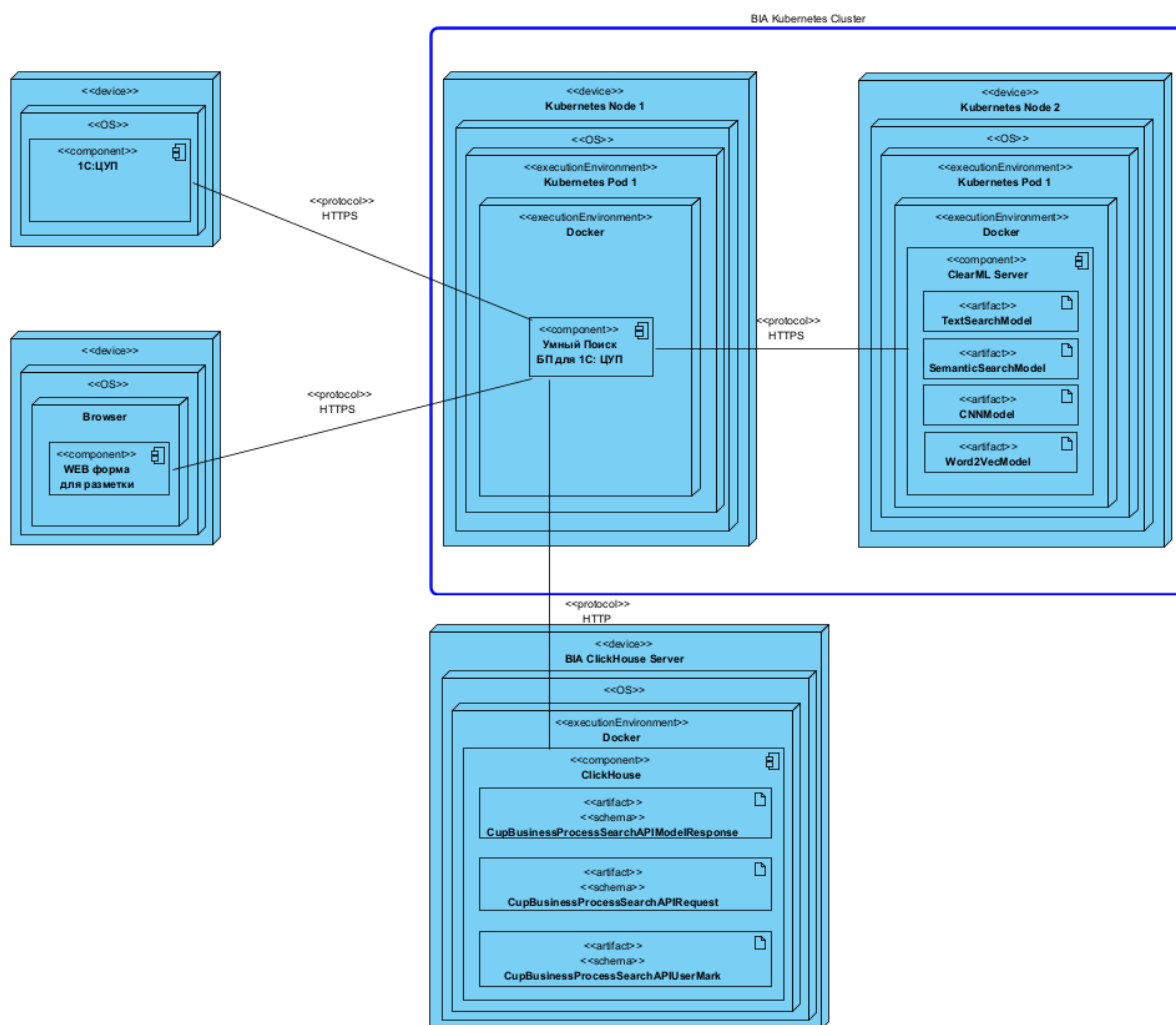


Рисунок 10 – Системная архитектура

5.2.3 Архитектура данных

В данном проекте таблицы в БД необходимы для сохранения отчётности.

- 1) CupBusinessProcessSearchAPIModelResponse – таблица, хранящая ответы ансамбля моделей на поступившие поисковые запросы.
- 2) CupBusinessProcessSearchAPIRequest – таблица, содержащая поисковые запросы пользователей к программному комплексу.
- 3) CupBusinessProcessSearchAPIUserMark – таблица, включающая результаты разметки заказчиком ответов сервиса.

CupBusinessProcessSearchAPIModelResponse	
RequestUUID	uuid
Timestamp	datetime
RowID	int8
BpUUID	uuid
BpGroupUUID	uuid
RequestTypeID	string
Confidence	float32
ModelType	string
ModelVersion	string

CupBusinessProcessSearchAPIUserMark	
RequestUUID	uuid
Timestamp	datetime
BpUUID	uuid
BpGroupUUID	uuid
RequestTypeID	string
Mark	int8
MarkType	string

CupBusinessProcessSearchAPIRequest	
RequestUUID	uuid
Timestamp	datetime
Text	string
UserUUID	uuid
UserName	string
OutputCount	int8
RequestSource	string

Рисунок 11 – Архитектура данных

5.2.4 Программная архитектура

На основании требований к реализации было принято решение разделить разрабатываемый сервис на 2 модуля, один из которых на этапе MVP будет подключаться к основному как библиотека.

Рабочее название основного модуля – CUP User Requests Service

Рабочее название модуля с моделями – CUP User Requests

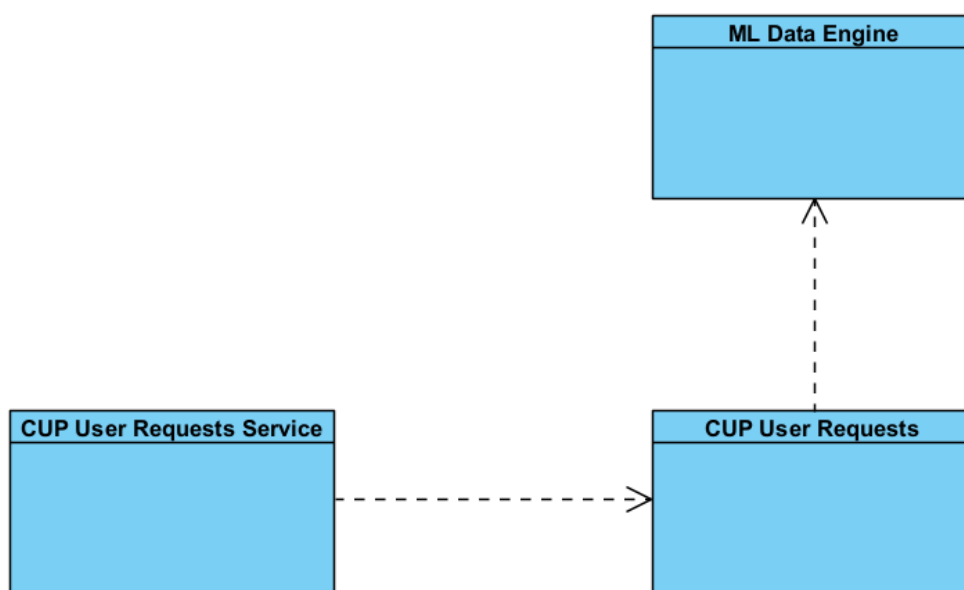


Рисунок 12 – Диграмма модулей программного комплекса

5.3 Разработка и тестирование

5.3.1 Реализация

Логика формирования предсказания сервисом описана в диаграмме активности формирования предсказания по тексту запроса: Приложение Д, рисунок 11

Для соответствия функциональным требованиям было разработано 2 программных компонента с наборами классов, которые можно рассмотреть на:

- 1) диаграмме классов CUP User Requests: Приложение Д, рисунок 12
- 2) диаграмме классов CUP User Requests Service: Приложение Д, рисунок 13

Была создана гибкая архитектура с возможностью подключения любой модели машинного обучения при помощи парадигмы объектно-ориентированного программирования и принципов SOLID.

Для демонстрации реализации компонентов были выбраны классы из каждого компонента модуля:

Реализация конечной точки `/api/v1/predict`: Приложение Д, рисунок 14

Реализация формирования ответа на `/api/v1/predict` в классе `ServiceBuilder`: Приложение Д, рисунок 15

Реализация логики метода `predict` в классе `Pipeline`: Приложение Д, рисунок 16

5.3.2 Тестирование

Мной был разработан ряд модульных тестов с использованием техник “разбиение на классы эквивалентности” и “анализ граничных значений” по методике “белый ящик”. Общее покрытие строк кода тестами составило 81%, отчёт о покрытии – Приложение Д, рисунок 17

Также был проведён нагрузочный системный тест на 6000 поисковых запросах, среднее время ответа сервиса составило 0.237 секунды, что удовлетворяет нефункциональному требованию.

5.3.3 Выводы

В ходе данного этапа был разработан программный комплекс с API для предсказания БП по текстовому запросу, который соответствует всем заявленным функциональным и нефункциональным требованиям, а также требованиям к реализации. Он будет внедрён в боевой контур компании, пройдёт системное тестирование и попадёт к конечному пользователю для сбора обратной связи.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной выпускной квалификационной работы был проведён полный цикл разработки комплекса программных компонентов: исследование данных, исследование моделей, проектирование и разработка.

Созданный сервис был протестирован на пользователях 1С: ЦУП и собрал положительные отзывы от сотрудников, точность предсказаний, собранная за время ОПЭ, соответствовала указанным в работе.

Метрика	Значение
TopN Accuracy	0.884
Accuracy	0.73
Mean Reciprocal Rank	0.79

Таблица 9 – Метрики сервиса за период ОПЭ

Заказчик был удовлетворён результатами и предложил продолжить сотрудничество на следующем этапе проекта. В качестве его развития были выдвинуты следующие предложения по системным функциям:

- 1) Автоматический сбор данных из боевых источников и обучение моделей на собранных актуальных данных.
- 2) Создание WEB формы для разметки заказчика.
- 3) Добавление мониторинга для отслеживания соответствия функциональным и нефункциональным требованиям

И доработкам для улучшения качества пользовательского опыта:

- 1) Добавление ранжирования предсказаний сервиса на основании статистики пользователей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Информация о ГК «Деловые Линии» [Электронный ресурс] — Режим доступа: <https://www.dellin.ru/company/>
2. Text generation web UI [Электронный ресурс] — Режим доступа: <https://github.com/oobabooga/text-generation-webui/>
3. Introduction to Whoosh [Электронный ресурс] — Режим доступа: <https://whoosh.readthedocs.io/en/latest/intro.html>
4. Okapi BM25 [Электронный ресурс] — Режим доступа: https://ru.wikipedia.org/wiki/Okapi_BM25
5. Faiss [Электронный ресурс] — Режим доступа: <https://github.com/facebookresearch/faiss>

ПРИЛОЖЕНИЕ А

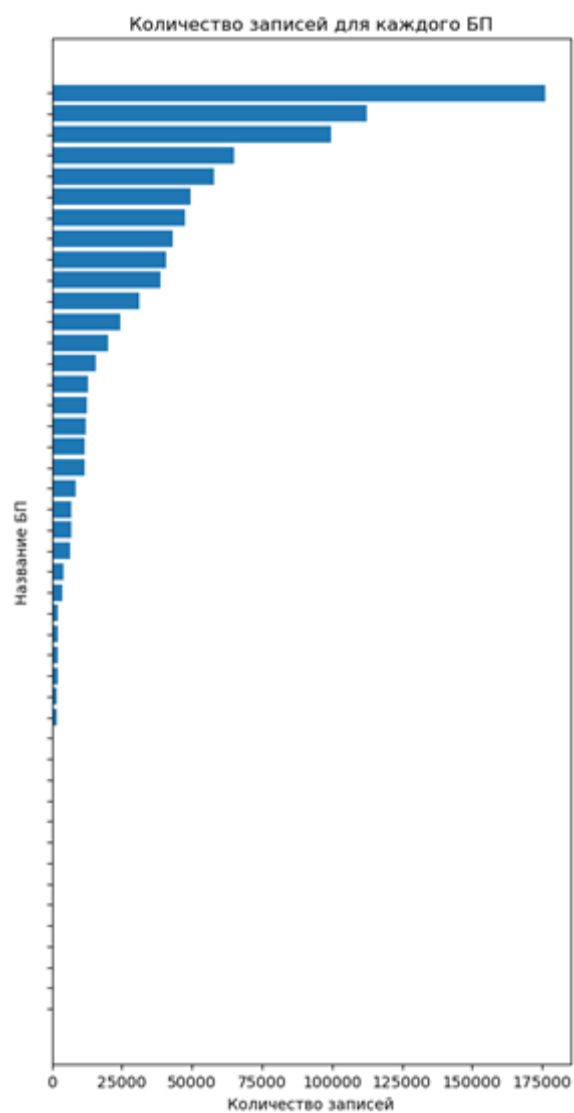


Рисунок 1 – Гистограмма распределение количества заявок в исторических данных по БП

ПРИЛОЖЕНИЕ Б

```
1 class Template:
2     def __init__(self, code: str, name: str, text: str, sep: str=';', user_input: str='<USER_INPUT>') -> list[str]:
3         self.keys = None
4         self.key_lens = None
5         self.all_tags = None
6         self.tag_idxs = None
7
8         self.sep = sep
9         self.user_input = user_input
10        self.code = code
11        self.name = name.replace('Шаблон: ', '').strip()
12
13        self.get_tags(text)
14        self.get_entity(text)
15
16    def get_tags(self, text: str) -> None:
17        self.all_tags = set(re.findall(r'<[A-Z_]{3,>', text))
18        tags = {tag: [i.span() for i in re.finditer(tag, text)] for tag in self.all_tags}
19        tags = dict(sorted({i: tag for tag in tags for i in tags[tag].items()}))
20        idxs = [e for e, v in enumerate(tags.values()) if v == self.user_input]
21        self.tag_idxs = idxs
22
23    def get_entity(self, text: str) -> None:
24        try:
25            self.keys = [i.strip() for i in re.split("|".join(self.all_tags), text) if i.strip()]
26            self.key_lens = [len(i) for i in self.keys if i]
27        except Exception as _:
28            pass
29
30    def get_keys(self) -> list | None:
31        return self.keys
32
33    def get_key_lens(self) -> list | None:
34        return self.key_lens
35
36    def get_tag_idxs(self) -> list | None:
37        return self.tag_idxs
```

Рисунок 1 – Код класса Template

```

1  class Extractor:
2      def __init__(self, templates: pd.DataFrame):
3          self.templates = self.get_templates(templates)
4
5      def get_templates(self, templates: pd.core.frame.DataFrame) -> list[Template]:
6          template_list = []
7          for row_index in range(len(templates)):
8              code, name, text = templates.loc[row_index]
9              template_list.append(Template(code, name, text))
10
11         return template_list
12
13     def match_items(self, text: str, keys: str) -> list[int]:
14         items = []
15         for key in keys:
16             index = text.find(key)
17             if index == -1:
18                 return items
19
20             text = text[index:]
21             items.append(items[-1] + index if items else index)
22
23         return items
24
25     def get_template_index(self, document: str) -> int | None:
26         for e, temp in enumerate(self.templates):
27             keys = temp.get_keys()
28             if not keys:
29                 continue
30
31             idxs = self.match_items(document, keys)
32             if len(keys)==len(idxs) and all(i >= 0 for i in idxs):
33                 return e
34
35         return None
36
37     def get_start_keys(self, text: str, keys: str) -> list[int]:
38         start_keys = []
39         for key in keys:
40             index = text.find(key)
41             text = text[index:]
42             start_keys.append(start_keys[-1] + index if start_keys else index)
43
44         return start_keys
45
46     def get_values(self, document: str) -> str | None:
47         index = self.get_template_index(document)
48         if index is None:
49             return None
50
51         template = self.templates[index]
52         keys = template.get_keys()
53         key_lens = template.get_key_lens()
54         tag_idxxs = template.get_tag_idxxs()
55         start_keys = self.get_start_keys(document, keys)
56         end_keys = [s + k - 1 for s, k in zip(start_keys, key_lens)]
57         start_value = [i + 1 for i in end_keys]
58         value_slice = [(k, v) for k, v in zip(start_value, start_keys[1:] + [len(document)])]
59         value = [document[k: v].strip() for e, (k, v) in enumerate(value_slice) if e in tag_idxxs]
60         value = " ".join([i for i in value if i])
61         return value

```

Рисунок 2 – Код класса Extractor

ПРИЛОЖЕНИЕ В

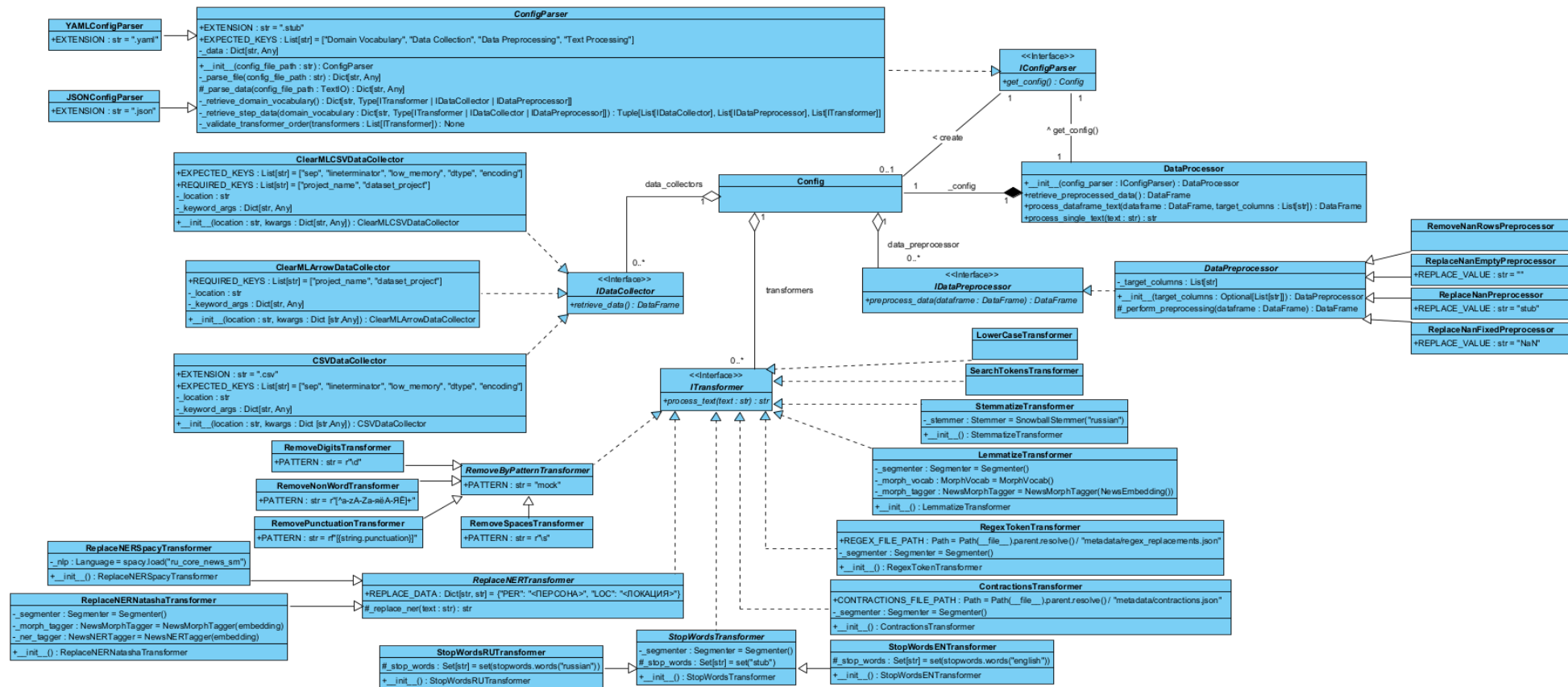


Рисунок 1 – Диаграмма классов ML Data Engin

```

1 class ClearMLCSVDDataCollector(IDataCollector):
2     EXPECTED_KEYS = ["sep", "lineterminator", "low_memory", "dtype", "encoding"]
3     REQUIRED_KEYS = ["project_name", "dataset_project"]
4
5     def __init__(self, location: str, **kwargs):
6         if any(key not in self.EXPECTED_KEYS + self.REQUIRED_KEYS for key in kwargs):
7             raise DataProcessorException("Встречен неожиданный keyword argument")
8
9         if any(key not in kwargs for key in self.REQUIRED_KEYS):
10            raise DataProcessorException("Не указан обязательный keyword argument")
11
12        self._location = location
13        self._keyword_args: Dict[str, Any] = kwargs
14
15    def retrieve_data(self) -> pd.DataFrame:
16        args = self._keyword_args.copy()
17        dataset_name = args.pop("project_name")
18        dataset_project = args.pop("dataset_project")
19        df_clear_ml = ClearMLDataset.get(dataset_name=dataset_name, dataset_project=dataset_project)
20        df_path = df_clear_ml.get_local_copy()
21        dataframe: pd.DataFrame = pd.read_csv(f"{df_path}/{self._location}", **args)
22        return dataframe

```

Рисунок 2 – Код класса ClearMLCSVDDataCollector

```

1 class ReplaceNanPreprocessor(DataPreprocessor):
2     REPLACE_VALUE = "stub"
3
4     def _perform_preprocessing(self, dataframe: pd.DataFrame) -> pd.DataFrame:
5         new_dataframe = dataframe.copy()
6         new_dataframe[self.target_columns] = new_dataframe[self.target_columns].fillna(self.REPLACE_VALUE)
7         return new_dataframe

```

Рисунок 3 – Код класса ReplaceNanPreprocessor

```

1 class LemmatizeTransformer(ITransformer):
2     def __init__(self):
3         self._segmenter = Segmenter()
4         self._morph_vocab = MorphVocab()
5         self._morph_tagger = NewsMorphTagger(NewsEmbedding())
6
7     def process_text(self, text: str) -> str:
8         doc = Doc(text)
9         doc.segment(self._segmenter)
10        doc.tag_morph(self._morph_tagger)
11        tokens = []
12        for token in doc.tokens:
13            token.lemmatize(self._morph_vocab)
14            tokens.append(token.lemma)
15
16        return " ".join(tokens)

```

Рисунок 4 – Код класса LemmatizeTransformer

Module	statements	missing	excluded	coverage
ml_data_engine__init__.py	3	0	0	100%
ml_data_engine\configuration__init__.py	4	0	0	100%
ml_data_engine\configuration\config_parser.py	101	1	0	99%
ml_data_engine\configuration\data__init__.py	0	0	0	100%
ml_data_engine\configuration\data\config.py	10	0	0	100%
ml_data_engine\configuration\interfaces__init__.py	0	0	0	100%
ml_data_engine\configuration\interfaces\i_config_parser.py	5	0	0	100%
ml_data_engine\configuration\json_config_parser.py	10	0	0	100%
ml_data_engine\configuration\yaml_config_parser.py	10	0	0	100%
ml_data_engine\data_collection__init__.py	4	0	0	100%
ml_data_engine\data_collection\clear_ml_arrow_data_collector.py	21	11	0	48%
ml_data_engine\data_collection\clear_ml_csv_data_collector.py	23	13	0	43%
ml_data_engine\data_collection\csv_data_collector.py	20	0	0	100%
ml_data_engine\data_collection\interfaces__init__.py	0	0	0	100%
ml_data_engine\data_collection\interfaces\i_data_collector.py	5	0	0	100%
ml_data_engine\data_preprocessing__init__.py	4	0	0	100%
ml_data_engine\data_preprocessing\data_preprocessor.py	17	1	0	94%
ml_data_engine\data_preprocessing\handle_nan__init__.py	0	0	0	100%
ml_data_engine\data_preprocessing\handle_nan\remove_nan_rows_preprocessor.py	7	0	0	100%
ml_data_engine\data_preprocessing\handle_nan\replace_nan_empty_preprocessor.py	3	0	0	100%
ml_data_engine\data_preprocessing\handle_nan\replace_nan_fixed_preprocessor.py	3	0	0	100%
ml_data_engine\data_preprocessing\handle_nan\replace_nan_preprocessor.py	8	0	0	100%
ml_data_engine\data_preprocessing\interfaces__init__.py	0	0	0	100%
ml_data_engine\data_preprocessing\interfaces\i_data_preprocessor.py	5	0	0	100%
ml_data_engine\data_processor.py	26	0	0	100%
ml_data_engine\text_processing__init__.py	13	0	0	100%
ml_data_engine\text_processing\contractions_transformer.py	28	0	0	100%
ml_data_engine\text_processing\interfaces__init__.py	0	0	0	100%
ml_data_engine\text_processing\interfaces\i_transformer.py	4	0	0	100%
ml_data_engine\text_processing\lemmatize_transformer.py	16	0	0	100%
ml_data_engine\text_processing\lower_case_transformer.py	4	0	0	100%
ml_data_engine\text_processing\regex_token_transformer.py	23	0	0	100%
ml_data_engine\text_processing\remove_by_pattern__init__.py	0	0	0	100%
ml_data_engine\text_processing\remove_by_pattern\remove_by_pattern_transformer.py	6	0	0	100%
ml_data_engine\text_processing\remove_by_pattern\remove_digits_transformer.py	3	0	0	100%
ml_data_engine\text_processing\remove_by_pattern\remove_non_word_transformer.py	3	0	0	100%
ml_data_engine\text_processing\remove_by_pattern\remove_space_transformer.py	3	0	0	100%
ml_data_engine\text_processing\replace_ner__init__.py	0	0	0	100%
ml_data_engine\text_processing\replace_ner\replace_ner_natasha_transformer.py	18	0	0	100%
ml_data_engine\text_processing\replace_ner\replace_ner_spacy_transformer.py	22	0	0	100%
ml_data_engine\text_processing\replace_ner\replace_ner_transformer.py	7	1	0	86%
ml_data_engine\text_processing\stemmatize_transformer.py	11	0	0	100%
ml_data_engine\text_processing\stop_words__init__.py	0	0	0	100%
ml_data_engine\text_processing\stop_words\stop_words_en_transformer.py	6	0	0	100%
ml_data_engine\text_processing\stop_words\stop_words_ru_transformer.py	6	0	0	100%
ml_data_engine\text_processing\stop_words\stop_words_transformer.py	12	0	0	100%
ml_data_engine\utils__init__.py	2	0	0	100%
ml_data_engine\utils\exceptions__init__.py	0	0	0	100%
ml_data_engine\utils\exceptions\data_processor_exception.py	10	0	0	100%
Total	486	27	0	94%

Рисунок 5 – Отчёт по покрытию модульными тестами ML Data Engine

ПРИЛОЖЕНИЕ Г

```
1 class CNNCore(nn.Module):
2     def __init__(self, model_config: Dict[str, Any], embedding_matrix: NDArray[Any]):
3         super(CNNCore, self).__init__()
4
5         self.embedding = nn.Embedding(
6             model_config["max_features"], model_config["embedding_length"]
7         )
8         self.embedding.weight = nn.Parameter(
9             torch.tensor(embedding_matrix, dtype=torch.float32)
10        )
11        self.embedding.weight.requires_grad = False
12        self.convs1 = nn.ModuleList(
13            [
14                nn.Conv2d(
15                    1,
16                    model_config["num_filters"],
17                    (K, model_config["embedding_length"]),
18                )
19                for K in model_config["filter_sizes"]
20            ]
21        )
22        self.dropout = nn.Dropout(model_config["dropout_rate"])
23        self.fc1 = nn.Linear(
24            len(model_config["filter_sizes"]) * model_config["num_filters"],
25            model_config["output_length"],
26        )
27
28    def forward(self, input_x: Tensor) -> Tensor:
29        output_x = self.embedding(input_x)
30        output_x = output_x.unsqueeze(1)
31        output_x = [nn_f.relu(conv(output_x)).squeeze(3)
32                     for conv in self.convs1]
33        output_x = [nn_f.max_pool1d(i, i.size(2)).squeeze(2) for i in output_x]
34        output_x = torch.cat(output_x, 1)
35        output_x = self.dropout(output_x)
36        return self.fc1(output_x) # type: ignore
```

Рисунок 1 – Класс с архитектурой CNN модели

ПРИЛОЖЕНИЕ Д

Пример

```
{
  api_version: "service_271023",
  model_1: "text_search: text_search_model_271023023",
  model_2: "sklearn: sklearn_model_271023023",
  ... (дальнейшие модели ансамбля)
}
```

Рисунок 1 – Заголовки ответа метода /api/v1/predict

Пример

```
{
  api_version: "service_271023"
}
```

Рисунок 2 – Заголовки ответа /api/v1/business-processes

Пример

```
[
  {
    "bp": {
      "id": "e892eacc-e1cb-11e5-999e-6eae8b6040f1",
      "name": "Заявка в службу (общая)"
    },
    "bp_group": {
      "id": "bcb49083-9642-11ed-819a-ccb971a2aa81",
      "name": "Согласование предоставления услуг FTL"
    }
  },
  {
    "bp": {
      "id": "461b2fe2-2897-11e0-b43f-001517531754",
      "name": "Установка цен номенклатуры"
    }
  },
  ... (другие бизнес-процессы)
]
```

Рисунок 3 - Тело ответа /api/v1/business-processes

Пример 1

```
{
  "request_UUID": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "text": "Установить светильник",
  "user": {
    "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "name": "Иванов Иван"
  },
  "output_count": 7,
  "request_source": "cup"
}
```

Пример 2

```
{
  "text": "Заявка в Административно-хозяйственный отдел"
}
```

Рисунок 13 – Тело запроса /api/v1/predict

Пример

```
{
  api_version: "service_271023",
  model_1: "text_search: text_search_model_271023023",
  model_2: "sklearn: sklearn_model_271023023",
  ... (дальнейшие модели ансамбля)
}
```

Рисунок 5 – Заголовки ответа /api/v1/predict

Пример

```
{
  "request_UUID": "e0efa6a1-f8a9-482c-8f11-15d6fc112a3c",
  "predictions": [
    {
      "row_id": 0,
      "bp": {
        "id": "e892eacc-e1cb-11e5-999e-6eae8b6040f1",
        "name": "Заявка в службу (общая)"
      },
      "bp_group": {
        "id": "86c7f3e5-a465-11e8-bf5a-42f2e90e1ed3",
        "name": "Заявка в Отдел по работе с производственным персоналом"
      }
    },
    {
      "row_id": 1,
      "bp": {
        "id": "c23e8a84-d9ea-11e0-847c-001a643625fe",
        "name": "Заявка в финансовую службу"
      },
      "request_types": [
        {
          "id": "07984a75-0ca5-11ec-81ce-005056834c43",
          "name": "Расчет ТЭО (оценка рентабельности клиента)"
        },
        {
          "id": "Премирование",
          "name": "Премирование"
        }
      ]
    }
  ],
  ... (дальнейшие ответы ансамбля)
}
```

Рисунок 6 – Тело ответа /api/v1/predict

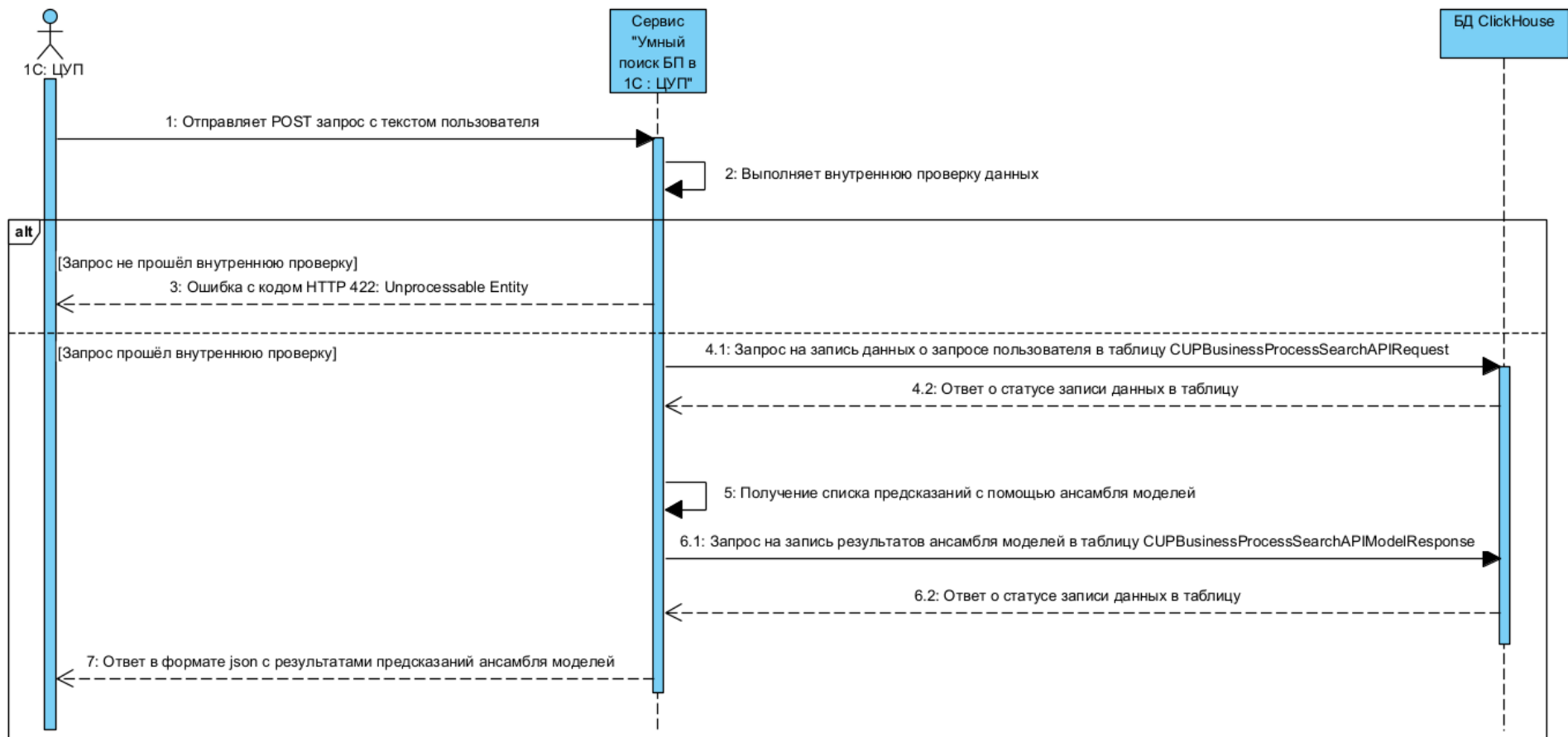


Рисунок 7 – Диаграмма последовательности /api/v1/predict

Пример для ответа predict из прошлого пункта

```
{
  "request_UUID": "e0efa6a1-f8a9-482c-8f11-15d6fc112a3c",
  "marks": [
    {
      "bp_id": "c23e8a84-d9ea-11e0-847c-001a643625fe",
      "request_type_id": "Премирование",
      "mark": 1,
      "mark_type": "model"
    },
    {
      "bp_id": "c83cf008-05da-11e5-8d31-6eae8b603f99",
      "mark": 3,
      "mark_type": "custom"
    }
  ]
}
```

Рисунок 8 – Тело запроса /api/v1/save-mark

Пример

```
{
  api_version: "service_271023"
}
```

Рисунок 9 – Заголовок ответа /api/v1/save-mark

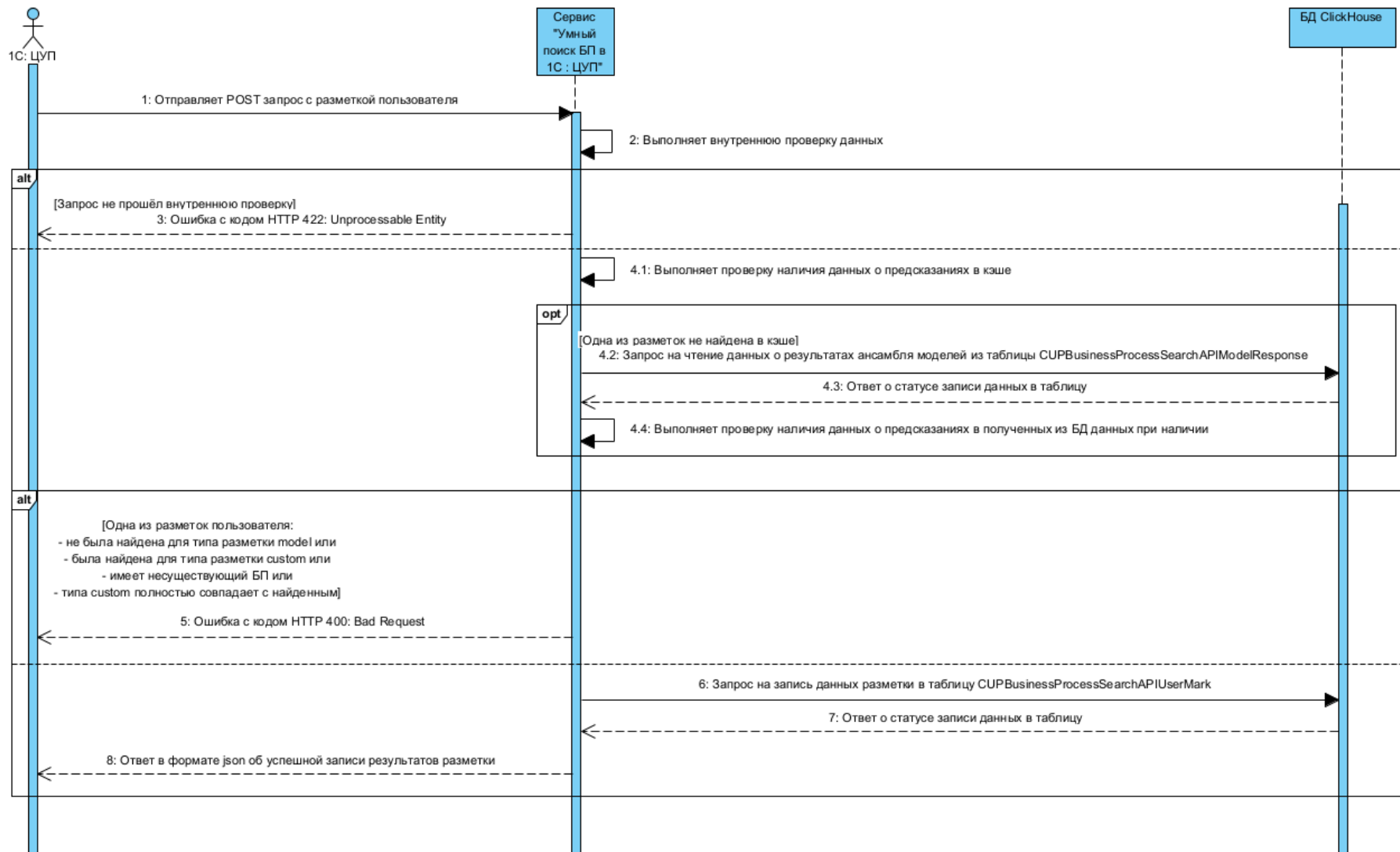


Рисунок 10 – Диаграмма последовательности /api/v1/save-mark

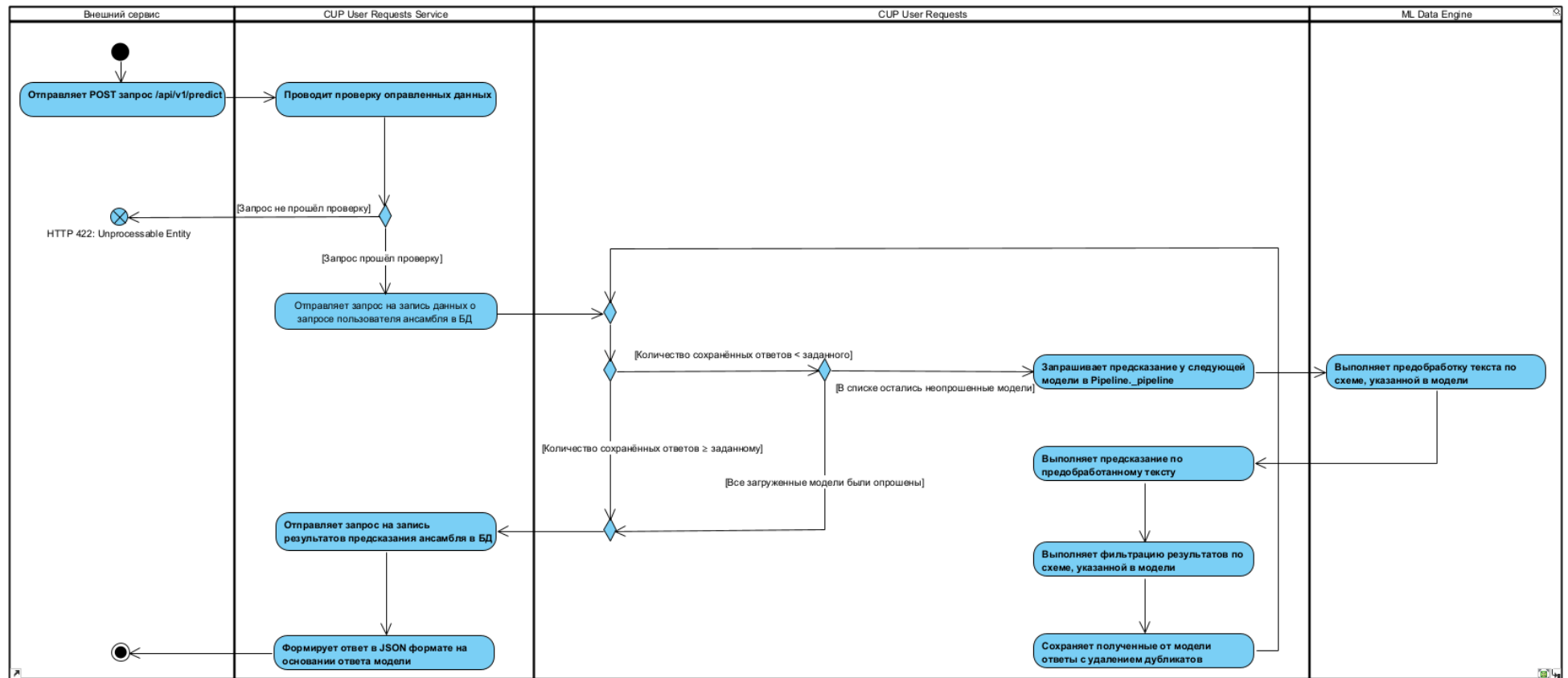


Рисунок 11 - Диаграмма активности формирования предсказания

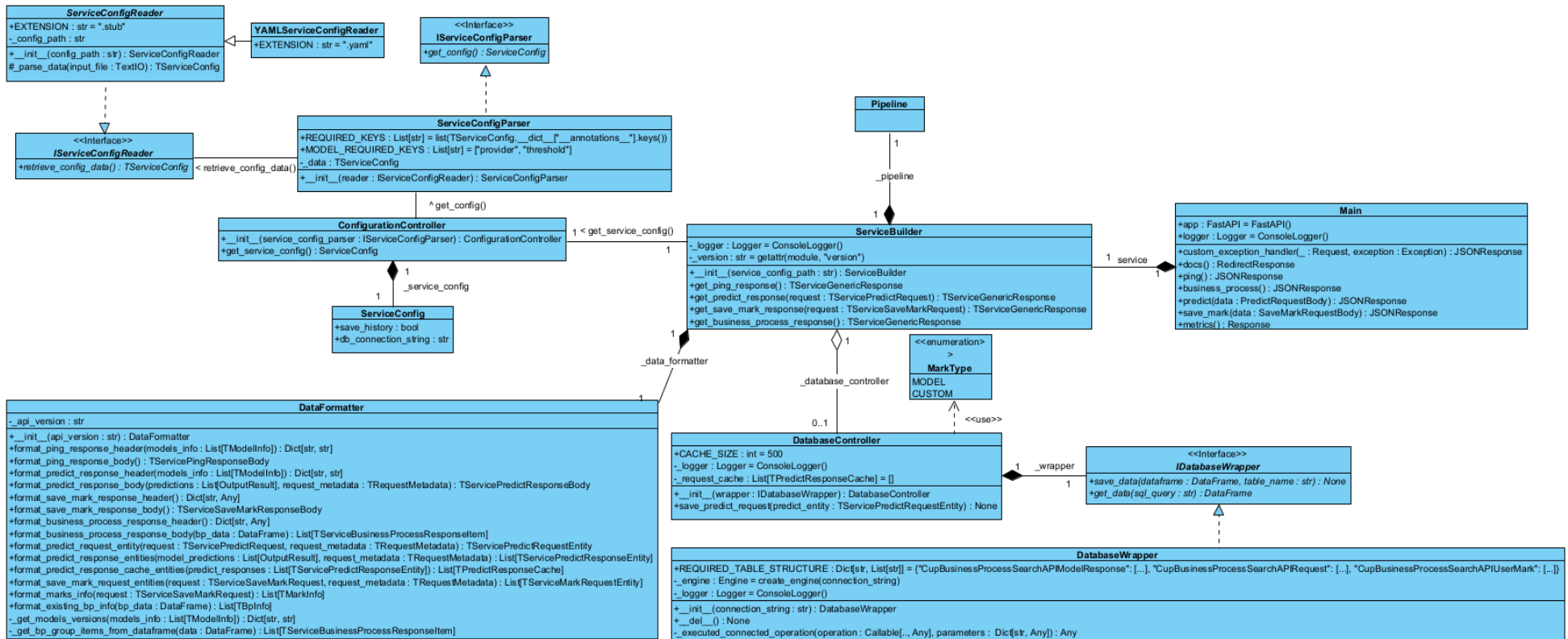


Рисунок 13 - Диграмма классов CUP User Requests Service

```

1  @app.post(
2      "/api/v1/predict",
3      description="Получение предсказанного списка БП по введённому тексту",
4      response_description="Список предсказанных БП, групп БП, видов заявок с UUID",
5      response_model=PredictResponseBody,
6      tags=["API"],
7      status_code=200
8  )
9  async def predict(data: PredictRequestBody) -> JSONResponse:
10     logger.info(f"Пришёл запрос predict с телом: {data}")
11     parameters = {
12         "request_uuid": str(data.request_UUID),
13         "text": data.text,
14         "user": None if data.user is None else data.user.model_dump(mode="json"),
15         "output_count": data.output_count,
16         "request_source": data.request_source
17     }
18     header, body = service.get_predict_response(parameters).values()
19     return JSONResponse(content=body, headers=header)

```

Рисунок 14 – Код конечной точки /api/v1/predict

```

1  def get_predict_response(self, request: TServicePredictRequest) -> TServiceGenericResponse:
2     request_metadata: TRequestMetadata = {
3         "request_uuid": request["request_uuid"],
4         "timestamp": datetime.datetime.now()
5     }
6     if self._database_controller is not None:
7         request_entity = self._data_formatter.format_predict_request_entity(request, request_metadata)
8         self._database_controller.save_predict_request(request_entity)
9
10    predictions = self._pipeline.predict(request["text"], request["output_count"])
11    self._logger.info(f"Получен ответ от ансамбля: {predictions}")
12    models_info: List[TModelInfo] = self._pipeline.get_model_info() # type: ignore
13    predict_body = self._data_formatter.format_predict_response_body(predictions, request_metadata)
14    if self._database_controller is not None:
15        response_entities = self._data_formatter.format_predict_response_entities(predictions, request_metadata)
16        cache_entities = self._data_formatter.format_predict_response_cache_entities(response_entities)
17        self._database_controller.save_predict_response(response_entities)
18        self._database_controller.cache_predict_response(cache_entities)
19
20    response: TServiceGenericResponse = {
21        "header": self._data_formatter.format_predict_response_header(models_info),
22        "body": predict_body
23    }
24    self._logger.info(f"Сформирован ответ на predict: {response}")
25    return response

```

Рисунок 15 – Формирование ответа на /api/v1/predict в классе ServiceBuilder

```

1 def predict(self, text: str, output_count: Optional[int] = None) -> List[OutputResult]:
2     prediction: List[ModelPrediction] = []
3     processed_keys: Set[Tuple[str, Optional[str]]] = set()
4     processed_bp: Set[str] = set()
5
6     stop_prediction = False
7     for model in self._pipeline:
8         if stop_prediction:
9             break
10        model_prediction: List[ModelPrediction] = model.predict(text)
11        for row in model_prediction:
12            row_key = (row.bp_code, row.request_type_id)
13            if row_key in processed_keys:
14                continue
15            prediction.append(row)
16            processed_keys.add(row_key)
17            processed_bp.add(row.bp_code)
18            if not row.is_search and len(processed_bp) >= self._config.common_properties.model_output_count:
19                stop_prediction = True
20                break
21
22    return self._output_processor.process_result(prediction, output_count)

```

Рисунок 16 – Логика метода predict в классе Pipeline

Coverage report: 81%

coverage.py v7.3.2, created at 2023-11-24 19:05 +0300

Module	statements	missing	excluded	coverage
model_service__init__.py	2	0	0	100%
model_service\configuration__init__.py	0	0	0	100%
model_service\configuration\configuration_controller.py	11	0	0	100%
model_service\configuration\parsers__init__.py	0	0	0	100%
model_service\configuration\parsers\interfaces__init__.py	0	0	0	100%
model_service\configuration\parsers\interfaces\i_service_config_parser.py	5	0	0	100%
model_service\configuration\parsers\service_config_parser.py	32	0	0	100%
model_service\configuration\readers__init__.py	0	0	0	100%
model_service\configuration\readers\interfaces__init__.py	0	0	0	100%
model_service\configuration\readers\interfaces\i_service_config_reader.py	5	0	0	100%
model_service\configuration\readers\yaml_service_config_reader.py	11	0	0	100%
model_service\custom_typing__init__.py	11	0	0	100%
model_service\custom_typing\bp_info_type.py	4	0	0	100%
model_service\custom_typing\endpoint_types__init__.py	5	0	0	100%
model_service\custom_typing\endpoint_types\mark_type.py	7	0	0	100%
model_service\custom_typing\endpoint_types\prediction_type.py	8	0	0	100%
model_service\custom_typing\endpoint_types\request_types__init__.py	3	0	0	100%
model_service\custom_typing\endpoint_types\request_types\service_predict_request_type.py	8	0	0	100%
model_service\custom_typing\endpoint_types\request_types\service_save_mark_request_type.py	5	0	0	100%
model_service\custom_typing\endpoint_types\response_types__init__.py	5	0	0	100%
model_service\custom_typing\endpoint_types\response_types\service_business_process_response_item_type.py	7	0	0	100%
model_service\custom_typing\endpoint_types\response_types\service_ping_response_body_type.py	3	0	0	100%
model_service\custom_typing\endpoint_types\response_types\service_predict_response_body_type.py	5	0	0	100%
model_service\custom_typing\endpoint_types\response_types\service_save_mark_response_body_type.py	3	0	0	100%
model_service\custom_typing\endpoint_types\service_generic_response_type.py	4	0	0	100%
model_service\custom_typing\entity_types__init__.py	4	0	0	100%
model_service\custom_typing\entity_types\service_mark_request_entity_type.py	10	0	0	100%
model_service\custom_typing\entity_types\service_predict_request_entity_type.py	10	0	0	100%
model_service\custom_typing\entity_types\service_predict_response_entity_type.py	12	0	0	100%
model_service\custom_typing\mark_cache_type.py	6	0	0	100%
model_service\custom_typing\mark_info_type.py	7	0	0	100%
model_service\custom_typing\model_info_type.py	4	0	0	100%

Рисунок 17 - Отчёт по покрытию модульными тестами CUP User Requests Service