



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

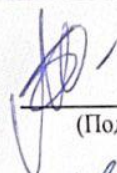
КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.03 ПРИКЛАДНАЯ ИНФОРМАТИКА

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ НА ТЕМУ:

«Интеллектуальная система анализа отзывов»

Студент ИУ6-54Б  
(Группа)

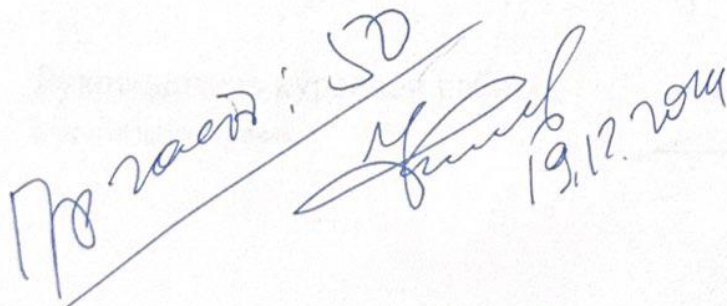
  
(Подпись, дата)

А. И. Мокшина  
(И.О.Фамилия)

Руководитель курсовой работы  
Старший преподаватель

  
(Подпись, дата)

М. В. Фетисов  
(И.О. Фамилия)

  
19.12.2024

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ6  
(Индекс)  
А.В. Пролетарский  
(И.О. Фамилия)  
« 16 » сентября 2024 г.

**ЗАДАНИЕ**  
**на выполнение курсовой работы**

по дисциплине Технология разработки программных систем

Студент группы ИУ6-54Б

Мокшина Анастасия Игоревна  
(Фамилия, имя, отчество)

Тема курсовой работы Интеллектуальная система анализа  
отзывов

Направленность КР (учебная, исследовательская, практическая, производственная, др.)  
учебная

Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения КР: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

Техническое задание см. техническое задание в приложении А

**Оформление курсовой работы:**

1. Расчетно-пояснительная записка (РПЗ) на 25-30 листах формата А4.
2. Техническое задание на 5-9 листах формата А4 – оформляется в качестве приложения А к РПЗ.
3. Руководство пользователя на 6-8 листах формата А4 – оформляется в качестве приложения Б к РПЗ.
4. Графический и иллюстративный материал оформляется в виде рисунков и помещается в РПЗ.
5. РПЗ загрузить на страницу дисциплины на сайте кафедры не позднее, чем накануне защиты.
6. После защиты заменить титул РПЗ на скан титула РПЗ с оценкой комиссии и заново загрузить РПЗ в формате .pdf на страницу дисциплины.

Дата выдачи задания « 2 » сентября 2024 г.

Руководитель курсовой работы

Студент

М.В. Петисов  
(Подпись, дата) (И.О. Фамилия)  
А.И. Мокшина  
(Подпись, дата) (И.О. Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

## РЕФЕРАТ

Записка 38 с., 3 ч., 13 рис., 11 табл., 9 источников, 4 прил.

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА, АНАЛИЗ ОТЗЫВОВ, МАШИННОЕ ОБУЧЕНИЕ, ТОНАЛЬНОСТЬ ТЕКСТА, POSTGRESQL, PYTHON, ОБРАБОТКА ДАННЫХ, ВИЗУАЛИЗАЦИЯ ДАННЫХ.

Объектом разработки является интеллектуальная система анализа отзывов (ИСаО), предназначенная для автоматизации сбора, обработки и анализа отзывов об автозаправочных станциях (АЗС).

Цель работы – создание системы, способной эффективно выявлять тональность отзывов, выделять ключевые темы и визуализировать результаты, что позволит предприятиям улучшать качество своих услуг на основе обратной связи.

В результате разработки была спроектирована и реализована автоматизированная система, включающая модули сбора данных, их обработки и анализа (определение тональности, выявление ключевых тем) с применением алгоритмов машинного обучения. Система сохраняет обработанные данные в базе данных PostgreSQL и предоставляет результаты через пользовательский веб-интерфейс, построенный на Streamlit.

Область применения разработанной системы – предприятия, работающие в сфере услуг и желающие улучшить качество обслуживания на основе анализа клиентских отзывов.

## СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ .....	5
ВВЕДЕНИЕ .....	6
1 Анализ требований и уточнение спецификаций .....	7
1.1 Анализ задания и выбор технологии, языка и среды разработки.....	7
1.2 Выбор модели жизненного цикла программного обеспечения .....	7
1.3 Построение диаграммы вариантов использования .....	8
1.4 Построение концептуальной модели предметной области.....	11
1.5 Выбор методов и алгоритмов решения задачи.....	12
2 Проектирование структуры и компонентов программного продукта.....	16
2.1 Проектирование даталогической модели данных .....	16
2.2 Проектирование структуры программного продукта .....	18
2.3 Разработка интерфейса пользователя.....	21
2.3.1 Проектирование пользовательского интерфейса .....	21
2.3.2 Построение графа состояний интерфейса .....	23
2.4 Построение диаграмм последовательности .....	24
3 Выбор стратегии тестирования и разработка тестов .....	29
3.1 Функциональное тестирование .....	29
3.2 Оценочное тестирование .....	30
3.2.1 Тестирование надежности .....	30
3.2.2 Тестирование удобства использования и эксплуатации .....	36
3.2.3 Тестирование на предельных объемах .....	36
ЗАКЛЮЧЕНИЕ.....	38
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	39
ПРИЛОЖЕНИЕ А. Техническое задание .....	40
ПРИЛОЖЕНИЕ Б. Фрагмент исходного кода.....	49
ПРИЛОЖЕНИЕ В. Руководство администратора системы .....	55
ПРИЛОЖЕНИЕ Г. Руководство пользователя .....	60

## ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БД	—	база данных.
ИСаО	—	интеллектуальная система анализа отзывов.
Парсер	—	часть программы, преобразующей входные данные в некий структурированный формат, нужный для задач последующего их (данных) анализа и использования. Технически, парсер выполняет синтаксический анализ данных (например, текста).
ТЗ	—	техническое задание.
API	—	интерфейс программирования приложений, набор инструментов для взаимодействия различных программных компонентов.
CSV	—	формат текстового файла для хранения данных в виде таблиц с разделением значений запятыми.
JSON	—	легковесный формат обмена данными, удобный для хранения и передачи текстовой информации.
SVM	—	метод опорных векторов, алгоритм машинного обучения, используемый для классификации и регрессии.

## **ВВЕДЕНИЕ**

Работа посвящена проектированию и разработке интеллектуальной системы анализа отзывов.

Актуальность разработки обоснована важностью отзывов клиентов для формирования репутации и повышения конкурентоспособности бизнеса. Отзывы на платформах, таких как Яндекс Карты, имеют значительное влияние на восприятие бренда и решения потенциальных клиентов. Автоматизация процессов сбора и анализа отзывов обеспечит возможность оперативно реагировать на обратную связь, выявлять как сильные, так и слабые стороны компании, а также принимать взвешенные управленческие решения для улучшения качества обслуживания.

В основе системы лежат передовые методы обработки данных, включая машинное обучение, которые позволяют автоматически определять тональность отзывов и выделять основные темы.

ИСаО отличается от аналогичных решений тем, что сочетает автоматический сбор отзывов, глубокий аналитический анализ данных с использованием технологий машинного обучения и удобный интерфейс для визуализации и интерпретации результатов. В отличие от других систем, ограничивающихся только анализом уже собранных данных, ИСаО представляет собой комплексное решение, идеально подходящее для оптимизации бизнес-процессов.

## **1 Анализ требований и уточнение спецификаций**

### **1.1 Анализ задания и выбор технологии, языка и среды разработки**

Техническое задание (ТЗ) определяет ключевые требования к разработке интеллектуальной системы анализа отзывов (ИСаО), основной целью которой является автоматизация процесса сбора, анализа и визуализации отзывов с Яндекс Карт. Подробнее с ТЗ можно ознакомиться в приложении А. Система должна выполнять следующие функции:

- автоматический сбор отзывов с API Яндекс Карт;
- анализ отзывов с использованием методов машинного обучения (выявление тональности, ключевых тем);
- экспорт аналитических данных и графиков.

Для реализации ИСаО выбран язык программирования Python. Этот выбор обусловлен наличием большого количества библиотек и инструментов для работы с данными [1] (pandas, NumPy), обработки текста (NLTK [2], Natasha), машинного обучения (scikit-learn [3], Word2Vec [4]); доступностью модулей для взаимодействия с API (requests).

PostgreSQL выбрана в качестве СУБД благодаря её высокой производительности, поддержке сложных запросов и возможностям масштабирования, что позволяет эффективно обрабатывать большие объёмы данных. Для взаимодействия с базой данных будет использоваться библиотека psycopg2. Эта библиотека поддерживает асинхронные операции, что позволяет оптимизировать процессы извлечения и обработки данных.

Для реализации пользовательского интерфейса был выбран инструмент Streamlit. Это библиотека Python, которая позволяет создавать интерактивные веб-приложения.

Для разработки будет использоваться Visual Studio Code благодаря поддержке расширений, автодополнению кода и интеграции с системами контроля версий. В качестве системы контроля версий был выбран Git.

### **1.2 Выбор модели жизненного цикла программного обеспечения**

Выбор модели жизненного цикла программного обеспечения является ключевым этапом проектирования.

Для данного проекта, связанного с созданием интеллектуальной системы анализа отзывов (ИСаО), выбрана итеративная модель разработки ПО.

Итеративная модель позволяет постепенно разрабатывать и дорабатывать систему, адаптируя её к изменяющимся требованиям и полученным данным.

Разделение проекта на итерации помогает выявить и устранить проблемы на ранних этапах, а также обеспечивает возможность изменений в дальнейшем.



Каждая итерация заканчивается созданием работающего прототипа или функционального компонента, что позволяет получить обратную связь от заинтересованных сторон на ранних стадиях.

Система включает такие компоненты, как интеграция с API, обработка данных, машинное обучение и визуализация. Итеративный подход позволяет поэтапно разрабатывать и тестировать эти модули.

Тестирование осуществляется на каждом этапе, что гарантирует качество программного обеспечения.

Итеративный подход идеально соответствует специфике разработки ИСАО, так как позволяет:

- начать с базового набора функций (например, сбор и сохранение данных);
- постепенно добавлять сложные функции, такие как анализ тональности и ключевых тем;
- постоянно улучшать систему, учитывая требования заказчика или новые технологические возможности.

Выбор итеративной модели обеспечивает эффективное управление проектом, минимизацию рисков и создание качественного продукта, соответствующего потребностям пользователей.

### **1.3 Построение диаграммы вариантов использования**

Диаграмма вариантов использования помогает отобразить, как пользователи и администраторы взаимодействуют с приложением. Она является первым шагом проектирования системы, определяя функциональные возможности приложения с точки зрения его пользователей [5].

После анализа ТЗ были получены следующие варианты взаимодействия пользователя, администратора и приложения:

- подключение к базе данных через пользовательский интерфейс;
- запуск процесса выполнения автоматического анализа новых отзывов;
- просмотр таблицы с результатами анализа отзывов;
- обновление базы данных отзывов;
- просмотр автоматически сгенерированных графиков;
- экспорт полученных графиков в формате PDF;
- запуск процесса сбора данных из Яндекс Карт;
- запуск процесс загрузки данных в БД;



- управление базой данных;
- изменение параметров сбора данных через конфигурационные файлы;
- ручная разметка данных для обучения модели.

Рассмотрим основные варианты использования в таблицах 1-4:

Таблица 1 – Описание варианта использования «Просмотр таблицы с результатами анализа отзывов»

Название варианта	Просмотр таблицы с результатами анализа отзывов
Цель	Просмотреть таблицу с выделенными ключевыми словами и тональностью отзывов
Действующие лица	Пользователь
Краткое описание	Пользователь запускает приложение, подключается к базе данных и имеет возможность просмотреть результаты анализа данных
Тип	Основной

Таблица 2 – Описание варианта использования «Запуск процесса сбора данных»

Название варианта	Запуск процесса сбора данных
Цель	Собрать данные о точках АЗС и связанных отзывах
Действующие лица	Администратор
Краткое описание	Администратор запускает Python-скрипт, который собирает данные об АЗС и связанных с ними отзывах. Данные сохраняются в виде файла для дальнейшей обработки
Тип	Основной

Таблица 3 – Описание варианта использования «Ручная разметка данных для обучения модели»

Название варианта	Ручная разметка данных для обучения модели
Цель	Создать размеченный набор данных для обучения модели анализа тональности.
Действующие лица	Администратор
Краткое описание	Администратор использует Jupyter Notebook для открытия файла с отзывами, вручную проставляет метки для тональности и ключевых слов, а затем сохраняет размеченные данные в файл.
Тип	Основной

По результатам анализа была реализована диаграмма вариантов использования, представленная на рисунке 1, которая позволяет более точно определить способы взаимодействия пользователя и администратора с системой.

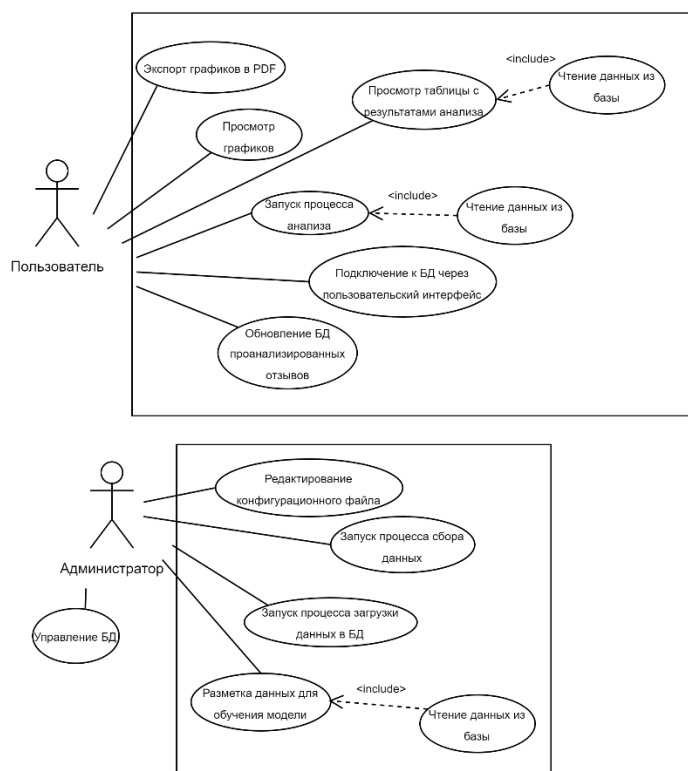


Рисунок 1 – Диаграмма вариантов использования

Представленная диаграмма вариантов использования наглядно демонстрирует, какие функции доступны пользователям, а также показывает, какую цепочку действий нужно пройти, чтобы воспользоваться нужной функцией.

#### **1.4 Построение концептуальной модели предметной области**

Для построения концептуальной модели системы анализа отзывов об автозаправочных станциях (АЗС), можно определить множество понятий-кандидатов: АЗС, отзыв, парсер, загрузчик, база данных, пользователь, модель анализа тональности, модель выделения ключевых слов, структура данных.

Цель основного варианта использования системы — анализ отзывов для получения ценной информации об АЗС. Основные понятия можно выделить следующим образом.

Отзыв — ключевой объект системы, представляющий текст, оставленный пользователем. Связан с другими понятиями, такими как Тональность, Ключевые слова, Пользователь, АЗС и Модели анализа.

АЗС — представляет автозаправочную станцию, к которой относится отзыв.

Парсер — компонент, отвечающий за извлечение данных из внешних источников.

Загрузчик — компонент, обеспечивающий запись извлеченных данных в базу данных.

База данных — хранилище всех данных системы, включая отзывы, информацию об АЗС, результаты анализа тональности и выделенные ключевые слова.

Модели анализа — включают модель анализа тональности и модель выделения ключевых слов, которые принимают текст отзыва как входные данные и возвращают результаты анализа.

Тональность — характеристика отзыва, которая может быть положительной, нейтральной или негативной. Тональность вычисляется моделью анализа.

Ключевые слова — значимые термины, извлеченные из текста отзыва, отражающие основные темы, обсуждаемые в отзыве. Это результат работы модели выделения ключевых слов.

Парсер можно разделить на два подкласса: Парсер АЗС и Парсер отзывов. Аналогично, Загрузчик имеет подклассы: Загрузчик АЗС и Загрузчик отзывов.

Парсер извлекает данные и передаёт их Загрузчику, который отвечает за загрузку информации в базу данных.

Основной класс-понятие — Отзыв. Он связан с классом АЗС через базу данных, а также с классом Модели анализа. Модели анализа обрабатывают текст отзыва, вычисляют тональность и выделяют ключевые слова.

Концептуальная модель также отражает иерархическую структуру компонентов системы, позволяющую обеспечить гибкость и масштабируемость при добавлении новых функций. Например, расширение системы для анализа отзывов на других языках или интеграции с альтернативными источниками данных может быть выполнено путем добавления новых подклассов Парсера и Моделей анализа.

На основе анализа была построена концептуальная модель предметной области, представленная на рисунке 2.

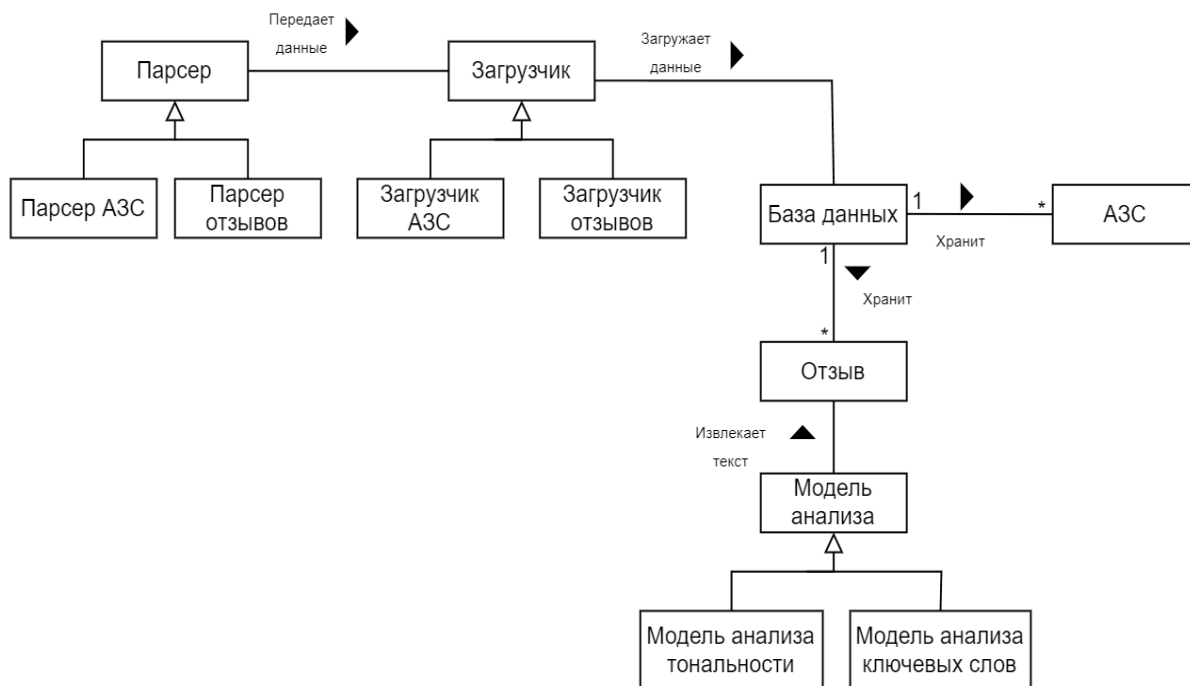


Рисунок 2 – Концептуальная модель предметной области

Таким образом, концептуальная модель отражает взаимосвязь понятий, позволяет формализовать систему анализа отзывов об АЗС и обеспечивает её разработку на основе логической структуры.

### 1.5 Выбор методов и алгоритмов решения задачи

В системе анализа отзывов для автозаправочных станций (АЗС) основными задачами являются анализ тональности отзывов (классификация на положительные, нейтральные и негативные) и выделение ключевых слов для понимания основных тем и проблем, обсуждаемых в отзывах.

Для решения этих задач были выбраны методы и алгоритмы, которые обеспечивают высокую точность, эффективную обработку больших объемов данных и адаптируемость к различным типам текстов.

Для эффективной работы с текстовыми данными на русском языке была выбрана Naves. Naves [6] — это предобученная модель для работы с текстами на русском языке, которая основана на алгоритме Word2Vec. Она использует принцип векторизации слов, что позволяет представлять каждое слово как точку в многомерном пространстве, где слова с похожими значениями или контекстами располагаются ближе друг к другу. Она представляет собой модель векторных представлений слов, обученную на большом корпусе текстов на русском языке.

Преимущество использования Naves заключается в том, что она уже содержит информацию о семантических и контекстуальных связях между словами, что позволяет эффективно работать с текстами, не требуя обучения модели с нуля.

Однако, несмотря на преимущества предобученной модели, в нашем случае существует необходимость адаптировать её под специфический контекст наших данных — отзывов об автозаправочных станциях (АЗС).

В процессе работы могут встречаться слова или выражения, которые не были учтены в корпусе, на котором обучалась модель Naves. Чтобы повысить точность векторных представлений в контексте конкретных отзывов, было решено дообучить модель Word2Vec на собственных текстах.

Процесс дообучения модели на новом корпусе текстов позволяет адаптировать их к специфике используемой лексики, сохраняя при этом все преимущества предобученной модели.

Мы будем использовать модель Naves как основу, а затем дообучим её на наших данных с помощью Word2Vec. Это объединение предобученной модели и адаптированной версии позволит нам извлечь максимальную пользу из обоих подходов.

Для объединения векторов создадим хэш-таблицу, содержащую слова и их векторные представления. Важно помнить о том, что необходимо согласовать размерность векторных представлений, чтобы обеспечить совместимость между моделями Naves и Word2Vec, а также гарантировать корректность их объединения.

При добавлении новых слов из Word2Vec учитывались только те, которых не было в Naves, что позволяет сохранить уникальность информации из каждой модели.

Теперь, чтобы более детально описать процесс интеграции этих моделей, а также алгоритм их дообучения, на рисунке 3 отобразим схему алгоритма, которая иллюстрирует процесс дообучения модели Word2Vec с использованием исходной модели Naves.

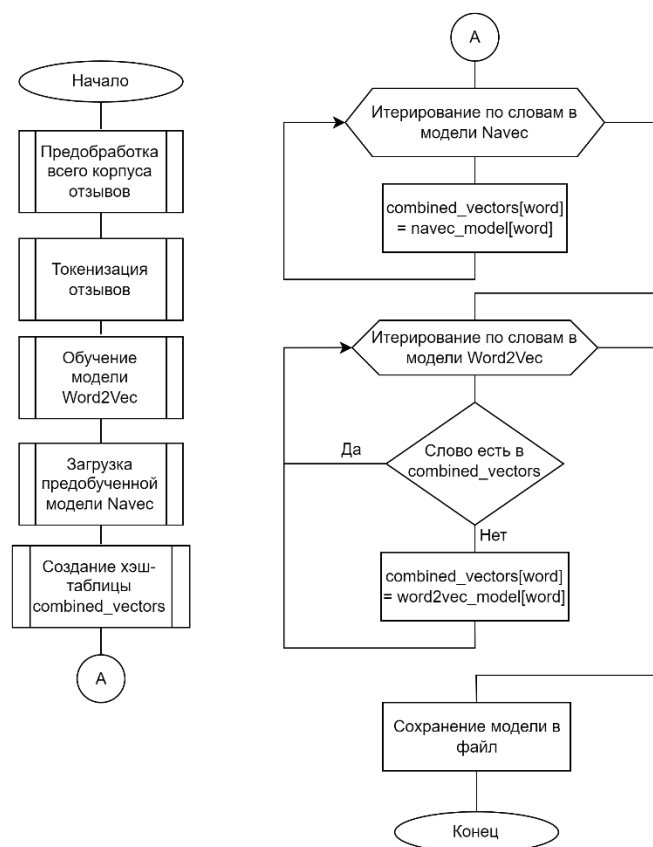


Рисунок 3 – Схема алгоритма дообучения модели

Для представления отзывов в виде матрицы каждый токен (слово) из отзыва необходимо преобразовать в векторное представление с использованием предварительно объединённой модели, созданной на основе Navec и Word2Vec. Если токен присутствует в модели, его вектор извлекается напрямую, в противном случае используется специальный вектор <unk>, который представляет неизвестные или редкие слова. Это позволяет обеспечить работу модели даже с уникальными выражениями, ранее не встречавшимися в данных.

Для того чтобы сделать представление отзывов унифицированным, было установлено ограничение на максимальную длину отзыва. Отзывы, длина которых превышает заданный предел, обрезаются до фиксированного количества слов. Напротив, если отзыв короче, он дополняется специальными векторами <pad>, которые несут нейтральное значение и служат исключительно для заполнения пустого пространства. Такой подход гарантирует, что каждый отзыв имеет одинаковую длину, что особенно важно при работе с алгоритмами, требующими фиксированного размера входных данных.

Чтобы более наглядно продемонстрировать процесс векторизации отзывов, на рисунке 4 представлена схема алгоритма. Данный алгоритм принимает на вход один отзыв,

состоящий из последовательности токенов, и возвращает его векторное представление фиксированной длины.

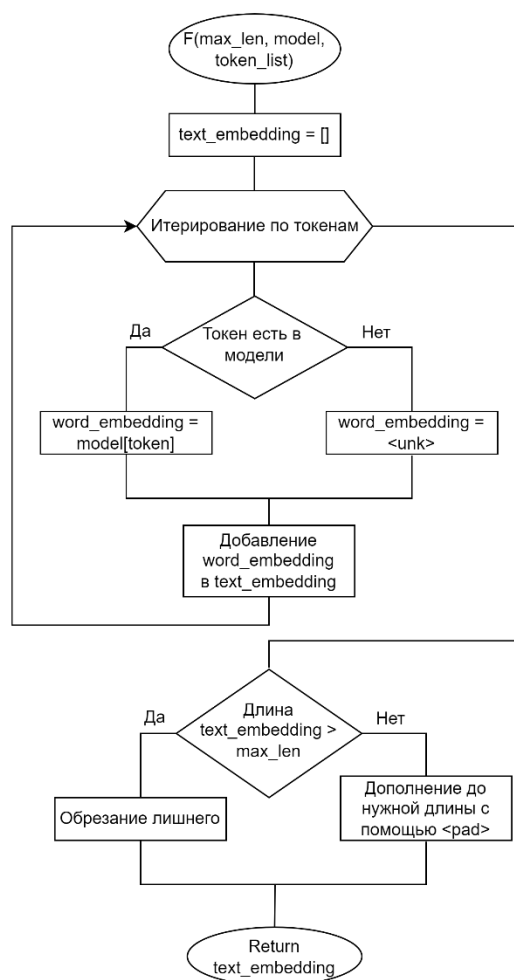


Рисунок 4 – Схема алгоритма векторизации отзывов

Метод опорных векторов (SVM) [7] был выбран для задачи классификации тональности отзывов, так как он хорошо работает с высокоразмерными данными и способен эффективно классифицировать отзывы на положительные, нейтральные и негативные. Это алгоритм, который строит гиперплоскость, разделяющую различные классы с максимальным отступом. Такой подход позволяет SVM успешно классифицировать текстовые данные, которые предварительно были векторизованы с использованием плотных векторных представлений.

В частности, для анализа тональности Word2Vec помогает выделять близость слов с эмоциональной окраской, а для определения ключевых слов можно применять косинусное сходство между векторами слов и основными темами отзывов. Такое сочетание SVM и Word2Vec обеспечивает высокую эффективность классификации текстовых данных и их глубокой семантической обработки.

С полным кодом модуля анализа отзывов можно ознакомиться в приложении Б.



## **2 Проектирование структуры и компонентов программного продукта**

### **2.1 Проектирование даталогической модели данных**

Проектирование даталогической модели данных является важным этапом в разработке системы анализа отзывов.

В системе предусмотрены несколько таблиц для хранения данных о АЗС и связанных с ними отзывов. Таблица `s_azs_address` хранит данные о координатах и адресах АЗС в формате JSON, а также связанные с ними исторические идентификаторы и дату вставки записи. Это позволяет хранить данные о географическом расположении АЗС в удобном формате и отслеживать изменения, если это необходимо.

Таблица `azs_info` хранит ключевую информацию о каждой АЗС: название, рейтинг, количество отзывов, адрес и регион. Также есть ссылка на координаты АЗС из таблицы `s_azs_address`, что позволяет легко связать эти две таблицы и получать полную информацию о местоположении АЗС.

Таблица `azs_review` содержит информацию об отзывах пользователей. Здесь хранятся такие данные, как тип отзыва, дата и время, информация о авторе, его профессиональном уровне, оценка, текст отзыва. Важно отметить, что каждая запись в таблице связана с определённой АЗС через поле `object_id`.

Данные о каждом авторе отзывов: уникальный идентификатор, имя, статус верификации, а также связанные с этим данные о загрузке файла хранятся в таблице `s_azs_rev_users`. Это позволяет отслеживать авторов и их изменения в системе.

Таблица `s_azs_categ` хранит информацию о категориях, к которым могут быть отнесены отзывы, например, о чистоте, обслуживании, качестве топлива и других аспектах работы АЗС. Таблица `azs_rev_categ` связывает АЗС с категориями и хранит статистику по числу отзывов в каждой категории (положительных и отрицательных).

В таблице `azs_review_analysis` хранятся результаты обработки отзывов, включая его очищенный текст, тональность (положительная, нейтральная, негативная) и ключевые слова. Эта таблица позволяет хранить информацию, полученную в процессе анализа текста, и использовать её для дальнейших исследований или отображения в приложении. Таблица `azs_review_analysis` связана с отзывами в таблице `azs_review` через поле `comment_text`. Это поле в таблице `azs_review_analysis` хранит текст отзыва, который после обработки может быть очищен и анализирован. Ключевым моментом здесь является использование хэша отзыва (`review_hash`), который служит уникальным идентификатором для каждого отзыва, прошедшего анализ.

ER-диаграмма базы данных представлена на рисунке 5.

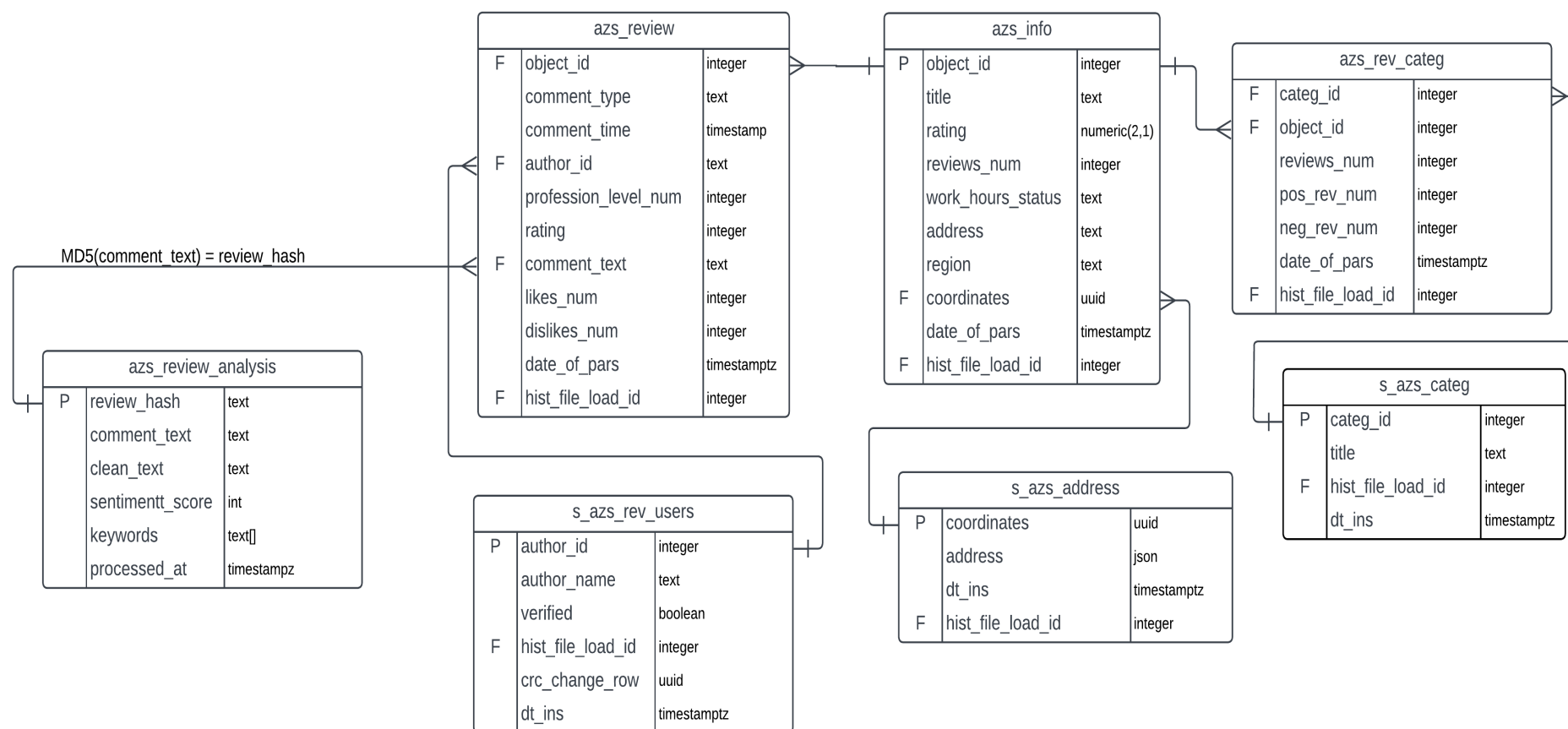


Рисунок 5 – Даталогическая модель базы данных

В таблицах предусмотрено поле `hist_file_load_id`, которое связывает данные с конкретной загрузкой данных. Это позволяет отслеживать изменения и обновления информации в базе данных, а также обеспечивать совместимость и интеграцию с другими процессами в системе.

Таким образом, проектирование даталогической модели данных для системы анализа отзывов ориентировано на создание структурированных таблиц, которые обеспечивают хранение и эффективную обработку информации об АЗС, отзывах пользователей, категориях и результатах анализа.

## **2.2 Проектирование структуры программного продукта**

Задачей проектирования при объектном подходе является разработка классов [5]. Для создания диаграммы классов предметной области на уровне спецификации, концептуальная модель должна быть разделена на два основных модуля: модуль получения данных и модуль анализа. Эти модули взаимодействуют исключительно через операции с базой данных, что соответствует изначальной концептуальной модели.

Для реализации этих модулей необходимо выделить шесть ключевых классов: Парсер (Parser), Загрузчик в базу данных (DatabaseLoader), Предобработчик (Preprocessor), Векторизатор (Vectorizer), Анализатор тональности (SentimentAnalyzer) и Извлекатель ключевых слов (KeywordExtractor). Эта диаграмма будет логическим продолжением концептуальной модели и продемонстрирует, как функциональные компоненты системы связаны друг с другом.

Рассмотрим модуль получения данных, включающий классы Парсера и Загрузчика в базу данных. Парсер извлекает и структурирует сырые данные, подготавливая их для передачи в Загрузчик, который, в свою очередь, сохраняет обработанные данные в базе. Таким образом, их взаимодействие организовано таким образом, чтобы обеспечить последовательность обработки и сохранения данных. Класс Парсер и класс Загрузчик связаны отношением ассоциации.

При реализации класса Парсера был выбран шаблонный метод, что позволило обеспечить гибкость и повторное использование кода. Базовый класс "Парсер" является абстрактным и включает общие методы и шаблонную логику извлечения данных, в то время как классы, наследуемые от базового, такие как Парсер АЗС и Парсер отзывов, реализуют специфические шаги обработки для каждой конкретной задачи. Это решение было принято, чтобы избежать дублирования кода и упростить расширение функциональности. Например, если в будущем потребуется реализовать обработку новых типов данных, достаточно будет создать новый подкласс, переопределяющий лишь

специфические этапы обработки, тогда как общая логика, заданная в базовом классе, останется неизменной.

Аналогичный подход применен при реализации Загрузчика в базу данных. В основе лежит базовый абстрактный класс, определяющий основные операции по подключению к базе, валидации данных и записи. Специализированные загрузчики, такие как Загрузчик информации об АЗС и Загрузчик отзывов, наследуются от базового и реализуют конкретные алгоритмы вставки данных, учитывающие особенности каждой таблицы или набора данных. Использование шаблонного метода в данном случае также оправдано, поскольку позволяет избежать дублирования кода для операций подключения, обработки ошибок или ведения логов. Этот подход делает систему более масштабируемой и легко поддерживаемой, обеспечивая возможность добавления новых загрузчиков с минимальными изменениями в существующем коде.

Диаграмма классов модуля получения данных изображена на рисунке 6.

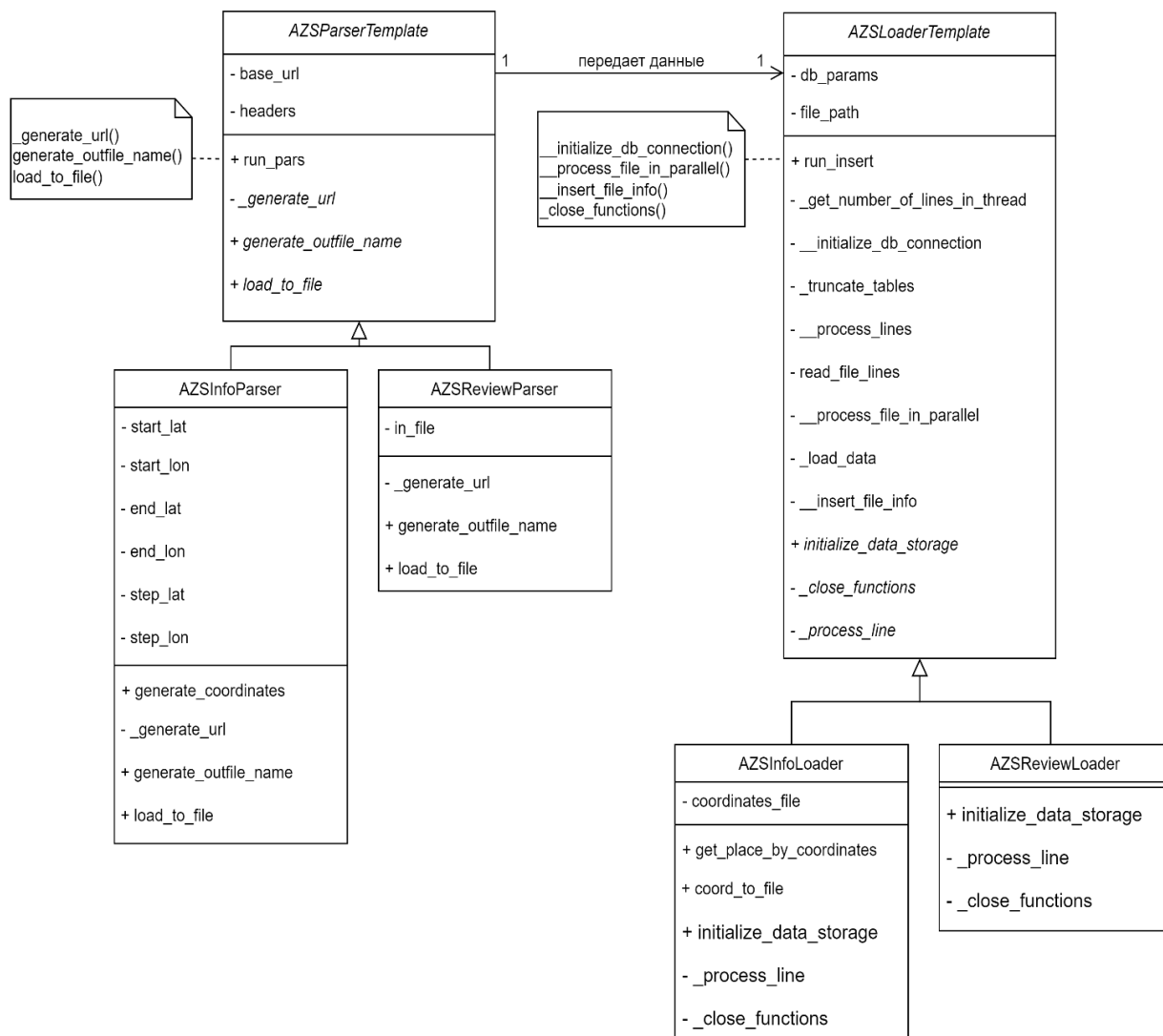


Рисунок 6 – Диаграмма классов модуля получения данных

Далее рассмотрим модуль анализа данных. Он включает классы Предобработчика, Векторизатора, Анализатора тональности и Извлекателя ключевых слов.

Класс предобработки текста отвечает за предварительную обработку текстов, включающую очистку от лишних символов, ссылок и знаков препинания; приведение текста к нижнему регистру; лемматизацию и удаление стоп-слов; коррекцию орфографии.

Векторизатор выполняет векторизацию текста с использованием модели, рассмотренной в п. 1.5, создавая векторное представление для каждого токена текста.

Класс анализа тональности использует модель машинного обучения на основе метода опорных векторов для предсказания тональности текста (положительная, нейтральная, негативная). Он получает векторизованный текст и на его основе делает предсказание.

KeywordExtractor извлекает ключевые слова из текста и расширяет их синонимами, используя модель Word2Vec для нахождения похожих слов. Ключевые слова могут быть связаны с определёнными темами, и для каждой темы вычисляются средние векторы.

Диаграмма классов модуля анализа данных изображена на рисунке 7.

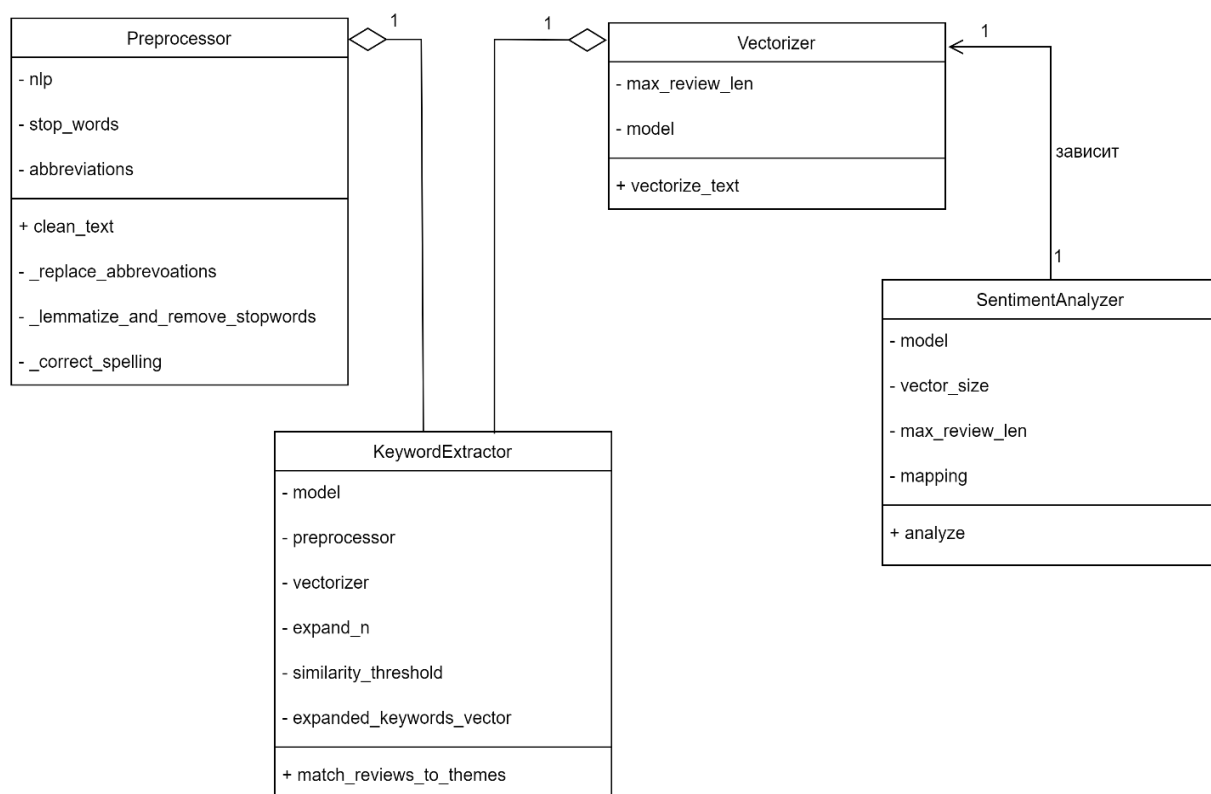


Рисунок 7 – Диаграмма классов модуля анализа данных

Предобработчик является независимым классом, который будет связан с Векторизатором и Извлекателем ключевых слов через ассоциацию, так как они используют его методы для обработки текста. KeywordExtractor зависит от существования Preprocessor

и Vectorizer для выполнения своей основной функции. Однако объекты Preprocessor и Vectorizer могут быть созданы и уничтожены независимо от KeywordExtractor, поэтому в данном случае имеет место отношение агрегации.

Анализатор тональности зависит от Векторизатора, так как ему нужны векторизованные тексты для анализа. Он хранит в себе размер вектора и максимальную длину анализируемого отзыва, использует ту же модель, что и была в векторизаторе.

Проектирование диаграмм классов является важным этапом в разработке системы, который позволяет не только структурировать информацию и функции, но и улучшить понимание взаимодействий внутри системы. Правильное использование различных типов отношений помогает достичь эффективной архитектуры, облегчая дальнейшую разработку и поддержку системы.

В приложении В предусмотрено руководство администратора, которое описывает порядок работы с модулями сбора и загрузки данных, настройку парсеров, управление базой данных.

## **2.3 Разработка интерфейса пользователя**

### **2.3.1 Проектирование пользовательского интерфейса**

В рамках разработки пользовательского интерфейса для системы был выбран процедурно-ориентированный подход [5]. Этот интерфейс предоставляет пользователю базовые функциональные возможности для взаимодействия с системой, без излишней сложности и дополнительных возможностей, которые могли бы усложнить процесс работы с приложением. Основной акцент был сделан на удобство и эффективность выполнения основных операций, таких как подключение к базе данных, просмотр и анализ отзывов, а также экспорт результатов анализа.

Основная цель заключалась в создании простого и интуитивно понятного интерфейса, который бы не перегружал пользователя лишними элементами и был удобен для работы. Было принято решение использовать темные тона, что соответствует современным тенденциям в дизайне и помогает уменьшить нагрузку на зрение при длительном использовании. Это решение также придает интерфейсу стильный и сдержанный вид.

В интерфейсе сделан акцент на функциональность, все элементы расположены логично и удобно. Использование минималистичного подхода позволяет пользователю быстро ориентироваться в системе, избегая излишней визуальной перегрузки. Визуальные акценты расставлены таким образом, чтобы основные элементы интерфейса, такие как

кнопки и поля ввода, были легко видны и доступны для взаимодействия. Это способствует улучшению восприятия и повышению удобства работы с программой.

На рисунках 8 и 9 продемонстрирован интерфейс пользователя.

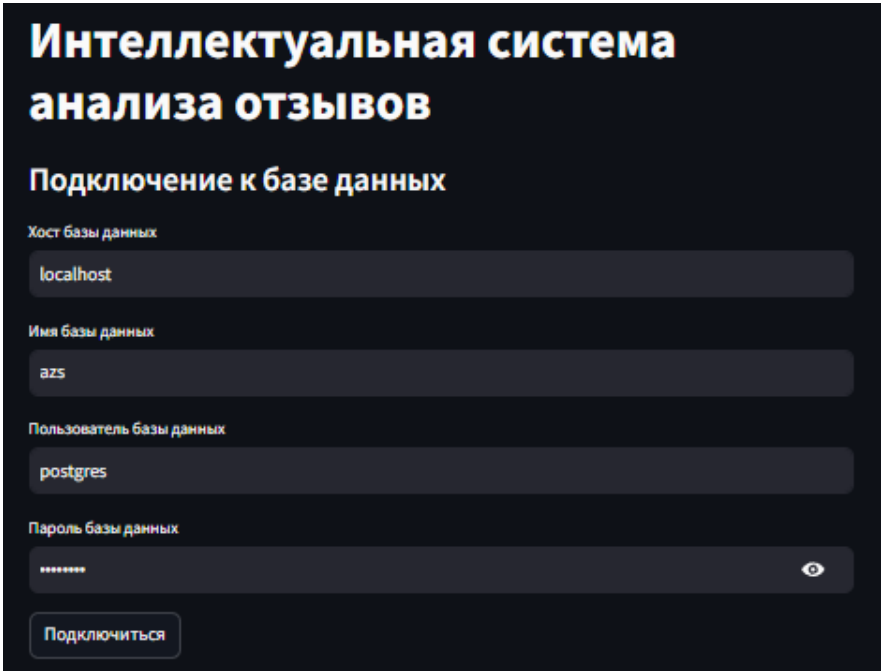


Рисунок 8 – Интерфейс пользователя

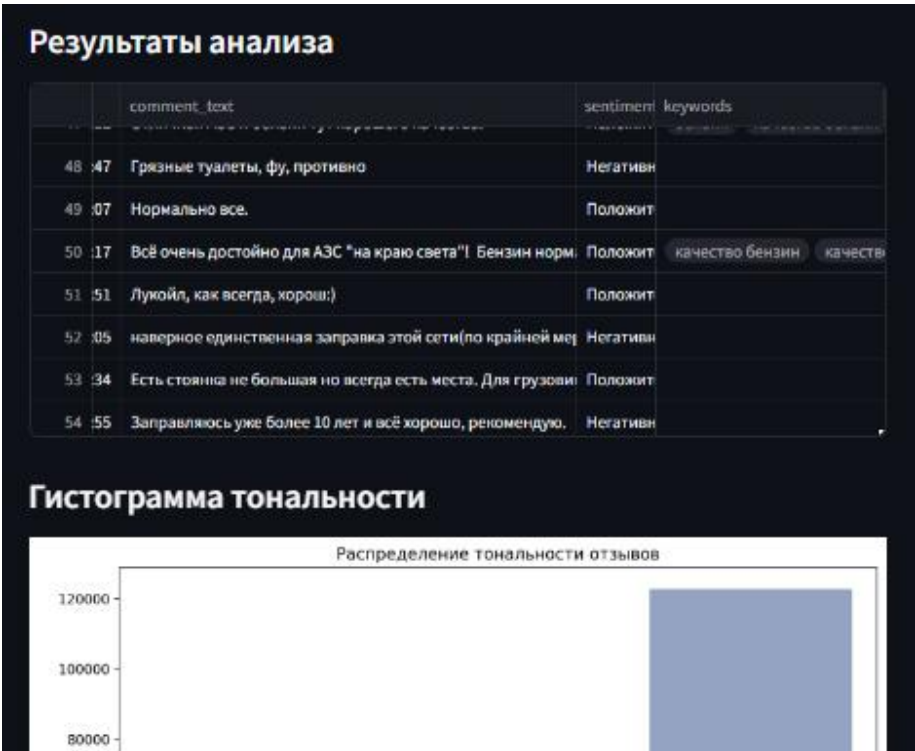


Рисунок 9 – Интерфейс пользователя

На данных скриншотах видно, как элементы управления, такие как кнопки и поля ввода, логично размещены, обеспечивая удобный и интуитивно понятный опыт для



пользователя. Цветовая палитра и простота в размещении элементов интерфейса способствуют комфортному взаимодействию с системой.

Для того чтобы упростить процесс использования и ознакомления с возможностями приложения, в приложении Г предусмотрено руководство пользователя. Оно включает подробные инструкции и шаги, которые помогут пользователям разобраться с функциональностью интерфейса.

### **2.3.2 Построение графа состояний интерфейса**

Процесс построения графа состояний интерфейса позволяет наглядно описать взаимодействие пользователя с системой и переходы между различными функциональными состояниями. В данном случае граф состоит из семи ключевых состояний, каждое из которых соответствует определенному действию пользователя или состоянию системы [8].

При открытии веб-приложения пользователь сначала попадает в состояние ожидания подключения к базе данных. При нажатии на кнопку «Подключиться» система пытается установить соединение. Если подключение успешно установлено, система переходит в состояние проверки наличия необработанных отзывов. Если же подключение установить не удалось, приложение остается в состоянии ожидания, информируя пользователя о необходимости повторить попытку.

После успешной проверки система определяет, есть ли в базе необработанные отзывы. Если таковые отсутствуют, пользователь попадает в состояние просмотра результатов анализа, где отображаются уже обработанные данные. В случае наличия необработанных отзывов пользователь переходит в состояние просмотра этих отзывов.

На этапе просмотра необработанных отзывов доступна кнопка «Анализировать необработанные отзывы». Нажав на нее, пользователь запускает процесс обработки, после чего переходит в состояние просмотра обработанных отзывов. Здесь отображаются результаты анализа, включая тональность, выделенные ключевые слова.

На этом этапе пользователь может обновить базу данных, добавив полученные данные, после чего приложение обновит результаты анализа и покажет их в расширенном виде. Полученные итоги анализа можно экспортировать в файл формата PDF, нажав соответствующую кнопку.

Граф состояний интерфейс продемонстрирован на рисунке 10.

Принятые на графе обозначения:

- С1 – нажатие кнопки «подключиться»;
- С2 – найдены необработанные отзывы;

- С3 – не найдены необработанные отзывы;
- С4 – нажатие кнопки «Анализировать необработанные отзывы»;
- С5 – нажатие кнопки «Обновить БД»;
- С6 – нажатие кнопки «Скачать PDF»;
- С7 – закрытие веб-приложения.

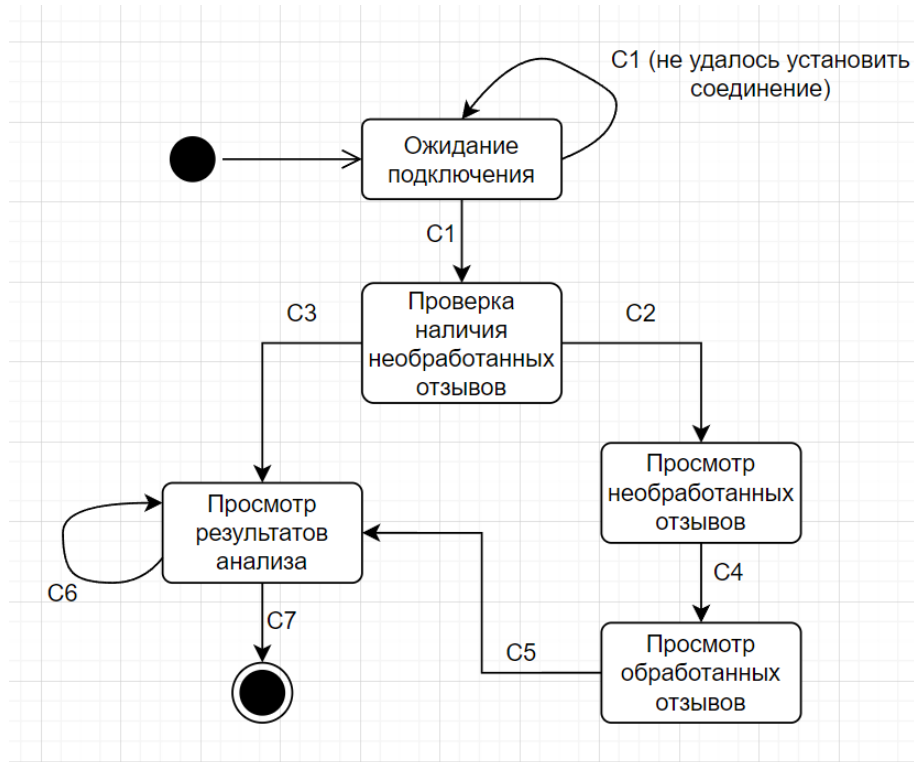


Рисунок 10 – Граф состояний интерфейса

Таким образом, построение графа состояний интерфейса отражает логику работы системы, фиксируя все основные сценарии взаимодействия пользователя с приложением.

## 2.4 Построение диаграмм последовательности

Диаграммы последовательности помогают наглядно представить, как различные компоненты системы взаимодействуют друг с другом, а также описать последовательность выполнения операций, что важно для разработки и дальнейшей оптимизации функционала приложения.

Диаграмма последовательности показывает временную последовательность сообщений, которые передаются между объектами системы, и описывает, как информация обрабатывается в ответ на действия пользователя [5]. В контексте системы интеллектуального анализа отзывов, диаграммы последовательности будут использоваться для отображения следующих вариантов использования:

- экспорт данных;
- запуск процесса сбора данных из Яндекс Карт;

– ручная разметка данных для обучения модели.

Для построения диаграмм опишем варианты использования в таблицах 4-6.

Таблица 4 – Вариант использования «Экспорт данных»

Действие пользователя	Отклик системы
1. Пользователь открывает веб-приложение. 3. Пользователь вводит параметры подключения (хост, порт, логин, пароль) и нажимает кнопку «Подключиться». 6. Пользователь запускает процесс обработки новых отзывов. 8. Пользователь нажимает кнопку «Обновить БД» для загрузки обработанных отзывов в базу данных. 10. Пользователь просматривает проанализированные данные в виде таблиц и графиков и нажимает кнопку «Экспорт PDF». 12. Пользователь получает файл с графиками.	2. Приложение отображает главное окно с формой для подключения к базе данных. 4. Приложение проверяет корректность введенных данных и пытается установить соединение с базой данных. 5. После успешного подключения выводятся все необработанные отзывы. 7. После окончания процесса обработки отзывов результаты предоставляются пользователю в табличном виде. 9. Система добавляет проанализированные отзывы в БД и отправляет запрос в базу данных для извлечения полной информации об АЗС и связанных отзывах. Полученные данные возвращаются в приложение, которое обрабатывает их и отображает пользователю в удобном формате. 11. Система формирует PDF файл из графиков и посылает его пользователю.

Диаграмма последовательности для варианта использования «Экспорт данных» представлена на рисунке 11.

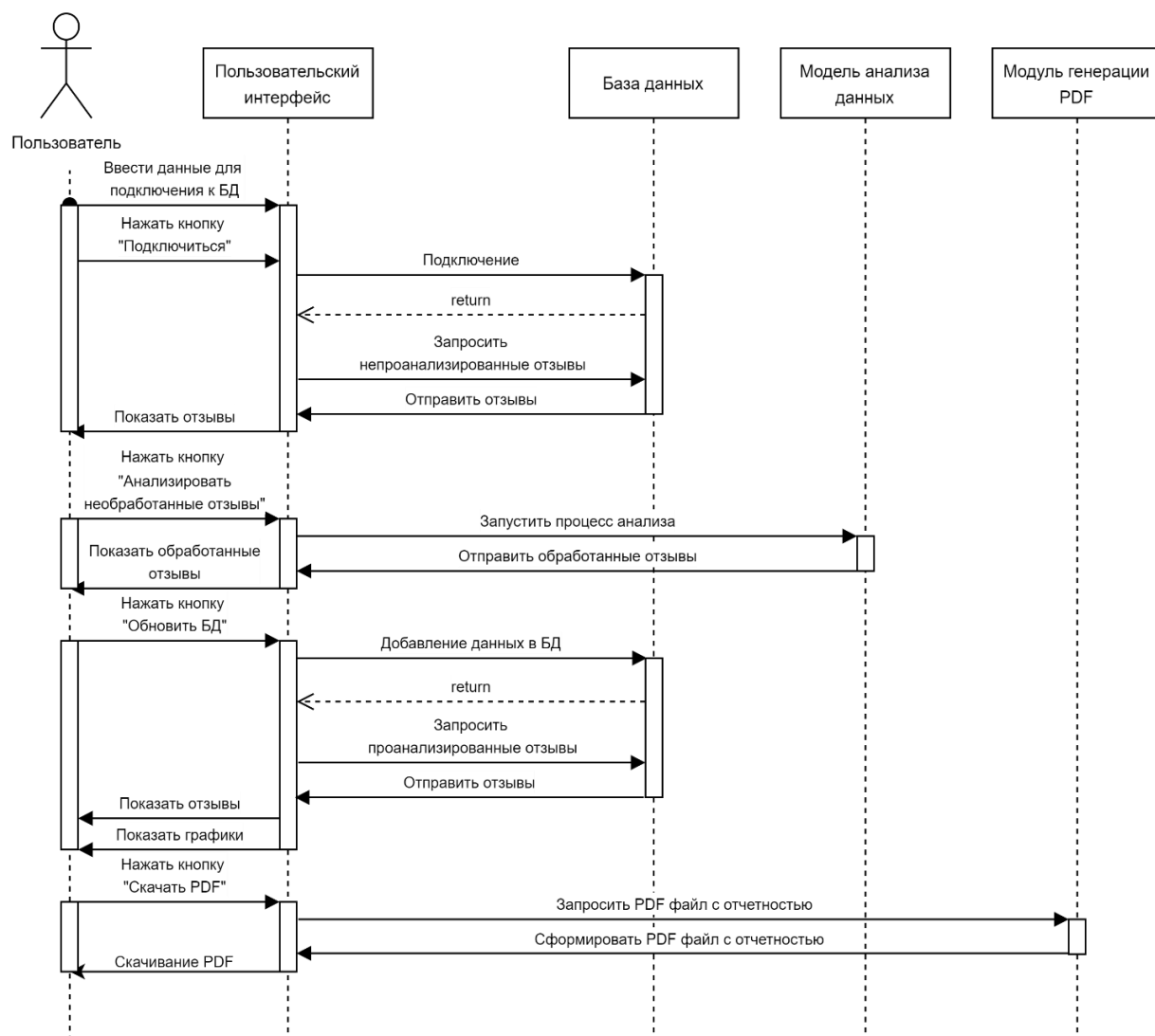


Рисунок 11 – Диаграмма последовательности для варианта использования «Экспорт данных»

Таблица 5 – Вариант использования «Запуск процесса сбора данных из Яндекс Карт»

Действие пользователя	Отклик системы
1. Администратор открывает интерфейс командной строки или файловый менеджер. 3. Администратор запускает Python-скрипт для сбора данных, введя соответствующую команду. 5. В случае успешного выполнения скрипт сохраняет собранные данные в виде файла/таблицы в указанной директории.	2. Система предоставляет доступ к директории с файлами проекта. 4. Скрипт инициирует процесс подключения к API Яндекс Карт и запускает сбор данных. 6. Система отображает уведомление о завершении процесса сбора данных.

Диаграмма последовательности для варианта использования «Запуск процесса сбора данных из Яндекс Карт» представлена на рисунке 12.

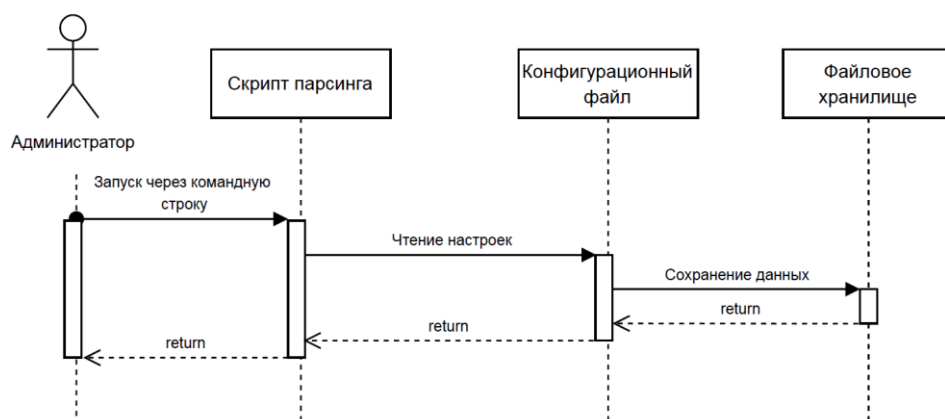


Рисунок 12 – Диаграмма последовательности для варианта использования «Запуск процесса сбора данных из Яндекс Карт»

Таблица 6 – Вариант использования «Ручная разметка данных для обучения модели»

Действие пользователя	Отклик системы
1. Администратор открывает Jupyter Notebook, предназначенный для разметки данных.	2. Jupyter Notebook отображает интерфейс разметки данных (например, отзыв и кнопки «положительный», «негативный», «нейтральный»).
3. Администратор вручную размечает данные, присваивая метки тональности или ключевые слова.	4. Система сохраняет изменения локально в файл.

Диаграмма последовательности для варианта использования «Ручная разметка данных для обучения модели» представлена на рисунке 13.

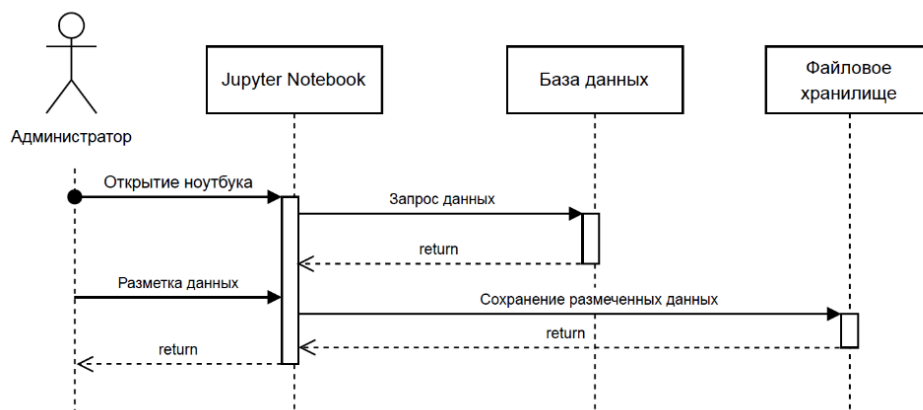


Рисунок 13 – Диаграмма последовательности для варианта использования «Ручная разметка данных для обучения модели»

Создание диаграмм последовательности для каждого из предложенных вариантов использования помогает четко определить последовательность операций, которые происходят в ответ на действия пользователя или администратора, а также наглядно показывает взаимодействие между основными компонентами системы — интерфейсом пользователя, базой данных и внешними сервисами. Эти диаграммы будут полезны как для дальнейшего совершенствования системы, так и для тестирования и отладки её функционала.

### 3 Выбор стратегии тестирования и разработка тестов

#### 3.1 Функциональное тестирование

В этом случае программа рассматривается как «черный ящик», и целью тестирования является выяснение обстоятельств, в которых поведение программы не соответствует спецификации [9].

Тестирование методом черного ящика фокусируется на внешних аспектах приложения, без предварительного знания внутренней структуры кода. Этот метод оценивает функциональность, интерфейс и общее взаимодействие с пользователем.

Метод причинно-следственных связей позволяет выявить корневые причины дефектов в функциональности приложения. Понимание того, какие действия или изменения привели к конкретному некорректному поведению, помогает лучше понимать и устранять проблемы. Этот метод оценивает функциональность, интерфейс и общее взаимодействие с пользователем.

Разработанные тесты приведены в таблице 7.

Таблица 7 – Результаты тестирования методом причинно-следственных связей

Тест	Ожидаемый результат	Полученный результат	Вывод
Проверка на пустые данные при подключении к базе данных	Вывод сообщения об ошибке	Вывод сообщения «Не удалось установить соединение с базой данных. Проверьте параметры подключения.»	Программа работает корректно
Проверка на некорректные данные при подключении к базе данных	Вывод сообщения об ошибке	Вывод сообщения «Не удалось установить соединение с базой данных. Проверьте параметры подключения.»	Программа работает корректно



Продолжение таблицы 7

Ввод корректных данных при подключении к базе данных	Получение результатов анализа / необработанных отзывов	Получение результатов анализа / необработанных отзывов	Программа работает корректно
Нажатие кнопки "Анализировать" во время выполнения анализа	Программа должна показать индикатор загрузки или сообщение о процессе	Индикатор загрузки отображается, анализ продолжается	Программа работает корректно
Проверка функциональности кнопки "Обновить БД"	Данные обновляются в базе данных, сообщение об успешном обновлении	Данные обновляются в базе данных и выводится сообщение "Успешно обновлено"	Программа работает корректно

### 3.2 Оценочное тестирование

Основная цель оценочного тестирования — проверить программное обеспечение на соответствие ключевым требованиям и убедиться, что оно готово к использованию конечными пользователями [5][9]. Оценочное тестирование особенно важно для программных продуктов, предназначенных для коммерческого применения, так как от этого этапа зависит не только их техническая состоятельность, но и восприятие пользователями.

В рамках этого раздела выделяются следующие ключевые аспекты: тестирование надежности, тестирование удобства использования и эксплуатации, тестирование на предельных объемах.

#### 3.2.1 Тестирование надежности

Надежность — это ключевой показатель качества программного обеспечения, отражающий способность системы работать без сбоев в течение определенного времени.

Тестирование надежности модуля парсинга данных об АЗС и отзывах включает несколько аспектов, связанных с функциональной надежностью.

Тестирование надежности модуля парсинга включает в себя несколько ключевых моментов. Модуль должен корректно обрабатывать возможные исключения, возникающие в процессе выполнения.

Например, при попытке разобрать некорректный JSON или при отсутствии обязательных данных в строках. В таких случаях должно быть предусмотрено логирование ошибок и продолжение обработки остальных данных, чтобы не блокировать весь процесс. В процессе работы могут встречаться строки с поврежденными данными, ошибками в формате или даже с пропущенными полями. Модуль должен быть устойчивым к таким случаям, игнорируя их или корректно обрабатывая с использованием значений по умолчанию (например, присваивая None в случае отсутствия данных).

Поскольку парсинг может включать работу с большими данными, важно проверить, что система не испытывает утечку памяти. Мы проверяем, что данные очищаются или сбрасываются из памяти после того, как они были обработаны и записаны в базу данных.

Для проверки надежности модуля необходимо также протестировать его с ошибочными входными файлами, например, с файлами, содержащими некорректные форматы JSON, пустые строки, некорректные идентификаторы и т.д. Это позволяет проверить, как модуль справляется с неожиданными входными данными.

Разработанные тесты приведены в таблице 8.

Таблица 8 – Тестирование надежности модуля парсинга

Номер теста	Назначение теста	Значения поступающих сигналов	Ожидаемый результат	Реакция программы	Вывод
1	Проверка на обработку корректного JSON	Корректный JSON с необходимыми данными	Все данные успешно собираются, строки корректно добавляются в базу	Все данные обрабатываются без ошибок	Программа работает корректно

Продолжение таблицы 8

2	Проверка, как система реагирует, если на странице нет доступных объектов	Пустые строки	Программа пропускает пустые строки и продолжает работу без остановки	Продолжение обработки следующих данных	Программа работает корректно
3	Проверка на некорректный формат JSON	Необработанный или поврежденный JSON, например, с ошибками в структуре или закрытых скобках	Программа должна вернуть ошибку, логируя проблему с конкретной строкой	Логируется ошибка с конкретным объектом данных, модуль продолжает обработку оставшихся данных	Программа работает корректно
4	Обработка ошибки HTTP 500	Статус код 500 от сервера	Программа должна выполнить повторную попытку через 1 секунду, если превышено количество попыток — завершить выполнение	Программа повторяет ежесекундные попытки до достижения максимального количества попыток, если не удастся, выводится сообщение и переход к следующему объекту	Программа работает корректно

Продолжение таблицы 8

5	Другие ошибки HTTP	Статус код отличен от 500 (например, 404 или 403)	Программа должна вывести сообщение об ошибке и завершить выполнение для координаты	Программа завершает выполнение цикла, выводя ошибку запроса, если статус-код не равен 500	Программа работает корректно
6	Другие сетевые сбои	Например, Timeout Error, SSLError	Программа должна вывести сообщение об ошибке и завершить выполнение для данной координаты	Программа завершает выполнение цикла, выводя ошибку запроса в случае сбоев соединения	Программа работает корректно
7	Тест на успешное завершение парсинга без ошибок	Успешный запрос с кодом 200 и корректными данными в ответе	Программа должна успешно продолжить парсинг и обработку данных	Программа успешно продолжает обработку данных, если статус-код равен 200	Программа работает корректно
8	Проверка обработки времени в миллисекундах	Время в формате миллисекунд (например, 168702225654)	Программа должна корректно преобразовать время в формат datetime	Время корректно преобразуется и сохраняется в нужном формате	Программа работает корректно

Все вышеперечисленные аспекты были протестированы с целью гарантировать, что при любых сбоях или ошибках процесс обработки данных будет завершен корректно, данные не будут потеряны, а приложение продолжит свою работу без серьезных сбоев.

Загрузка в базу данных — важная часть работы системы. Модуль должен корректно обрабатывать ошибки подключения и неудачные запросы. Мы проверяем работу модуля при нестабильном соединении с базой данных и наблюдаем восстановление системы после сбоев.

Разработанные тесты приведены в таблице 9.

Таблица 9 – Тестирование надежности модуля загрузки данных

Номер теста	Назначение теста	Значения поступающих сигналов	Ожидаемый результат	Реакция программы	Вывод
1	Тест на отсутствие обязательных полей	JSON с отсутствующими обязательными полями	Программа должна игнорировать такие строки и логировать ошибки	Логируется предупреждение, строка пропускается, данные в БД не вставляются	Программа работает корректно
2	Тест на корректное сохранение данных в БД	Корректный JSON с полями для записи в БД	Данные успешно сохраняются в соответствующие таблицы базы данных	Программа сохраняет данные в БД без ошибок, записывает все в соответствующие таблицы	Программа работает корректно
3	Проверка на ошибку при подключении к БД	Ошибка соединения с базой данных (например, база данных недоступна)	Программа должна логировать ошибку подключения и попытаться повторить операцию	Логируется ошибка подключения, выполнение программы блокируется до восстановления соединения	Программа работает корректно

Продолжение таблицы 9

4	Тест на работу с различными типами данных в полях	В одном поле могут быть строки, числа, булевы значения (например, в rating)	Программа должна корректно обработать данные разных типов, приводя их к нужному формату	Все данные обрабатываются и приводятся к нужному типу без ошибок	Программа работает корректно
5	Проверка обработки пустых значений в полях	Поля, такие как author_id, comment_text, могут быть пустыми или None	Программа должна обрабатывать такие значения, например, заменяя их на значения по умолчанию	Пустые значения заменяются на значения по умолчанию, данные сохраняются	Программа работает корректно
6	Тест на асинхронную обработку	Несколько файлов обрабатываются одновременно	Программа должна корректно обрабатывать несколько файлов параллельно, не создавая конфликтов в записи	Модуль обрабатывает несколько файлов одновременно и корректно сохраняет данные	Программа работает корректно

Результаты тестирования надежности показали, что программа корректно обрабатывает возможные неполадки, связанные с некорректными входными данными, такими как поврежденный JSON, отсутствующие обязательные поля и ошибки формата. Модуль парсинга успешно логирует возникающие ошибки и продолжает обработку остальных данных, не блокируя весь процесс.

### 3.2.2 Тестирование удобства использования и эксплуатации

Этот вид тестирования направлен на проверку того, насколько система удобна и понятна для администраторов и пользователей. Основное внимание уделяется интерфейсу, соответствию документации и правильности работы.

Для проведения тестирования были созданы сценарии, которые позволили оценить каждый из перечисленных аспектов. Участники тестирования включали как представителей целевой аудитории, так и специалистов, отвечающих за администрирование системы.

Результаты тестирования представлены в таблице 10.

Таблица 10 – Тестирование удобства использования и эксплуатации

№	Удобство интерфейса	Работоспособность приложения	Правильность работы	Соответствие документации
1	10	10	10	10
2	10	10	9	10
3	10	10	10	10
4	10	10	8	10
5	9	10	9	10
ср.	9,8	10	9,2	10

Таким образом, тестирование показало, что система в целом удобна и интуитивно понятна как для пользователей, так и для администраторов. Выявленные замечания не критичны и не препятствуют использованию программного продукта в рабочей среде.

### 3.2.3 Тестирование на предельных объемах

Тестирование на предельных объемах направлено на проверку работоспособности системы при обработке максимально возможного объема данных. Цель этого вида тестирования — определить, как система справляется с обработкой больших объемов информации и выявить возможные узкие места, такие как недостаток памяти, снижение производительности или сбои.

Для тестирования на предельных объемах данных были подготовлены наборы с 100 тысяч, 200 тысяч и 300 тысяч отзывов. Сценарии тестирования включали импорт данных в базу, выполнение анализа данных (определение тональности и выделение ключевых слов). В ходе тестирования замерялись время выполнения операций, использование оперативной



памяти и процессора, а также общая эффективность работы системы при разных объемах данных. Для сбора данных использовались стандартные средства мониторинга ОС и профилировщики для анализа производительности.

Результаты тестирования представлены в таблице 11.

Таблица 11 – Тестирование на предельных объемах

Объем данных	Время импорта (с)	Время анализа (с)	Использование RAM (%)	Использование CPU (%)	Ошибки обработки
100 тыс.	360	306	40	50	0
200 тыс.	700	502	60	65	0
300 тыс.	1076	712	80	75	0

Результаты тестирования показали, что система эффективно работает при объемах данных до 300 тысяч отзывов. Время импорта и обработки увеличивается с ростом объема данных, однако система продолжает функционировать без сбоев. Это подтверждает, что архитектура и алгоритмы обработки данных хорошо оптимизированы для работы с большими объемами.

Тем не менее, с увеличением объема данных может потребоваться дополнительная оптимизация для повышения производительности, например, внедрение алгоритмов пакетной обработки или использование технологий масштабирования для улучшения скорости работы системы при более высоких нагрузках.

## ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы был создан программный продукт, предназначенный для сбора, обработки и анализа отзывов о автозаправочных станциях. Система включает функционал для сбора данных, их очистки и предобработки, классификации тональности, выделения ключевых слов и тематического распределения этих слов.

В ходе разработки был проведен тщательный анализ и проектирование, направленные на создание эффективной и устойчивой системы для обработки и анализа отзывов. В процессе проектирования были учтены все требования, изложенные в техническом задании, и использованы современные подходы и методы для решения поставленных задач. Получившийся продукт успешно реализует все заявленные функции, включая сбор данных, классификацию отзывов по тональности, выделение ключевых слов и тем, а также загрузку данных в базу и предоставление удобного интерфейса для взаимодействия с пользователем.

Результаты тестирования показали, что система отвечает всем критериям надежности и стабильности работы, гарантируя корректную обработку данных и минимизацию ошибок. Полученный продукт полностью соответствует требованиям технического задания и является эффективным инструментом для анализа отзывов.

Рекомендации по использованию программного продукта включают его применение в системах мониторинга качества обслуживания на АЗС, а также в аналитических системах для оценки настроений потребителей. Благодаря возможностям настройки и масштабирования, продукт может быть использован в других сферах, где требуется обработка текстовых данных и анализ общественного мнения.

Возможные направления дальнейшего усовершенствования включают расширение функционала по улучшению точности классификации с использованием нейронных сетей, оптимизацию обработки больших данных для ускорения работы системы и добавление новых методов для более точного выделения ключевых слов с учетом контекста. Также перспективным является улучшение интерфейса пользователя для более удобного взаимодействия с системой.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Уэс Маккинни Python и анализ данных: Первичная обработка данных с применением pandas, NumPy и Jupyter / пер. с англ. А. А. Слинкина. 3-е изд. – М.: МК Пресс, 2023. – 536 с.: ил.
- 2 Документация библиотеки NLTK. "Natural Language Toolkit (NLTK)" [Электронный ресурс] – URL: <https://www.nltk.org/> (дата обращения 15.09.2024).
- 3 Документация библиотеки scikit-learn [Электронный ресурс] – URL: <https://scikit-learn.org/> (дата обращения 18.09.2024).
- 4 Документация библиотеки Word2Vec [Электронный ресурс] – URL: <https://radimrehurek.com/gensim/models/word2vec.html> (дата обращения 15.09.2024).
- 5 Иванова Г.С. Технология программирования – М.: КНОРУС, 2016. – 334 с.
- 6 Документация библиотеки Navex [Электронный ресурс] – URL: <https://github.com/nccr-itmo/NAVE> (дата обращения 15.09.2024).
- 7 Коэн М. И. Прикладная линейная алгебра для исследователей данных / пер. с англ. А. В. Логунова. – М.: ДМК Пресс, 2023. – 328 с.: ил.
- 8 Иванова Г.С., Ничушкина Т.Н., Пугачев Е.К., Самарев Р.С., Фетисов М.В., Бычков Б.И. Методические указания по выполнению курсовой работы по дисциплине «Технология разработки программных систем».: Электронное учебное издание. – М.: МГТУ им. Н.Э. Баумана, 2021. – 41 с.
- 9 Иванова Г.С., Ничушкина Т.Н., Пугачев Е.К. Выбор алгоритмов обработки данных, тестирование и повышение качества программ: учебно-методическое пособие – М.: Издательство МГТУ им. Н. Э. Баумана, 2020. – 65 с.

**ПРИЛОЖЕНИЕ А**  
**Техническое задание**  
Листов 8

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

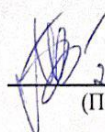
ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ  
КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ  
НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.03 ПРИКЛАДНАЯ ИНФОРМАТИКА

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА АНАЛИЗА ОТЗЫВОВ

Техническое задание на курсовую работу  
по дисциплине Технология разработки программных систем

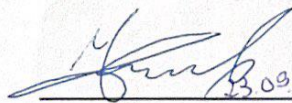
Листов 8

Студент гр. ИУ6-54Б  
(Группа)

  
23.09 2024  
(Подпись, дата)

А. И. Мокшина  
(И.О. Фамилия)

Руководитель курсовой работы,  
Старший преподаватель

  
23.09.2024  
(Подпись, дата)

М. В. Фетисов  
(И.О. Фамилия)

## 1 ВВЕДЕНИЕ

Настоящее техническое задание распространяется на разработку программной системы «Интеллектуальная система анализа отзывов» [ИСаО], используемой для автоматизированного сбора, анализа и визуализации отзывов клиентов. В данной работе система будет рассматриваться на примере анализа отзывов об автозаправочных станциях (АЗС).

Актуальность разработки обусловлена возрастающей ролью отзывов клиентов в формировании репутации и конкурентоспособности предприятий. В условиях современного рынка, когда доступ к информации становится всё более быстрым и простым, отзывы на онлайн-платформах, таких как Яндекс Карты, оказывают прямое влияние на восприятие бренда и принятие решений потенциальными клиентами. Автоматизация процессов сбора и анализа отзывов позволит оперативно реагировать на обратную связь, выявлять сильные и слабые стороны своего сервиса, и принимать обоснованные управленческие решения для повышения качества обслуживания.

Система будет использовать современные методы обработки данных, включая машинное обучение, для выявления тональности отзывов и ключевых тем, что позволит предприятиям оперативно оценивать качество своих услуг на основе мнений клиентов.

ИСаО отличается от аналогичных решений тем, что сочетает автоматический сбор отзывов, глубокую аналитическую обработку данных с применением технологий машинного обучения и интуитивно понятный интерфейс для визуализации и интерпретации результатов. В отличие от других систем, которые ограничиваются только анализом уже собранных данных, ИСаО предлагает комплексное решение, идеально подходящее для оптимизации бизнес-процессов и улучшения качества обслуживания.

## 2 ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ

Интеллектуальная система анализа отзывов разрабатывается в соответствии с тематикой кафедры ИУ6 «Компьютерные системы и сети» факультета ИУ «Информатика и системы управления» МГТУ им. Н.Э. Баумана.

## 3 НАЗНАЧЕНИЕ РАЗРАБОТКИ

Основное назначение ИСаО заключается в автоматизации процесса сбора, анализа и визуализации клиентских отзывов с Яндекс Карт. Система позволяет предприятиям быстро и эффективно оценивать качество своих услуг на основе отзывов клиентов.

## 4 ТРЕБОВАНИЯ К ПРОГРАММНОМУ ИЗДЕЛИЮ

### 4.1 Требования к функциональным характеристикам

#### 4.1.1 Выполняемые функции

##### 4.1.1.1 Для пользователя:

- подключение к базе данных через пользовательский интерфейс;
- запуск процесса выполнения автоматического анализа новых отзывов;
- просмотр таблицы с результатами анализа отзывов;
- обновление базы данных проанализированных отзывов;
- просмотр автоматически сгенерированных графиков;
- экспорт полученных графиков в формате PDF.

##### 4.1.1.2 Для администратора системы:

- запуск процесса сбора данных из Яндекс Карт;
- запуск процесса загрузки данных в базу данных;
- управление базой данных;
- изменение параметров сбора данных через конфигурационные файлы;
- ручная разметка данных для обучения модели.

##### 4.1.2 Исходные данные:

- конфигурационный файл;
- API Яндекс Карт;
- API Open Street Map;
- размеченные данные для обучения модели машинного обучения.

##### 4.1.3 Результаты:

- база данных, хранящая структурированные данные о АЗС, отзывах, пользователях, категориях; содержит таблицу, включающую результаты анализа (тональность отзывов и ключевые слова);
- графики, созданные на основе анализа данных, доступные через интерфейс Streamlit.

### 4.2 Требования к надежности

#### 4.2.1 Предусмотреть контроль вводимой информации.

#### 4.2.2 Предусмотреть блокировку некорректных действий пользователя.

4.2.3 Обеспечить целостность информации в базе данных.

4.2.4 Должна быть предусмотрена возможность быстрого и полного восстановления базы данных из резервной копии в случае сбоев или потери данных.

#### 4.3 Условия эксплуатации

4.3.1 Условия эксплуатации в соответствии с СанПиН 2.2.2/2.4.1340-03.

##### 4.3.2 Обслуживание

4.3.2.1 Регулярное резервное копирование и восстановление данных, управление базой данных и её оптимизация.

4.3.2.2 Регулярное обновление конфигурационного файла.

4.3.2.3 Диагностика и устранение неполадок, связанных с работой системы.

##### 4.3.3 Обслуживающий персонал

Системный администратор. Должен иметь навыки в управлении серверным оборудованием и решении проблем. Необходим опыт работы с базами данных, знание SQL и систем управления базами данных, умение выполнять резервное копирование и восстановление данных.

#### 4.4 Требования к составу и параметрам технических средств

4.4.1 Программное обеспечение должно функционировать на IBM-совместимых персональных компьютерах.

4.4.2 Минимальная конфигурация технических средств:

4.4.2.1 Тип процессора ..... Intel Core i5 или эквивалентный процессор AMD

4.4.2.2 Объем ОЗУ ..... 16 Гб

4.4.2.3 Объем свободного места на жестком диске ..... 30 Гб

4.4.2.4 Пропускная способность сетевого соединения ..... 10 Мбит/с

#### 4.5 Требования к информационной и программной совместимости

4.5.1 Программное обеспечение должно работать под управлением операционных систем семейств WIN64.

4.5.2 Пользовательский интерфейс должен работать в браузерах Google Chrome, Mozilla Firefox, Safari, Opera и Microsoft Edge.

4.5.3 Исходные данные включают конфигурационный файл в формате TOML, который содержит настройки системы и параметры подключения. Данные от API Яндекс Карт предоставляются в формате JSON. API Open Street Map также предоставляет данные



в формате JSON. Размеченные данные для обучения модели машинного обучения представлены в формате CSV, и содержат текстовые отзывы с метками тональности.

## 5 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1 Разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии.

5.2 Разрабатываемое программное обеспечение должно включать справочную информацию с инструкцией по использованию приложения.

5.3 В состав сопровождающей документации должны входить:

5.3.1 Расчетно-пояснительная записка на 30-40 листах формата А4 (без приложений 5.3.2, 5.3.3 и 5.3.4).

5.3.2 Техническое задание (Приложение А).

5.3.3 Фрагмент исходного кода (Приложение Б).

5.3.4 Руководство администратора системы (Приложение В).

5.3.5 Руководство пользователя (Приложение Г).

5.4 Графическая часть должна быть включена в расчетно-пояснительную записку в качестве иллюстраций:

5.4.1 Диаграмма вариантов использования.

5.4.2 Концептуальная модель предметной области.

5.4.3 Диаграммы последовательности.

5.4.4 Диаграммы классов предметной области программного обеспечения.

5.4.5 Граф состояний интерфейса.

5.4.6 Даталогическая модель базы данных

5.4.7 Схемы алгоритмов модулей (подпрограмм).

5.4.8 Таблицы тестов.

## 6 СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Этап	Содержание этапа	Сроки и объем	Представляемые результаты	
			Спецификации и программный продукт	Документы
1.	Выбор темы, составление задания, решение организационных вопросов	1..2 недели (10 %)	-	<b>Заполненный бланк задания на курсовую работу – вывешивается на сайт кафедры для получения утверждающей подписи заведующего кафедрой</b>
2.	Анализ предметной области, разработка ТЗ. Исследование методов решения, выбор основных проектных решений	3..4 недели	Определение архитектуры системы, включая компоненты сбора данных, обработки данных и визуализации. Определение структуры базы данных для хранения отзывов и результатов анализа. Определение основных вариантов использования системы. Выбор инструментов и библиотек для анализа и визуализации данных. Разработка ТЗ.	Фрагмент расчетно-пояснительной записки с обоснованием выбора средств и подходов к разработке
3.	<b>Сдача ТЗ</b>	<b>4 неделя (25 %)</b>	-	<b>Техническое задание – утверждается руководителем</b>
4.	Проектирование и реализация основных компонентов – ядра программы	5..7 недели	Технический проект основной части: алгоритмы программ, диаграммы классов предметной области программного обеспечения. Программный продукт, реализующий основные функции и содержащий модули сбора, анализа и визуализации данных (демонстрируется руководителю)	Фрагмент расчетно-пояснительной записки с обоснованием разработанных спецификаций Тексты части программного продукта, реализующего основные функции.

Этап	Содержание этапа	Сроки и объем	Представляемые результаты	
			Спецификации и программный продукт	Документы
5.	Сдача прототипа программного продукта	7 неделя (50 %)	Прототип программного продукта – демонстрируется руководителю	
6.	Разработка компонентов, обеспечивающих функциональную полноту	8..10	Рабочий проект программы. Готовая программа	Черновик расчетно-пояснительной записки. Тексты программного продукта.
7.	Сдача программного продукта	11 неделя (75 %)	Готовая программа – оценивается руководителем в баллах	-
8.	Тестирование программы и подготовка документации	12..14	Тесты и результаты тестирования.	РПЗ, Руководство пользователя и Руководство администратора.
9.	Оформление и сдача документации	14 неделя (90 %)	–	Расчетно-пояснительная записка, Руководство пользователя и Руководство администратора – проверяются и подписываются руководителем
10.	Защита курсовой работы	15..16 недели (100%)	–	Доклад (3-5 минут). Защита курсовой работы. Подписанная документация – вывешивается на сайт кафедры

## 7 ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

### 7.1 Порядок контроля

Контроль выполнения осуществляется руководителем еженедельно.

### 7.2 Порядок защиты

Защита осуществляется комиссии преподавателей кафедры.

### 7.3 Срок защиты

Срок защиты: 15-16 недели.

## 8 ПРИМЕЧАНИЕ

В процессе выполнения работы возможно уточнение отдельных требований технического задания по взаимному согласованию руководителя и исполнителя.

**ПРИЛОЖЕНИЕ Б**  
**Фрагмент исходного кода**  
Листов 5

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ

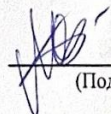
НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.03 ПРИКЛАДНАЯ ИНФОРМАТИКА

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА АНАЛИЗА ОТЗЫВОВ

Фрагмент исходного кода

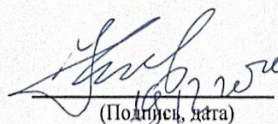
Листов 5

Студент гр. ИУ6-54Б  
(Группа)

 19.12.2024  
(Подпись, дата)

А. И. Мокшина  
(И.О. Фамилия)

Руководитель курсовой работы,  
Старший преподаватель

  
(Подпись, дата)

М. В. Фетисов  
(И.О. Фамилия)

Москва, 2024

## Листинг Б.1 – Модуль обработки и анализа отзывов

```
import pandas as pd # для работы с табличными данными
from nltk.corpus import stopwords # слова, которые игнорируются
в текстовом анализе
import re # для работы с регулярными выражениями
import spacy # для обработки естественного языка
import pyaspeller # для проверки орфографии
import time # для работы с временными метками

class Preprocessor:
    def __init__(self, stop_words=None, abbreviations=None):
        self.nlp = spacy.load("ru_core_news_sm")
        self.stop_words = stop_words if stop_words else
set(stopwords.words('russian'))
        self.abbreviations = abbreviations

    def clean_text(self, text):
        # Удаление знаков препинания чисел и ссылок
        text = re.sub(r'http\S+|www\S+|[\^\w\s]', '', text)
        # Приведение текста к нижнему регистру
        text = text.lower()
        text = self.correct_spelling(text)
        tokens = self.lemmatize_and_remove_stopwords(text)

        return tokens

    def replace_abbreviations(self, text):
        for abbr, full_form in self.abbreviations.items():
            text = text.replace(abbr, full_form)
        return text

    def lemmatize_and_remove_stopwords(self, text):
        # Применение spaCy к тексту
        doc = self.nlp(text)
        # Лемматизация и удаление стоп-слов
        tokens = [token.lemma_ for token in doc if token.lemma_
not in self.stop_words and not token.is_punct and 2 <=
len(token.text) <= 15]
        return tokens

    def correct_spelling(self, text):
        try:
            checker = pyaspeller.YandexSpeller(lang='ru')
            errors = checker.spell(text)
            for error in errors:
                text = text.replace(error['word'],
error['s'][0])
            return text
        except:
```

## Продолжение листинга Б.1

```
        print(f"Ошибка correct_spelling. Оставляю текст без
изменений.")
    return text

from gensim.models import KeyedVectors # для работы с
предварительно обученными векторными представлениями слов
import ast # преобразование строки в список
import numpy as np # для матричных вычислений

class Vectorizer:
    def __init__(self, model, max_review_len=100):
        self.max_review_len = max_review_len
        # Создание модели KeyedVectors
        vector_size = len(next(iter(model.values())))
        # Определение размерности векторов
        self.model = KeyedVectors(vector_size=vector_size)
        # Добавление векторов в KeyedVectors
        self.model.add_vectors(list(model.keys()),
list(model.values()))

    def vectorize_text(self, token_list):
        unk = self.model['<unk>']
        text_embeddings = []

        if not isinstance(token_list, list):
            token_list = ast.literal_eval(token_list)
        for token in token_list:
            # Проверяем, есть ли токен в модели, и добавляем его векторы
            if token in self.model:
                embedding = self.model[token]
            else:
                embedding = unk # Если слова нет, используем
вектор для <unk>
            text_embeddings.append(embedding)
            # Дополняем или обрезаем отзывы для фиксированной длины
            l = len(text_embeddings)
            if l > self.max_review_len:
                text_embeddings =
text_embeddings[:self.max_review_len]
            else:
                text_embeddings.extend([self.model['<pad>']] *
(self.max_review_len - l))
        return text_embeddings

class SentimentAnalyzer:

    def __init__(self, model, vector_size=300,
max_review_len=100): # x - векторизированный текст
```



## Продолжение листинга Б.1

```
self.model = model
self.vector_size = vector_size
self.max_review_len = max_review_len
self.mapping = {0: 'Негативный', 1: 'Нейтральный',
2: 'Положительный'}

def analyze(self, x):
    X = np.array(x).reshape(-1, self.vector_size *
self.max_review_len)
    y_pred = self.model.predict(X)
    return y_pred[0].item()

from sklearn.metrics.pairwise import cosine_similarity
from concurrent.futures import ThreadPoolExecutor

class KeywordExtractor:
    def __init__(self, keywords, model, preprocessor,
vectorizer, expand_n=5, similarity_threshold=0.5):
        # Создание модели KeyedVectors
        start_time = time.time()
        vector_size = len(next(iter(model.values())))) #
Определение размерности векторов
        self.model = KeyedVectors(vector_size=vector_size)
        # Добавление векторов в KeyedVectors
        self.model.add_vectors(list(model.keys()),
list(model.values()))
        self.preprocessor = preprocessor
        self.vectorizer = vectorizer
        self.expand_n = expand_n
        self.similarity_threshold = similarity_threshold
        with ThreadPoolExecutor(max_workers=10) as executor:
            self.keywords =
list(executor.map(self.preprocessor.clean_text, keywords))
            expanded_keywords = {}

        for keyword in self.keywords:
            # Объединяем подслова в строку, чтобы использовать как ключ
            combined_keyword = ' '.join(keyword)
            # Ищем ближайшие слова для каждого подслова в ключевом слове
            for subword in keyword:
                # Ищем ближайшие слова для данного ключевого слова
                try:
                    similar_words =
self.model.most_similar(subword, topn=self.expand_n)
                    # Фильтруем слова по порогу сходства
                    filtered_words = [word for word, similarity
in similar_words if similarity >= self.similarity_threshold]
```

## Продолжение листинга Б.1

```
        filtered_words.append(subword)
        # Добавляем в словарь расширенных ключевых слов
        # Используем объединённое ключевое слово как ключ
        if combined_keyword in expanded_keywords:
            expanded_keywords[combined_keyword].extend(filtered_words)
        else:
            expanded_keywords[combined_keyword] = filtered_words
    except KeyError:
        print(f"Слово '{subword}' отсутствует в модели.")

    # Удаляем дубликаты в значениях
    for key in expanded_keywords:
        expanded_keywords[key] = list(set(expanded_keywords[key]))
    self.expanded_keywords = expanded_keywords
    # Предварительная векторизация ключевых слов для каждой темы
    self.expanded_keywords_vectors = {
        theme:
        np.mean(np.array(self.vectorizer.vectorize_text(keywords)),
        axis=0)
        for theme, keywords in self.expanded_keywords.items()
    }
    # Проверка наличия векторов
    end_time = time.time()
    print(f'Время выполнения: {end_time-start_time}')

    def match_review_to_themes(self, review_vector):
        matched_themes = []
        review_vector_mean = np.mean(review_vector,
        axis=0).reshape(1, -1)
        for theme, theme_vector in self.expanded_keywords_vectors.items():
            similarity = cosine_similarity(review_vector_mean,
            theme_vector.reshape(1, -1))[0][0]
            if similarity > self.similarity_threshold:
                matched_themes.append(theme)

        return matched_themes
```

Ссылка на местоположение полного комплекта исходных модулей программы:  
[https://github.com/amokshina/course\\_work](https://github.com/amokshina/course_work)

## **ПРИЛОЖЕНИЕ В**

### **Руководство администратора системы**

Листов 4

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

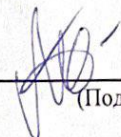
ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ  
КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ  
НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.03 ПРИКЛАДНАЯ ИНФОРМАТИКА

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА АНАЛИЗА ОТЗЫВОВ

Руководство администратора системы

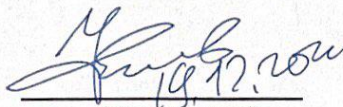
Листов 4

Студент гр. ИУ6-54Б  
(Группа)

 19.12.2024  
(Подпись, дата)

А. И. Мокшина  
(И.О. Фамилия)

Руководитель курсовой работы,  
Старший преподаватель

 19.12.2024  
(Подпись, дата)

М. В. Фетисов  
(И.О. Фамилия)

## **1 Общие сведения о программном продукте**

Программный продукт представляет собой систему для анализа отзывов с использованием данных, собранных с Яндекс Карт. Основное назначение программы — автоматизация процесса сбора, хранения, обработки и анализа текстовых данных, а также предоставление отчетов и результатов анализа.

Основные функции:

- сбор данных о точках интереса (например, АЗС) с Яндекс Карт;
- анализ отзывов с использованием предобученных моделей обработки текст;
- управление базой данных, включая добавление, обновление и удаление записей;
- обучение моделей машинного обучения с использованием размеченных данных.

Требования к техническим и программным средствам:

- операционная система: Windows/Linux;
- язык программирования: Python (рекомендуемая версия 3.9 или выше);
- объем оперативной памяти: минимум 16 ГБ;
- дисковое пространство: минимум 30 ГБ для базы данных и логов;
- программное обеспечение: PostgreSQL, Python и библиотеки указанные в requirements.txt;

## **2 Структура программы**

Система включает следующие основные компоненты:

- модуль сбора данных: взаимодействует с API Яндекс Карт для получения данных;
- модуль загрузки данных в БД: загружает данные, полученные на этапе сбора, из файла в базу данных;
- модуль обработки и анализа данных: выполняет очистку, предварительную обработку текстов и векторизацию, определяет тональность и выделяется ключевые слова в отзыве;
- модуль базы данных: обеспечивает хранение собранных данных и результатов анализа;
- модуль интерфейса: предоставляет доступ к функциям системы через веб-приложение;
- конфигурационный файл: содержит параметры для настройки системы (например, параметры подключения к API).

## **3 Настройка программы**

Для установки программного продукта выполните следующие шаги:

1) Перейдите по следующей ссылке на GitHub и клонируйте репозиторий:  
[https://github.com/amokshina/course\\_work](https://github.com/amokshina/course_work);

2) Перейдите в директорию проекта;

3) Установите все необходимые зависимости, указанные в файле requirements.txt, с помощью команды: `pip install -r requirements.txt`;

4) Убедитесь, что на вашем компьютере установлен Python версии 3.8 или выше. Если Python не установлен, его можно скачать с официального сайта [python.org](https://python.org);

5) Настройте параметры подключения к базе данных в конфигурационном файле config.toml.

Для настройки сбора данных:

1) Откройте файл config.toml;

2) Укажите URL, предоставляющий доступ к API Яндекс Карт, содержащему общую информацию об объекте в параметре url\_info;

3) Укажите URL, предоставляющий доступ к API Яндекс Карт, содержащему отзывы о заданном объекте в параметре url\_reviews;

4) Чтобы скорректировать область поиска можно задать параметры start\_lat, start\_lon, end\_lat, end\_lon, обозначающими область поиска с помощью географической широты и долготы;

5) Также можно поменять параметр шага перебора широты и долготы с помощью изменения параметров step\_lat, step\_lon.

Для корректной работы программы обновлять URL необходимо перед каждым запуском программы.

Для настройки загрузки данных в БД:

1) Откройте файл config.toml;

2) Укажите актуальные параметры доступа к базе данных user, password, database, host, port.

#### **4 Проверка программы**

Для проверки работоспособности выполните следующие шаги:

1) Запустите модуль сбора данных:

```
python full_pars.py
```

Проверьте, наличие сгенерированных файлов в папке data.

2) Запустите модуль загрузки данных в базу данных:

```
python db_loading.py
```

Проверьте, что данные корректно сохраняются в базе.

3) Запустите пользовательский интерфейс и проверьте выполнение функций в нем:

```
streamlit run app.py;
```

Убедитесь, что результаты анализа сохраняются и отображаются в интерфейсе.

4) Проверьте экспорт данных в PDF: выполните команду экспорта и откройте сгенерированный файл.

## **5 Дополнительные возможности**

Ручная разметка данных. Запустите модуль разметки `text_markup.ipynb`, используя интерфейс JupyterNotebook.

При запуске ячеек с кодом вы сможете реализовать интерактивную разметку. Вам будет показан отзыв и предложено 3 кнопки: «положительный», «отрицательный», «нейтральный». После нажатия на одну из этих кнопок, результат будет сохранен, вам будет предложен следующий отзыв для разметки.

**ПРИЛОЖЕНИЕ Г**  
**Руководство пользователя**  
Листов 6



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

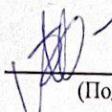
ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ  
КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ  
НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.03 ПРИКЛАДНАЯ ИНФОРМАТИКА

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА АНАЛИЗА ОТЗЫВОВ

Руководство пользователя

Листов 6

Студент гр. ИУ6-54Б  
(Группа)

 19.12.2024  
(Подпись, дата)

А. И. Мокшина  
(И.О. Фамилия)

Руководитель курсовой работы,  
Старший преподаватель

 19.12.2024  
(Подпись, дата)

М. В. Фетисов  
(И.О. Фамилия)

Москва, 2024

## **1 Общие сведения о программном продукте**

Данный программный продукт предназначен для анализа и классификации отзывов пользователей об автозаправочных станциях (АЗС). Программа использует методы обработки естественного языка для извлечения значимой информации из отзывов. Реализована возможность анализа тональности текста и выделение ключевых слов и тем, связанных с конкретным сервисом. Программное обеспечение позволяет интегрировать данные с базой данных, экспортировать результаты анализа, представленные в виде графиков в PDF.

## **2 Установка**

Для установки программного продукта выполните следующие шаги:

- 1) Перейдите по следующей ссылке на GitHub и клонируйте репозиторий:  
[https://github.com/amokshina/course\\_work](https://github.com/amokshina/course_work);
- 2) Перейдите в директорию проекта;
- 3) Установите все необходимые зависимости, указанные в файле requirements.txt, с помощью команды:

```
pip install -r requirements.txt;
```

- 4) Убедитесь, что на вашем компьютере установлен Python версии 3.8 или выше.

Если Python не установлен, его можно скачать с официального сайта [python.org](https://python.org).

## **3 Запуск программы**

Для запуска программы выполните следующие действия:

- 1) Откройте командную строку или терминал и перейдите в папку с установленной программой;
- 2) Запустите приложение с помощью команды:  
`streamlit run app.py`;
- 3) После успешного запуска откроется веб-интерфейс программы, через который вы сможете взаимодействовать с системой.

## **4 Инструкции по работе**

При запуске приложения пользователь попадает в состояние ожидания подключения к базе данных. На этом этапе система ждет действия пользователя.

Для продолжения работы необходимо ввести данные, необходимые для подключения к базе данных в соответствующие ячейки. После того как данные будут введены нажмите кнопку "Подключиться". На рисунке Г.1 продемонстрировано подключение к БД.

# Интеллектуальная система анализа отзывов

## Подключение к базе данных

Хост базы данных

Имя базы данных

Пользователь базы данных

Пароль базы данных

Рисунок Г.1 – Подключение к БД

После успешного подключения система проверяет наличие необработанных отзывов в базе данных.

Если в базе данных имеются необработанные отзывы, пользователь может просматривать их на этом экране. На рисунке Г.2 продемонстрирован этот вариант.

Имя базы данных

Пользователь базы данных

Пароль базы данных

### Необработанные отзывы

	comment_text
0	Отличное заправка!
1	Очень долгая обслуживание.

Рисунок Г.2 – Интерфейс при наличии необработанных отзывов

Для начала анализа новых отзывов нажмите кнопку "Анализировать необработанные отзывы".

Приложение начинает процесс анализа, в зависимости от количества новых отзывов, процесс может иметь разную длительность. После завершения анализа вам будет предоставлен доступ к результатам в формате таблицы, содержащей текст отзыва, определенную тональность и выделенные ключевые слова. На рисунке Г.3 показан пример данного варианта использования.

### Необработанные отзывы

	comment_text
0	Отличное заправка!
1	Очень долгая обслуживание.

Анализировать необработанные отзывы

Время выполнения: 0.6194226741790771 секунд

### Обработанные отзывы

	comment_text	clean_text	vectorized_text
0	Отличное заправка!	отличный заправка	[-0.569233238697052,-0.395483285]
1	Очень долгая обслуживание.	очень долгая обслуживание	[-0.28339293599128723,0.14809653]

Обновить БД

Рисунок Г.3 – Обработанные отзывы

На этом этапе вы можете обновить данные в базе, добавив в нее полученные результаты. Для этого необходимо нажать кнопку "Обновить БД". В случае успешного обновления вы получите соответствующее сообщение. На рисунке Г.4 представлен пример.

### Обработанные отзывы

	comment_text	clean_text	vectorized_text
0	Отличное заправка!	отличный заправка	[-0.569233238697052,-0.395483285]
1	Очень долгая обслуживание.	очень долгая обслуживание	[-0.28339293599128723,0.14809653]

Обновить БД

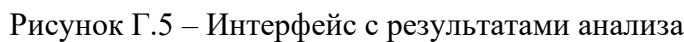
Успешно обновлено

### Результаты анализа

	object_id	azs_rating	address	region	profession_level_num	review_rating	comment_text	comn
--	-----------	------------	---------	--------	----------------------	---------------	--------------	------

Рисунок Г.4 – Успешное обновление БД

Вы видите заголовок «Результаты анализа» и таблицу, содержащую дату отзыва, его текст, определенную тональность, выделенные ключевые слова, а также адрес этой АЗС и ее рейтинг, полученный из внешних источников. Эту же таблицу вы увидите сразу после подключения к базе данных, если новые необработанные не были найдены. Пример интерфейса можно увидеть на рисунке Г.5.



### Частотное распределение тональности по категориям

Рисунок Г.6 – Сгенерированные графики



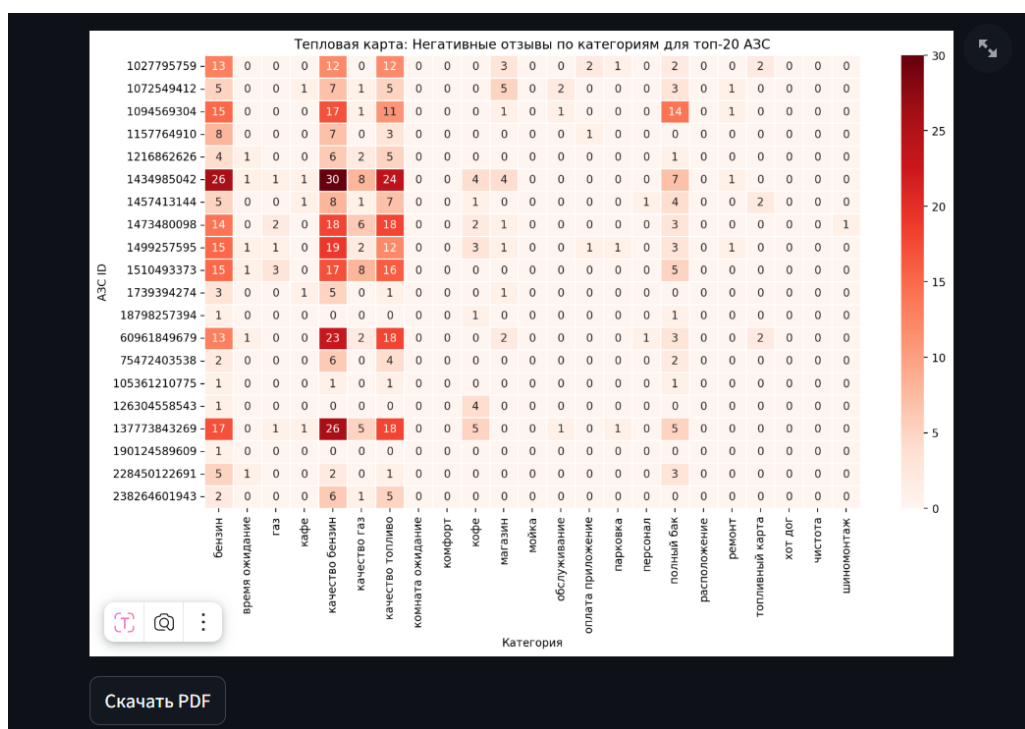


Рисунок Г.7 – Сгенерированные графики

Вы можете нажать кнопку "Скачать PDF", чтобы скачать отчет в формате PDF. В нем будут содержаться все полученные графики.

## 5 Сообщения пользователю

Программа выводит различные сообщения для информирования пользователя о статусе операций. Ниже представлен перечень возможных сообщений и действий, которые необходимо предпринять.

«Не удалось установить соединение с базой данных. Проверьте параметры подключения.» При получении этого сообщения необходимо проверить параметры подключения, убедиться в их корректности и повторить попытку.

«Время выполнения: ...» Это сообщение информирует о том, сколько времени было затрачено на анализ необработанных отзывов.

«Успешно обновлено» Сообщение появляется при успешной загрузке новых отзывов в базу данных.