МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт кибербезопасности и цифровых технологий
Кафедра КБ–3 «Разработка программных решений и системное программирование»

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

вид практики

ПРЕДДИПЛОМНАЯ ПРАКТИКА

тип практики

Тема практики: Разработка программного средства суммаризации текстов

приказ университета о направлении на практику от «18» апреля 2024 г. № 3383–С

Отчет представлен к рассмотрению:
Студент группы БСБО–17–20 «__» _____ 2024 г._____/Фролкин А.А./
(подпись и расшифровка подписи)

Отчет утвержден.
Допущен к защите:

Руководитель практики
от кафедры «__»_____2024 г._____/Филатов В.В./
(подпись и расшифровка подписи)

Руководитель практики
от профильной организации «__»_____2024 г._____/Осипов М.С./
(подпись и расшифровка подписи)

Москва 2024 г.

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт кибербезопасности и цифровых технологий
Кафедра КБ–3 «Разработка программных решений и системное программирование»

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРОИЗВОДСТВЕННУЮ ПРАКТИКУ

вид практики

ПРЕДДИПЛОМНАЯ ПРАКТИКА

тип практики

Студенту 4 курса учебной группы БСБО–17–20 Фролкину Артему Александровичу
Место и время практики: ООО “ТОРГСЕРВИС”, с 19.04.2024 по 16.05.2024 г.

Должность на практике: практикант.
1. ЦЕЛЕВАЯ УСТАНОВКА: закрепление, углубление и систематизация знаний, полученных в процессе обучения, приобретение необходимых умений, навыков и опыта практической работы по направлению подготовки.
2. СОДЕРЖАНИЕ ПРАКТИКИ
2.1. Изучить

№	Перечень теоретических вопросов, подлежащих разработке	Литература
1	Изучить готовые решения в области суммаризации текстов	отчет

2.2. Практически выполнить

№	Перечень практических заданий	Планируемый результат
1	Разработать программное обеспечение для суммаризации текстов	отчет
2	Разработать фронтенд приложение для взаимодействующее с реализованным программным обеспечением	отчет

2.3. Ознакомиться

№	Перечень вопрос для ознакомления	Планируемый результат
1	Ознакомиться с методами и программными средствами суммаризации текстов	отчет

2	Ознакомиться с принципами HTTP API	отчет
3	Ознакомиться с правилами оформления ВКР	отчет

3. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ

№	Перечень дополнительных вопросов для разработки	Планируемый результат
1	Разработать доклад по окончании прохождения практики	отчет

4. ОРГАНИЗАЦИОННО–МЕТОДИЧЕСКИЕ УКАЗАНИЯ: совместно с руководителем практики от профильной организации или кафедры разработать календарно–тематический план, составить перечень теоретических и практических вопросов для изучения, организовать рабочее место; сформировать итоговый отчет по практике.

Зав. кафедрой:
«19» апреля 2024 г. _____ (Горин Д.С.)
(Подпись)

СОГЛАСОВАНО:
Руководитель практики от кафедры
«19» апреля 2024 г. _____ (Филатов В.В.)
(Подпись)

Руководитель практики от профильной организации
«19» апреля 2024 г. _____ (Осипов М.С.)
(Подпись)

Задание получил:
«19» апреля 2024 г. _____ (Фролкин А.А.)
(Подпись)

Проведенные инструктажи:
Охрана труда: «22» апреля 2024 г.

Инструктирующий _____ (Осипов М.С., ген. дир)
(Подпись)

Инструктируемый _____ (Фролкин А.А.)
(Подпись)

Техника безопасности «22» апреля 2024 г.

Инструктирующий _____ (Осипов М.С., ген. дир)
(Подпись)

Инструктируемый _____ (Фролкин А.А.)
(Подпись)

Пожарная безопасность: «22» апреля 2024 г.

Инструктирующий _____ (Осипов М.С., ген. дир)
(Подпись)

Инструктируемый _____ (Фролкин А.А.)
(Подпись)

С правилами внутреннего распорядка ознакомлен: «22» апреля 2024 г.
_____ (Фролкин А.А.)
(Подпись)

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт кибербезопасности и цифровых технологий
Кафедра КБ–3 «Разработка программных решений и системное программирование»

СОВМЕСТНЫЙ РАБОЧИЙ ГРАФИК ПРОВЕДЕНИЯ ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ

студента Фролкина Артема Александровича 4 курса группы БСБО–17–20 очной формы обучения, обучающегося по направлению подготовки 09.03.02 «Информационные системы и технологии»

Недели практики	Сроки выполнения практики	Этап практики	Отметка о выполнении этапа практики
1	19.04.2024	Выбор темы практики. Формирование и утверждение основного и дополнительного задания на практику	
1	19.04.2024	Вводная лекция	
2	22.04.2024	Прохождение инструктажей	
2 – 4	23.04.2024 – 08.05.2024	Выполнение практических работ, консультации. Оформление и обсуждение результатов	
5	10.05.2024–16.05.2024	Представление отчетных материалов по практике и их защита. Передача обобщенных материалов на кафедру для архивного хранения. Зачетная аттестация	

Согласовано:

Зав. кафедрой _____ Горин Д.С., к.э.н., доцент
Подпись

Руководитель практики от кафедры _____ Филатов В.В.,к.т.н, доцент
Подпись

Руководитель практики от организации _____ Осипов М.С., ген. дир
Подпись

Обучающийся _____ Фролкин А.А.
Подпись

Введение

В современном информационном обществе, где объем данных растет экспоненциально, освоение и анализ текстов из открытых источников представляют собой значительную технологическую и когнитивную задачу. От моментальных новостных статей до научных исследований, текстовая информация становится не только неисчерпаемым источником знаний, но и вызовом для эффективного её усвоения и использования. В этом контексте разработка программного средства суммаризации текстов из открытых источников представляет собой существенный шаг в направлении оптимизации обработки информации и выделения ключевых аспектов из обширных текстовых массивов.

Актуальность темы данной дипломной работы определяется не только быстрым расширением объемов информации, доступной в сети, но и необходимостью в поиске эффективных инструментов для её фильтрации и конденсации. Исследование и разработка программного средства суммаризации текстов обретают важное значение в контексте информационной перегрузки, когда пользователи сталкиваются с огромными объемами текста, исследованиями, новостей и другой информации, требующей выделения наиболее значимых и интересных моментов. Такое программное средство может значительно улучшить процесс работы с текстами, сэкономив время и силы и обеспечивая эффективный доступ к необходимой информации.

Цель настоящей дипломной работы заключается в разработке и реализации программного средства суммаризации текстов из открытых источников, способного автоматически выделять и конденсировать ключевую информацию, делая её более доступной и удобной для восприятия. Данное программное средство будет основано на передовых методах обработки естественного языка, машинного обучения и алгоритмах текстового анализа, с целью предоставления пользователям эффективного инструмента для получения кратких, но содержательных обобщений текстов из различных источников.

Результаты работы программного средства будут подвергнуты тщательной оценке, что позволит определить эффективность и применимость предложенных методов суммаризации в условиях современного информационного общества.

Для достижения поставленной цели необходимо решить следующие задачи:

Анализ существующих методов суммаризации текстов. Исследование современных подходов и алгоритмов суммаризации текстов для выявления их преимуществ, ограничений и областей применения;

Выбор и адаптация методов для разработки программного средства. Определение наиболее подходящих методов с учетом конкретных требований к программному средству и его целевой аудитории;

Разработка программного средства суммаризации текстов. Создание интуитивно понятного интерфейса и интеграция выбранных методов суммаризации в функциональное программное обеспечение;

Оценка эффективности и качества программного средства. Проведение тестирования и оценка результатов с использованием соответствующих метрик для определения точности и пригодности программного средства в реальных условиях.

Этот комплекс задач предполагает создание не только инновационного программного продукта, но и углубленное исследование в области суммаризации текстов, что в конечном итоге способствует улучшению доступа к информации и повышению эффективности её использования в условиях информационного общества.

1. Готовые решения в области суммаризации текстов

1.1. Библиотеки и готовые программные решения

Суммаризация текстов стала неотъемлемой частью информационной эры, где объемы текстовой информации растут экспоненциально. Для облегчения этого информационного потока существует ряд программных средств, предназначенных для автоматической суммаризации текста. В данном обзоре рассмотрим некоторые из них, выявив их преимущества и недостатки.

1. Gensim;

Преимущества:

– Простота использования – Gensim предоставляет простой интерфейс для работы с алгоритмами суммаризации;

– Масштабируемость – Эффективно обрабатывает большие объемы текста и может масштабироваться на различные задачи.

Недостатки:

– Требуется подготовка данных – Некоторые алгоритмы требуют предварительной обработки текста.

2. Sumy;

Преимущества:

– Разнообразные методы суммаризации – Sumy поддерживает различные методы, включая LSA, TextRank, и KL–Sum;

– Настраиваемость – Пользователь может выбрать подходящий метод суммаризации в зависимости от конкретных потребностей.

Недостатки:

– Низкая степень настройки – В сравнении с некоторыми другими инструментами, у Sumy меньше опций настройки алгоритмов.

3. BERTSUM;

Преимущества:

– Использование BERT – Интегрирует мощность BERT для создания более точных и контекстно–зависимых суммаризаций;

– Учет семантики – Учитывает семантическую структуру предложений для более глубокого понимания текста.

Недостатки:

– Требует вычислительных ресурсов – Использование BERT может потребовать значительных вычислительных ресурсов.

4. NLTK (Natural Language Toolkit);

Преимущества:

– Обширная функциональность – NLTK предоставляет не только средства суммаризации, но и другие инструменты для обработки естественного языка;

– Активное сообщество – Обширное сообщество пользователей и разработчиков, что обеспечивает поддержку и обновления.

Недостатки:

– Менее современные алгоритмы – Некоторые алгоритмы могут быть менее совершенными по сравнению с более новыми подходами.

Проанализировав несколько готовых решений, можно сделать вывод, что выбор программного средства для суммаризации текстов зависит сугубо от конкретных задач и требований.

Каждый из рассмотренных инструментов обладает своими преимуществами и недостатками, и успешное использование зависит от понимания их характеристик, что в свою очередь подчеркивает важность постоянного мониторинга и выбора наилучшего решения для конкретных задач.

1.2. Автоматическая обработка естественного языка

NLP(Natural Language Processing) объединяет компьютерные науки, лингвистику и искусственный интеллект с целью создания систем, способных взаимодействовать с естественным языком так, как это делают люди. Это включает в себя обработку текста, речи, анализ сентимента, машинный перевод и многое другое.

Преимущества NLP:

1. Улучшенное понимание контекста. NLP позволяет системам понимать контекст и выражать сложные смыслы, что особенно важно при анализе больших объемов текстовой информации, например, в социальных сетях или новостных потоках;

2. Машинный перевод и глобальная связь. Системы NLP активно используются для машинного перевода, сокращая барьеры языкового взаимодействия и способствуя глобальной связности;

3. Автоматизированный анализ сентимента. NLP помогает предприятиям анализировать отзывы, комментарии и обсуждения в социальных сетях, что может быть важным инструментом для управления репутацией и улучшения продуктов и услуг.

Технологии NLP:

1. Трансформеры. Трансформеры, такие как BERT или GPT, представляют собой мощные архитектуры, обученные на огромных блоках текста, позволяют моделям понимать и генерировать текст с высоким уровнем сложности;
2. Обучение с подкреплением. Методы обучения с подкреплением в области NLP позволяют системам улучшать свои навыки через взаимодействие с окружающей средой, что схоже с процессом обучения человека.

NLP продолжает активно развиваться, и будущее обещает еще более продвинутые системы, в том числе системы суммаризации текстов.

С улучшением алгоритмов, расширением обучающих корпусов и углублением понимания человеческого языка, NLP с огромной вероятностью станет играть ключевую роль в создании более умных, адаптивных и человекоориентированных технологий и обработке, суммаризации информации.

1.3. Сравнительный анализ типов суммаризации

Проведем сравнительный анализ экстрактивной и абстрактивной суммаризации, а также рассмотрим их связь с технологиями обработки естественного языка(NLP).

Таблица 1 – Экстрактивная и абстрактивная суммаризация, основные различия

Характеристика	Экстрактивная суммаризация	Абстрактивная суммаризация
Метод извлечения информации	Прямое извлечение предложений	Творческое формирование нового текста
Структура суммаризации	Сохранение структуры исходного текста	Создание новой структуры, не связанной с оригиналом
Контекст и семантика	Ограниченное понимание семантики, ориентированное на отбор важных фрагментов	Глубокое понимание контекста, способность формировать более обширные суммаризации

NLP является обширным полем, охватывающим множество задач, включая суммаризацию текстов. В нашем случае оно является комбинирующим в себе методы обеих типов под управлением искуственного интеллекта.

Таблица 2 – Общий сравнительный анализ типов суммаризации

Критерий	Экстрактивная Суммаризация	Абстрактивная Суммаризация	NLP в Суммаризации
Гибкость в обработке текста	Ограничено выбором предложений	Гибкость в создании нового текста	Глубокое понимания и адаптация
Точность извлечения информации	Высокая	Зависит от сложности алгоритма	Глубочайшая точность через анализ контекста и семантики
Ресурсоемкость	Низкая	Высокая	Зависит от комплексности NLP–методов и объема данных

Выводы

Литературный обзор, проведенный в рамках данной работы, предоставил глубокое понимание текущего состояния области суммаризации текстов и технологий обработки естественного языка (NLP). Анализ различных методов и программных решений выявил ряд ключевых тенденций и вызовов, стоящих перед современными системами суммаризации:

1. Экстрактивная и абстрактивная суммаризация – Были рассмотрены два основных подхода к суммаризации текстов: экстрактивный, основанный на извлечении фрагментов из исходного текста, и абстрактивный, ориентированный на создание нового, смыслового содержания. Однако оба метода имеют свои ограничения, такие как потеря контекста при экстрактивной суммаризации или сложность в создании качественных абстрактов в абстрактивном подходе;

2. Использование технологий NLP – Важную роль в развитии сферы суммаризации играют технологии обработки естественного языка. Глубокие модели, такие как BERT или GPT, предоставляют новые перспективы для улучшения точности анализа текстов и повышения качества суммаризации. Эксплуатация морфологического, синтаксического и семантического анализа при помощи NLP–технологий содействует формированию более гибких и точных алгоритмов;

3. Преимущества и Ограничения существующих программных средств – Анализ существующих программных средств позволил выделить их преимущества,

такие как высокая производительность и простота использования, а также недостатки, такие как ограниченность в адаптации к различным типам текстов и трудность в работе с нестандартными форматами данных;

4. Тренды и Перспективы – Одним из явных трендов в области суммаризации текстов является углубление интеграции с технологиями NLP, использование гибридных методов, а также применение глубокого обучения для повышения качества суммаризации. Перспективы развития включают в себя улучшение алгоритмов с учетом синтаксических и семантических особенностей текстов, а также создание универсальных решений, способных адаптироваться к различным языкам и типам контента.

Дальнейшие шаги в рамках дипломной работы предполагают создание программного средства, учитывающего выявленные тренды и вызовы, с акцентом на применении гибридных методов и современных технологий для достижения более высокого качества анализа и суммаризации текстовых данных.

2. Технологический раздел

2.1. Описание структуры программного комплекса

2.1.1 Проектирование программных модулей

В разработанном веб–приложении используется модульное разделение, что позволяет происходить доработки информационной системы более гибко, а также позволяет избежать сбоев в работе основной части программы при возникновении неисправности в каком–либо модуле.

Программный комплекс будет состоять из следующих частей:

1. Модуль подготовки текстов;
2. Модуль создания заголовков;
3. Поисковый модуль;
4. Модуль суммаризации.

Схема программного комплекса представлена на рисунке 1.

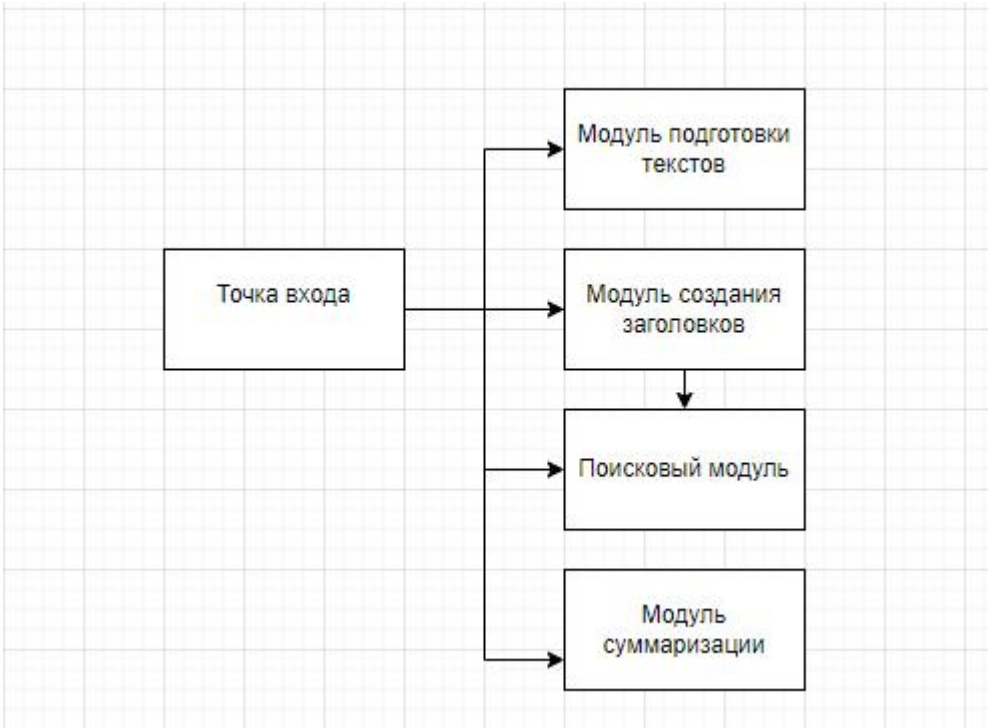

Рисунок 1 – Схема программного комплекса

Так как каждый модуль работает вне зависимости от других, на вход каждому из них данные поступают отдельно.

На вход модуля подготовки поступает текст в необработанном виде в формате. Для него выполняются ряд действий по предопработке, такие как токенизация, исключение стоп–слов и лемматизация. На выход модуль выдает

обработанный и подготовленный текст соответственно. Данный модуль разработан для корректной работы и восприятия текстов следующими модулями.

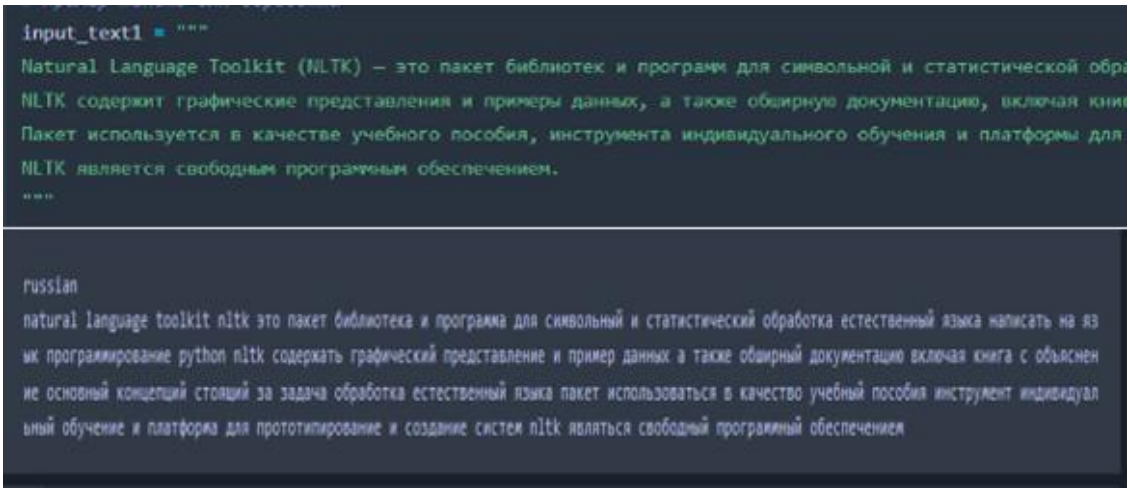

Рисунок 2 – Тексты до и после работы модуля подготовки

Работа модуля создания заголовков крайне похожа модуль подготовки, но на вход ему поступает уже обработанный текст, по которому специальный алгоритм создает удобный для дальнейшего поиска схожих текстов заголовок.

На вход поисковому модулю поступает подготовленный в предыдущем модуле запрос, по которому модуль ищет выбранное пользователем количество схожих статей со ссылками на них.

На вход модуля классификации подается подготовленный набор текстов, заранее обученная модель, а также набор слов, используемый для классификации.

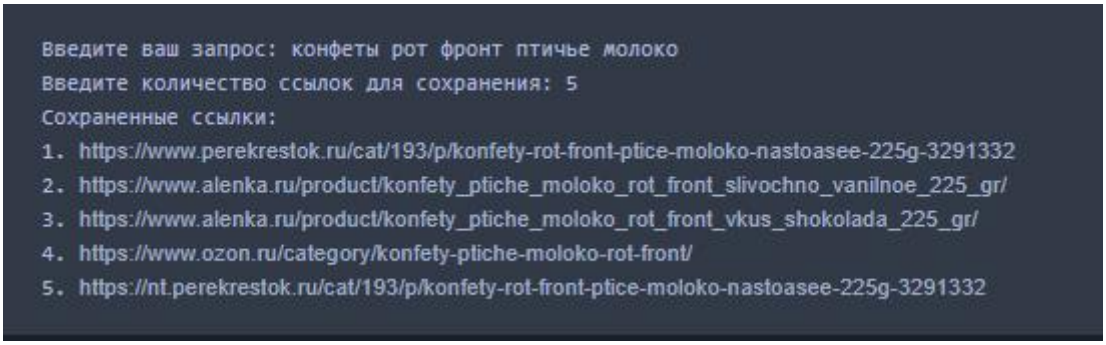

Рисунок 3 – Пример поиска страниц с текстом из запроса

Принципиальная схема работы основного модуля, модуля суммаризации текстов представлена на рисунке 4.

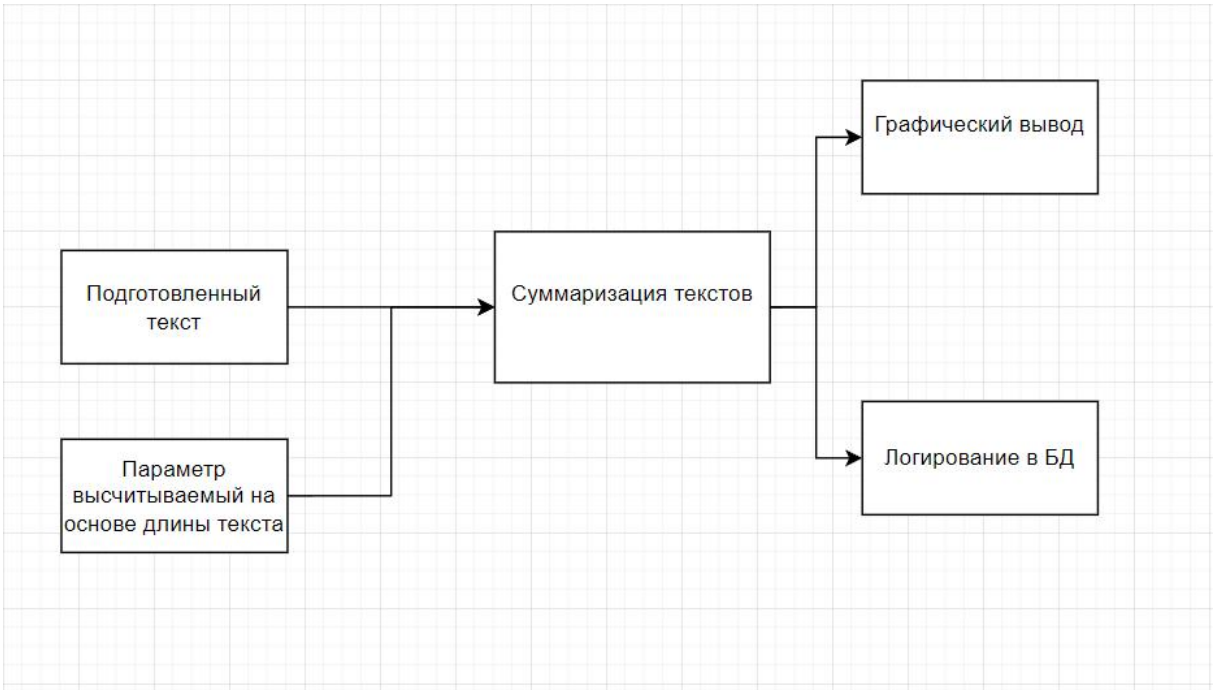

Рисунок 4 – Схема работы суммаризатора

Модуль суммаризации принимает в себя подготовленный текст и специальный параметр, высчитываемый автоматически, загружает данные в алгоритм, выполняет суммаризацию, представляет резюме в удобном для анализа формате, а также логирует операцию в БД.

2.1.2 Проектирование архитектуры клиент–серверного взаимодействия

Для данной работы была разработана структура клиент–серверного взаимодействия на основе WEB API. Схема архитектуры представлена на рисунке 5.

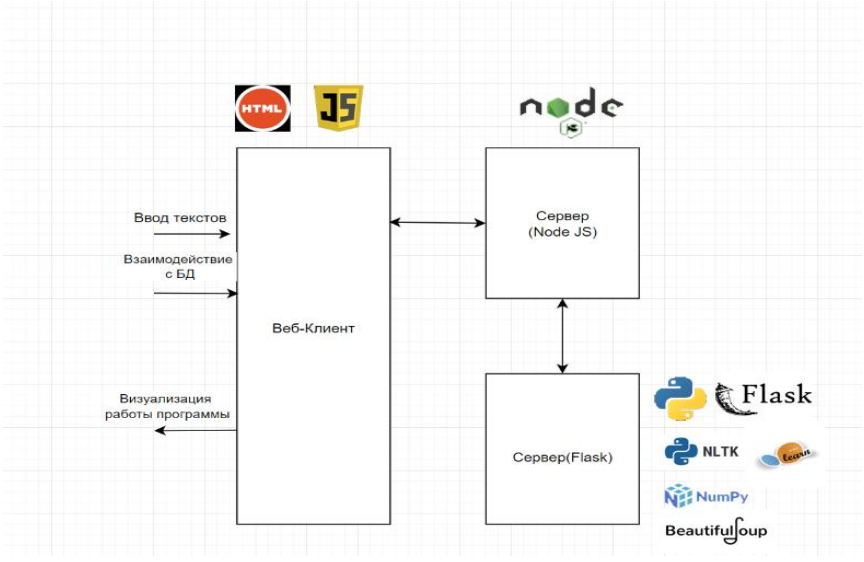

Рисунок 5 – Архитектура клиент–серверного взаимодействия

Архитектура включает в себя клиент и серверы, один для удобного взаимодействия клиента с базами данных, второй для удаленного вызова процедур. В качестве клиента выступает компьютер пользователя, который отправляет запросы на сервер и получает ответы. Обмен данными происходит с использованием запросов, данные передаются в формате JSON.

Соответственно разработаны запросы, которые отправляются при нажатии кнопок, для баз данных это запросы GET, POST, PUT, DELETE для всех таблиц. И запросы POST и GET для работы с текстом.

Все запросы работают по принципу, что клиента им будет передавать ожидаемые данные в формате json, которые попадут на сервер, на котором уже будут вызваны необходимые функции или взяты необходимые пользователю данные, после чего сервер вернет ответный запрос передав сформированный json файл.

Сервер node–js в данной архитектуре выступает неким посредником для корректной работы API и передачи данных в json, через которого пользователь–клиент пользуется процедурами хранящимися на втором сервере и базами данных.

2.2. Обоснование выбора инструментальных средств для разработки ПО

2.2.1. Операционная система Windows 10

Операционная система, используемая для разработки – Windows 10.

Windows 10 – операционная система для персональных компьютеров, выпущенная компанией Microsoft, является частью семейства операционных систем Windows NT.

Разработка программного продукта будет вестись с использованием языка программирования Python и Java Script в среде разработки Visual Studio Code 2019 x64

2.2.1. Visual Studio Code 2019 x64

Visual Studio Code (VS Code) – мощное средство разработки с открытым исходным кодом, пользующееся огромной популярностью благодаря своей гибкости и богатому экосистему расширений. Вот основные возможности VS Code для работы с разными языками обладающая целым рядом преимуществ:

1. Легковесность и быстрота – VS Code представляет собой легковесную IDE, которая быстро запускается и эффективно работает даже на машинах с ограниченными ресурсами. Это делает его идеальным выбором для разработчиков, которым важна производительность;
2. Расширяемость – Одно из основных преимуществ VS Code – это его богатая экосистема расширений. Разработчики могут настраивать и расширять функциональность с помощью тысяч бесплатных и платных расширений, позволяющих адаптировать среду разработки под свои потребности;
3. Интеграция с Git и другими инструментами разработки – VS Code обладает встроенной поддержкой систем контроля версий, включая Git, что облегчает работу с кодом в команде. Кроме того, он интегрируется с различными инструментами разработки, такими как терминал, Docker и другие;
4. Кроссплатформенность – VS Code доступен для всех основных операционных систем (Windows, macOS, Linux), что делает его удобным выбором для разработчиков, работающих на различных платформах;
5. Бесплатность – VS Code предоставляется бесплатно и с открытым исходным кодом, что делает его доступным для всех разработчиков без ограничений.

Visual Studio Code с его богатым набором расширений и интегрированными инструментами обеспечивает комфортное и продуктивное окружение для разработки требуемого программного обеспечения.

2.2.2. Язык программирования Python

Язык программирования Python является одним из самых простых в освоении и популярных языков программирования.

Язык Python обладает некоторыми примечательными особенностями, которые обуславливают его широкое распространение. Поэтому, прежде чем изучать Python, следует рассказать о его достоинствах и недостатках:

1. Python – интерпретируемый язык программирования. С одной стороны, это позволяет значительно упростить отладку программ, с другой – обуславливает сравнительно низкую скорость выполнения;
2. Динамическая типизация. В Python не надо заранее объявлять тип переменной, что очень удобно при разработке;

3. Хорошая поддержка модульности. Вы можете легко написать свой модуль и использовать его в других программах;

4. Встроенная поддержка Unicode в строках. В Python необязательно писать всё на английском языке, в программах вполне может использоваться ваш родной язык;

5. Поддержка объектно–ориентированного программирования. При этом его реализация в Python является одной из самых понятных;

6. Автоматическая сборка мусора, отсутствие утечек памяти;

7. Интеграция с C/C++, если возможностей Python недостаточно;

8. Понятный и лаконичный синтаксис, способствующий ясному отображению кода. Удобная система функций позволяет при грамотном подходе создавать код, в котором будет легко разобраться другому человеку в случае необходимости. Также вы сможете научиться читать программы и модули, написанные другими людьми;

9. Огромное количество модулей, как входящих в стандартную поставку Python, так и сторонних. В некоторых случаях для написания программы достаточно лишь найти подходящие модули и правильно их скомбинировать. Таким образом, вы можете думать о составлении программы на более высоком уровне, работая с уже готовыми элементами, выполняющими различные действия;

10. Кроссплатформенность. Программа, написанная на Python, будет функционировать совершенно одинаково вне зависимости от того, в какой операционной системе она запущена. Отличия возникают лишь в редких случаях, и их легко заранее предусмотреть благодаря наличию подробной документации;

11. Python характеризуется ясным синтаксисом. Читать код на нем легче, чем на других языках программирования, т. к. в Питоне мало используются такие вспомогательные синтаксические элементы как скобки, точки с запятыми. С другой стороны, правила языка заставляют программистов делать отступы для обозначения вложенных конструкций. Понятно, что хорошо оформленный текст с малым количеством отвлекающих элементов читать и понимать легче.

Python – это полноценный во многом универсальный язык программирования, используемый в различных сферах. Основная, но не

единственная, поддерживаемая им парадигма, – объектно–ориентированное программирование. Однако в данном курсе мы только упомянем об объектах, а будем изучать структурное программирование, так как оно является базой. Без знания основных типов данных, ветвлений, циклов, функций нет смысла изучать более сложные парадигмы, т. к. в них все это используется.

Интерпретаторы Python распространяется свободно на основании лицензии подобной GNU General Public License.

2.2.3. Язык программирования Java Script

Использование JavaScript в коде может быть обосновано по нескольким причинам:

1. Широкое применение – JavaScript является одним из самых популярных и широко используемых языков программирования. Он поддерживается практически всеми современными веб–браузерами и может быть использован для разработки веб–приложений, веб–сайтов, мобильных приложений и даже серверных приложений с использованием Node.js;
2. Динамическая природа – JavaScript – это динамически типизированный язык, который позволяет легко создавать и изменять объекты во время выполнения программы. Это делает его удобным для быстрой разработки и прототипирования приложений;
3. Асинхронное программирование – JavaScript имеет встроенную поддержку асинхронного программирования, что позволяет создавать реактивные и отзывчивые приложения. Это особенно полезно для веб–приложений, которые должны обрабатывать множество запросов от пользователей одновременно;
4. Богатый экосистем – JavaScript имеет огромное сообщество разработчиков и обширную экосистему библиотек и фреймворков, таких как React, Angular, Vue.js и другие, которые облегчают разработку сложных приложений;
5. Кроссплатформенность – JavaScript может быть запущен практически на любой платформе благодаря своей широкой поддержке и наличию интерпретаторов и сред выполнения на множестве платформ;
6. Интеграция с HTML и CSS – JavaScript тесно интегрируется с HTML и CSS, что позволяет создавать динамические и интерактивные пользовательские интерфейсы для веб–приложений.

Таким образом, использование JavaScript обосновано из–за его популярности, универсальности, гибкости и широких возможностей разработки, что делает его идеальным выбором для создания разнообразных приложений и веб–сайтов.

2.2.4. Библиотеки NLTK

NTLK является пакетом библиотек и программ для статической и символьной обработки естественного языка, написанного на языке программирования Python. Содержит в себе графические представления и примеры данных для работы. Преимуществом также является то, что данная библиотека сопровождается обширной документацией, книгами и помощью сообщества форумов, так как является на данный момент популярным инструментом. Он предоставляет простые в использовании интерфейсы для большинства лексических ресурсов, таких как WordNet, а также набор библиотек обработки текста для классификации, токкенизации, генерации, тегирования, разбора и семантических рассуждений.

Благодаря практическому руководству по внедрению основ программирования наряду с темами в вычислительной лингвистике, а также обширной документацией по API, NLTK подходит для лингвистов, инженеров, студентов, преподавателей, исследователей и пользователей в различных отраслях.

Библиотека NLTK распространяется бесплатно, имеет открытый исходный код, позволяющий использовать данный пакет программ любому интересующемуся в данной области.

2.2.5. Библиотека Scikit–Learn

Scikit–learn – это библиотека машинного обучения для языка программирования Python. Она предоставляет простой и эффективный способ решения задач классификации, регрессии, кластеризации и других типов анализа данных. Scikit–learn включает в себя широкий спектр алгоритмов машинного обучения, а также инструменты для предобработки данных, выбора моделей, оценки их производительности и многое другое.

Благодаря своей простоте в использовании и обширной документации, Scikit–learn будет легок и эффективнен в предобработке и работе с текстами.

2.2.6. Библиотека Beautiful Soup 4

Beautiful Soup 4 – это библиотека для извлечения данных из HTML и XML файлов. Она предоставляет простые методы парсинга и навигации по структуре веб–страниц, позволяя легко извлекать информацию из тегов, атрибутов и текстового содержимого.

Beautiful Soup упрощает процесс веб–скрапинга и анализа данных, делая его доступным для разработчиков Python всех уровней опыта.

2.2.7. Библиотека Numpy

NumPy – это библиотека для научных вычислений на языке программирования Python. Она предоставляет мощные структуры данных, такие как многомерные массивы (ndarrays), а также функции для работы с этими массивами. NumPy позволяет выполнять математические операции над массивами, такие как сложение, умножение, транспонирование, индексацию и срезы, а также предоставляет широкий набор функций для выполнения операций линейной алгебры, преобразований Фурье, статистики и многое другое. Благодаря своей эффективной реализации на языке C, NumPy является основным инструментом для работы с данными в Python и используется во многих научных и инженерных приложениях.

2.2.8. Библиотека pymorphy 3

pymorphy3 – это библиотека для морфологического анализа слов русского языка на языке программирования Python. Она предоставляет простой интерфейс для выполнения таких операций, как определение части речи, склонение, согласование и прочие морфологические преобразования. pymorphy3 использует морфологический словарь и грамматические правила для выполнения анализа слов, что позволяет получать точные и надежные результаты.

Эта библиотека будет использоваться в задачах обработки естественного языка (Natural Language Processing, NLP), в том числе в поисковых системах, системах автоматического перевода, анализе текстов и многих других приложениях.

2.2.5. Фреймворк Flask

Небольшой и легкий фреймворк для создания веб–приложений на языке программирования Python. Предлагает полезные инструменты и функции для обеспечения процесса создания веб–приложений с использованием Python. Flask использует механизм шаблонов Jinja для динамического создания HTML–страниц с использованием понятий в Python, таких как переменные, циклы, списки и т.д.

Будет использоваться для удобного взаимодействия со всеми функционалом реализованным на python.

2.2.6. Node.JS

Node.js – это среда выполнения JavaScript, построенная на движке V8 JavaScript от Google Chrome. Она позволяет разрабатывать серверные приложения на языке JavaScript. Одним из ключевых преимуществ Node.js является его асинхронная и событийно–ориентированная модель программирования, которая позволяет эффективно обрабатывать большое количество одновременных операций ввода–вывода.

Вот некоторые ключевые характеристики Node.js:

1. Высокая производительность – Благодаря асинхронной модели выполнения и использованию неблокирующих операций ввода–вывода Node.js может эффективно обрабатывать большое количество запросов;
2. Единая языковая платформа – Использование JavaScript как языка как для frontend, так и для backend разработки позволяет создавать полностью одноязычные приложения, что упрощает их разработку и обслуживание;
3. Богатая экосистема – Node.js имеет огромное сообщество разработчиков и обширную экосистему библиотек и фреймворков, таких как Express.js, Nest.js, Koa.js и другие, что ускоряет разработку приложений и обеспечивает широкие возможности расширения;
4. Масштабируемость – Node.js легко масштабируется горизонтально и вертикально, что позволяет обрабатывать растущую нагрузку и распределять задачи между несколькими серверами;

5. Современные технологии – Node.js активно развивается, поэтому поддерживает последние технологии и стандарты, такие как ECMAScript 6 (и более новые), HTTP/2, WebSockets и другие.

Node.js необходим для создания масштабируемых и производительных веб–приложений, API и микросервисов благодаря своей эффективности, удобству и мощным возможностям разработки.

2.3. Проектирование системы, UML диаграмма

Архитектура системы:

1. Web–сервер на Node.js – Node.js был выбран как технология для веб–сервера благодаря его высокой производительности и асинхронной природе. Веб–сервер будет отвечать за обработку HTTP запросов от клиентов и взаимодействие с базой данных PostgreSQL через API.
2. Сервер на Flask – Flask будет использован для создания серверной логики на Python. Этот компонент будет ответственен за выполнение бизнес–логики приложения, обработку запросов от веб–сервера Node.js и взаимодействие с базой данных.
3. База данных PostgreSQL – PostgreSQL был выбран в качестве реляционной базы данных из–за его надежности, мощности и расширяемости. Он будет использоваться для хранения данных приложения, таких как пользователи, товары и другие сущности.
4. Docker контейнеризация – Для облегчения управления и развертывания приложения мы будем использовать Docker. Каждый компонент системы будет упакован в отдельный контейнер, что позволит обеспечить изоляцию, масштабируемость и консистентность окружения разработки и продукции.
5. Взаимодействие через API запросы – Основным методом взаимодействия между компонентами системы будет использование API запросов. Веб–сервер Node.js и сервер Flask будут предоставлять HTTP API для выполнения операций с данными и получения результатов. Это позволит клиентскому приложению взаимодействовать с серверной частью независимо от технологий, используемых на клиентской стороне.

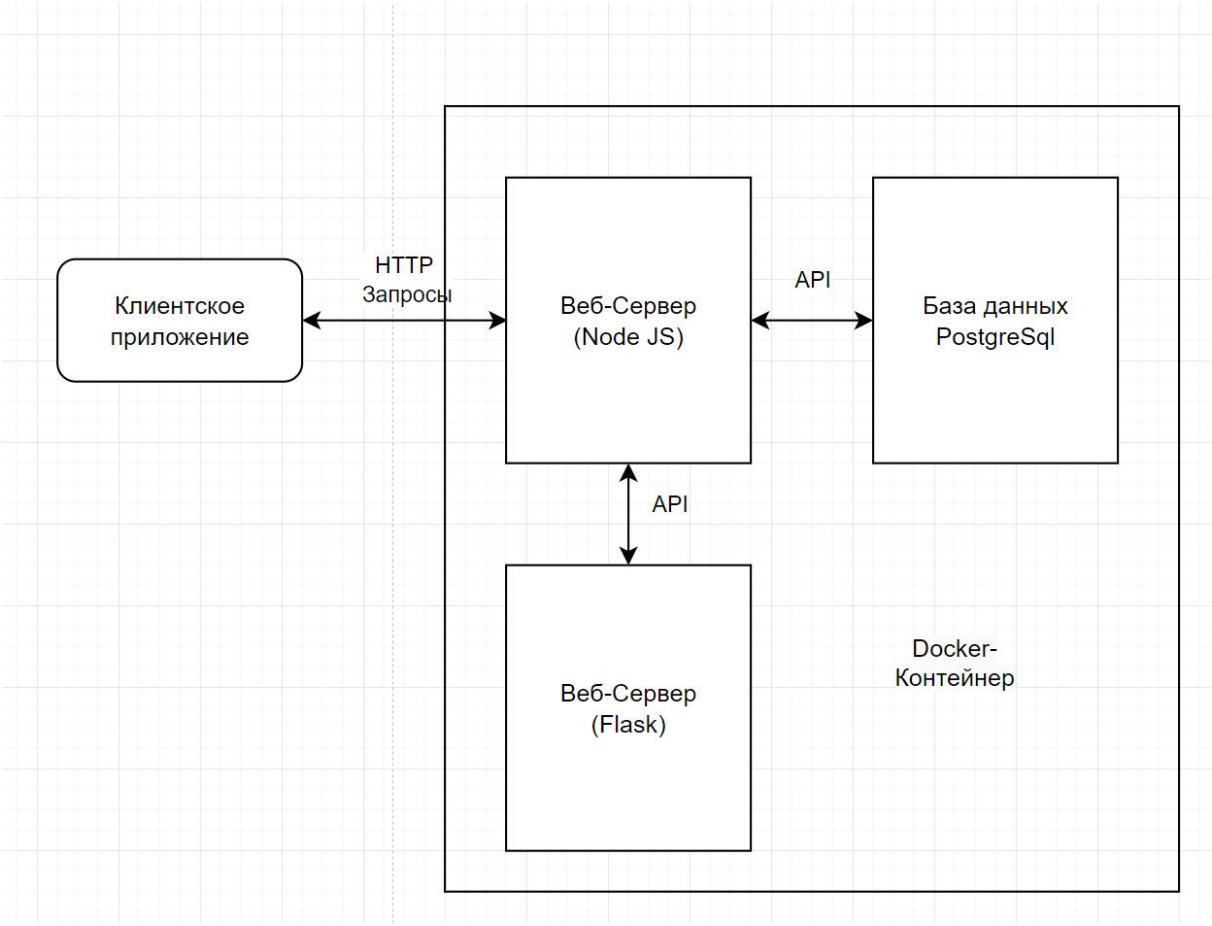

Рисунок 6 – Диграмма компонентов системы

2.4. Описание основных функций и алгоритмов программного обеспечения

2.4.1. Описание алгоритма суммаризации

Алгоритм суммаризации основан на стандартном конвейере обработки текста.

Схема алгоритма представлена на рисунке 7.

Рисунок 7 – Алгоритм суммаризации

Для реализации данного алгоритма применялась следующая последовательность действий:

1. Предподготовка текстов(токенизация, удаление стоп–слов и лемматизация) с помощтю nltk;

2. Вычисление числа встречаемости ключевых слов и их нормализация с помощью nltk;

3. Оценка важности слов, важности предложений и формирование результата.

Далее будут подробно описаны все алгоритмы, используемые для реализации вышеуказанных действий

2.4.2. Алгоритм предподготовки текстов

Алгоритм переподготовки текстов происходит в три этапа:

1. Получение данных от клиента к серверу через HTTP API – на данном этапе происходит получение файла в формате json, хранящего в себе текстовую информацию

2. Преподготовка данных – данные файла обрабатываются и представляются в виде, пригодном для дальнейшей работы.

3. Методы предобработки

На рисунке 8 представлена блок–схема модуля переподготовки текстов.

Рисунок 8 – Схема алгоритма модуля предподготовки текстов

Получение текста представляет собой передача текста с HTML страницы клиента методом POST запроса к API сервера.Текст передается через json файл, передает он данные, которые необходимо обработать

– Токенизация выполнена стандартными средствами библиотеки NLTK.

– Исключение стоп–слов выполняется путем прохождения по всем элементам массива слов и сравнения их с набором стоп–слов библиотеки NLTK.

– Лемматизация выполнена проходом цикла по массиву слов и применение функции normal_forms из библиотеки pymorphy2 к каждому слову.

2.4.3. Алгоритм вычисления ключевых слов и их нормализация

Данный алгоритм модели вынесен в отдельный шаг, так как это довольно ресурсоемкая задача и эффективнее вызывать ее с уже настроенными параметрами. Блок–схема алгоритма подготовки модели представлена на рисунке 9.

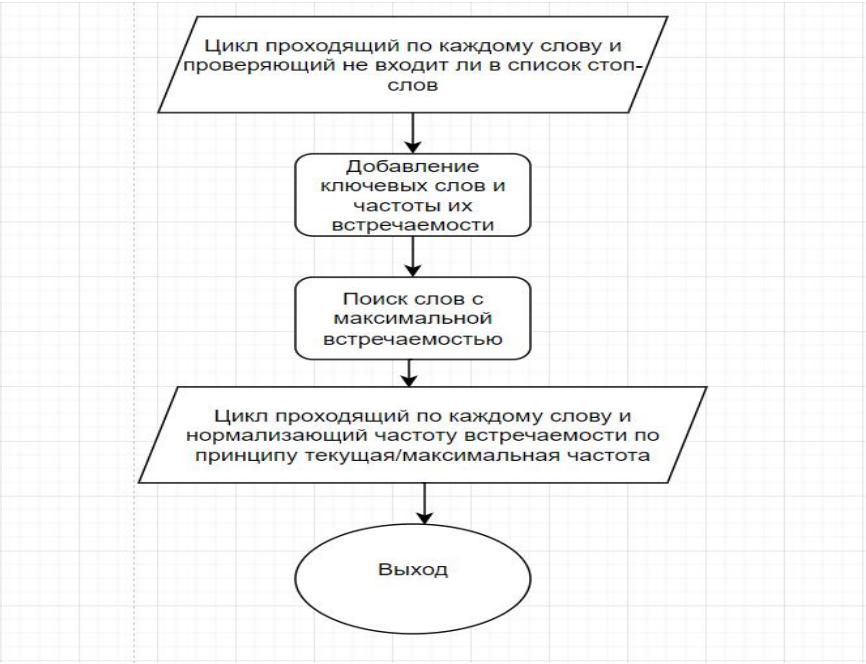

Рисунок 9 – Схема алгоритма подготовки модели

В данном алгоритме используется текстовые данные, этот текст запускается в цикл, который проходит по всем словам, и после этого заполняет массив слов. После того, как массив слов подготовлен, подсчитывается частота их встречаемости.

Следующий шаг – поиск наиболее важных, часто встречаемых слов, нормализация частоты встречаемости и пересборка массива, после чего мы получаем нормализованный список наиболее важных слов в тексте.

2.4.4. Алгоритм суммаризации

После подготовки входных текстов, и вычеслении ключевых слов, следует шаг самой суммаризации. В основе классификации лежит сравнение числовых значений слов и предложений. Блок схема алгоритма классификации представлена на рисунке 10.

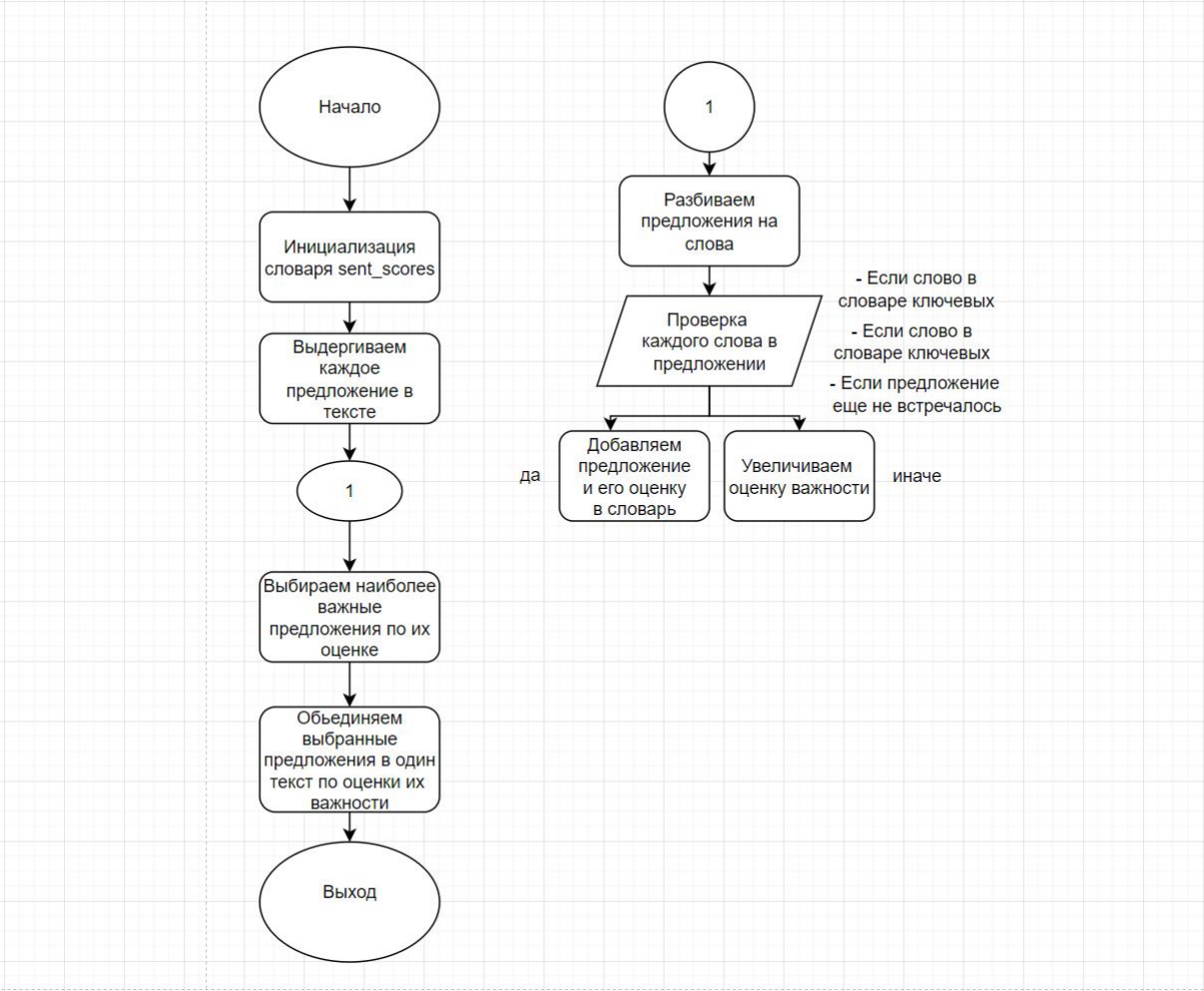

Рисунок 10 – Схема алгоритма классификации

Инициализация словаря важности предложений:

На первом этапе алгоритма создаётся пустой словарь sent_scores, предназначенный для хранения оценок важности каждого предложения в тексте.

1. Анализ каждого предложения – Алгоритм проходит по каждому предложению в тексте и анализирует его содержимое. Для этого каждое предложение разбивается на слова с последующим приведением их к нижнему регистру;
2. Оценка важности предложения – Для каждого слова в предложении проверяется его наличие в словаре частоты встречаемости слов word_freq. Если слово встречается в данном словаре, а также количество слов в предложении не превышает 30, то оценка важности данного предложения увеличивается на частоту встречаемости этого слова;
3. Выбор наиболее важных предложений – После анализа всех предложений выбираются наиболее важные из них. Для этого используется функция nlargest, которая сортирует предложения по их оценкам важности и выбирает заданное количество предложений, переданное в переменной num_sentences;

4. Составление краткого содержания – Выбранные предложения объединяются в единый текст, который представляет собой краткое содержание текста;

5. Возвращение результатов – Полученный текст краткого содержания возвращается в качестве результата работы алгоритма.

Этот метод позволяет автоматически извлекать ключевую информацию из текстовых документов, что в свою очередь и являлось ключевым требованием к ПО.

2.5. Инфологическая/даталогическая модель БД

В рамках проекта для логирования и организации системы безопасности была реализована база данных summarization_db. В качестве СУБД был выбран PostrgreSql. Далее представлена схема базы данных.

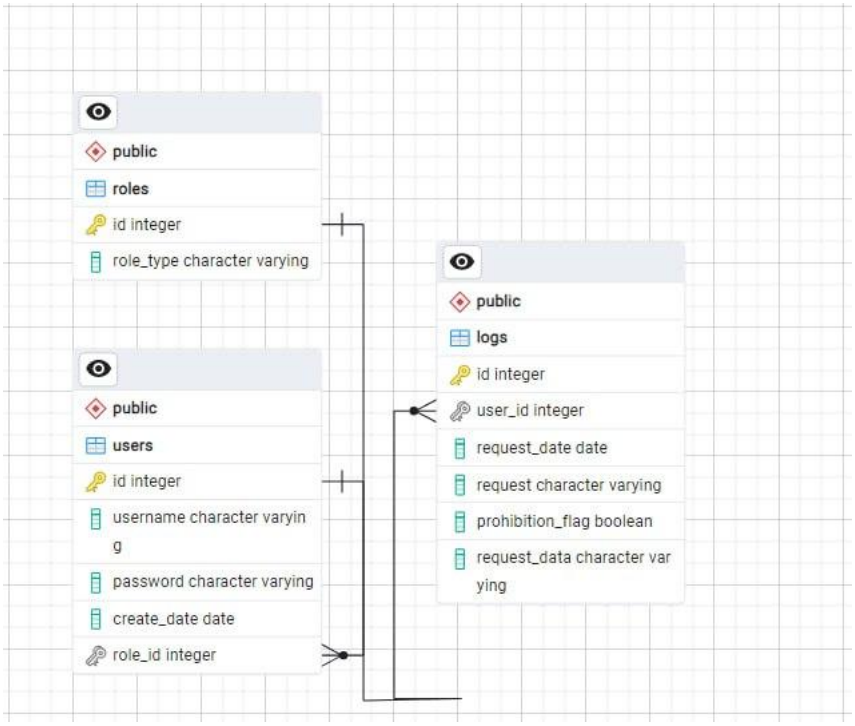

Рисунок 11 – Диаграмма базы данных

Внутри базы данных были подняты три таблицы.

1. Roles – для определения прав доступа;
2. Users – данные о пользователе;
3. Logs – логи запросов пользователей с флагом подозрительных запросов на суммаризацию текста.

В рамках базы данных функционал ПО будет распределен между пользователями согласно их ролям, что позволит обеспечить должный контроль за безопасностью.

2.6. Тестирование программного обеспечения

Во время тестирования ПО проводилась серия экспериментов, в результате чего было сделано заключение, что разработанное ПО может применяться для решения задачи суммаризации текстов. На рисунке 12 изображен пример работы программы.

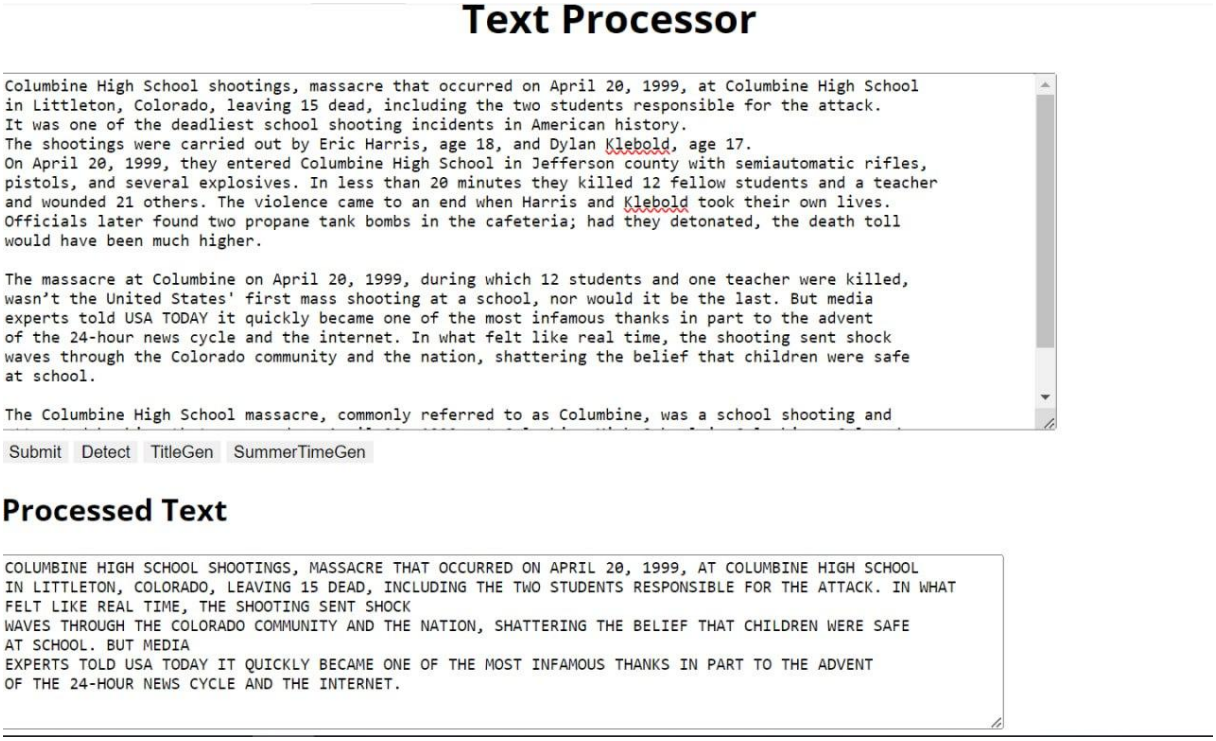

Рисунок 12 – Пример работы программы

Выводы

В данном разделе были описаны используемые технологии для реализации программного обеспечения. Были разработаны структуры, интерфейс и основные функции программного обеспечения. Также было проведено тестирование разработанной программы, которое показало работоспособность данного программного обеспечения и соответствие поставленным задачам в рамках выпускной квалификационной работы.

Заключение

В ходе преддипломной практики был проведен обширный анализ и выполнены практические задания, направленные на разработку программного обеспечения для суммаризации текстов. В рамках теоретической части были изучены существующие решения в данной области, что позволило получить полное представление о современных методах и подходах к автоматическому созданию кратких обзоров текстов.

Практическая часть включала разработку двух ключевых компонентов: программного обеспечения для суммаризации текстов и front–end приложения для взаимодействия с ним. Оба компонента были успешно реализованы и протестированы, что позволило создать работоспособную систему, способную эффективно выполнять задачи по суммаризации текстов.

Кроме того, было уделено внимание изучению дополнительных аспектов, таких как методы и программные средства для суммаризации текстов, принципы работы с HTTP API, а также правила оформления выпускных квалификационных работ (ВКР). Это помогло не только в успешной реализации проекта, но и в подготовке к дальнейшему написанию и оформлению ВКР.

На заключительном этапе практики был разработан доклад, который подробно описывает все этапы выполненной работы, полученные результаты и достигнутые цели. Доклад служит итоговым документом, отражающим все аспекты прохождения практики и готовность к защите выпускной квалификационной работы.

Таким образом, преддипломная практика была выполнена в полном объеме и достигла всех поставленных целей, что подтверждается разработанным программным обеспечением и подготовленными отчетами по каждому этапу работы.

Список используемых источников

1. Дуглас Крокфорд, Как устроен JavaScript – 2019 – С. 73–80;
2. Марейн Хавербек, Выразительный JavaScript – 2020 – №. 2 – С. 202–224;
3. Адитья Бхаргава, Грокаем Алгоритмы – 2017 – С. 47–66;
4. Суммаризация текста: подходы, алгоритмы, рекомендации и перспективы [Электронный ресурс] / 2020 / Текст электронный – URL, https://habr.com/ru/articles/514540/ (дата обращения 25.03.2024);
5. Summarizing Lengthy Articles [Электронный ресурс] / 2022 / Текст электронный – URL, https://medium.com/globant/summarizing–lengthy–articles–cdbe4b27ee6/ (дата обращения 29.03.2024);
6. Natural Language Toolkit [Электронный ресурс] / 2024 / Текст электронный – URL, https://www.nltk.org/ (дата обращения 03.04.2024);
7. Анализ текстовых данных с помощью NLTK и Python // Блог компании OTUS [Электронный ресурс] / 2023 / Текст электронный – URL, https://habr.com/ru/companies/otus/articles/774498/ (дата обращения 03.04.2024);
8. Васильев А.Н. Программирование на Python в примерах и задачах – 2021 – С. 211–248;
9. Билл Любанович, Простой Python. Современный стиль программирования – №. 2 – 2021 – С. 242–302.