НПОУ «ЯКУТСКИЙ КОЛЛЕДЖ ИННОВАЦИОННЫХ ТЕХНОЛОГИЙ» (НПОУ «ЯКИТ»)

Отделение информационных технологий и туризма

Специальность 09.02.07 «Информационные системы и программирования»

ПРОЕКТНАЯ РАБОТА

по дисциплине: Языки программирования

по теме:

Создание десктопного приложения для семантического анализа русского языка с использованием библиотеки Hugging Face Transformers на Python

Исполнитель студент гр. КИСП-23(1)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Л.П. Лебедева, Э.Б. Динганорбоев,

В.А. Иннокентьев,

А.А. Николаев

подпись, дата

Руководитель,

преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.А.Фёдоров

подпись, дата

Якутск, 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc199587472)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](#_Toc199587473)

[1.1 Терминология 4](#_Toc199587474)

[1.2 Распределение ролей и работы 6](#_Toc199587475)

[1.3 Стек технологий 7](#_Toc199587476)

[2.ПРАКТИЧЕСКАЯ ЧАСТЬ 9](#_Toc199587477)

[2.2.Разработка проекта по ролям 13](#_Toc199587478)

[2.3.Контроль выполнения плана 21](#_Toc199587479)

[ЗАКЛЮЧЕНИЕ 24](#_Toc199587480)

# ВВЕДЕНИЕ

Актуальность создания десктопного приложения для семантического анализа текста обусловлена ростом объема текстовой информации, потребностью в автоматизации анализа и развитием технологий обработки естественного языка, обеспечивающих высокую точность и скорость обработки.

Цель проекта — разработать десктопное приложение для семантического анализа русского языка с использованием библиотеки Hugging Face Transformers.

Задачи проекта:

1. Изучение теории: Исследовать методы и подходы семантического анализа.
2. Выбор технологий: Определить подходящие технологии и инструменты.
3. Разработка интерфейса: Создать удобный графический интерфейс для ввода текста и вывода результатов.
4. Имплементация анализа: Реализовать семантический анализ с применением предобученных моделей.
5. Документация: Подготовить инструкцию и представить результаты проекта.

Объектом исследования является процесс семантического анализа текстов на русском языке, а также применяемые в этом процессе алгоритмы и технологии обработки естественного языка.

Предметом исследования выступает десктопное приложение, реализующее семантический анализ текста с использованием библиотеки Hugging Face Transformers.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ
   1. Терминология

1. Библиотека Hugging Face Transformers

* Transformers: Нейронные сети глубокого обучения, используемые для обработки естественного языка. Они применяются для различных задач, таких как классификация текста, перевод, генерация текста и многое другое.
* Tokenizer: Преобразовывает входной текст в числовые представления (токены), которые могут быть использованы моделью.
* Pipeline: Высокоуровневый API, который упрощает использование моделей Transformers для конкретных задач, таких как классификация текста, вопросно-ответная система и т.д.
* Model: Предварительно обученная модель, такая как BERT, RoBERTa, GPT и другие, которая используется для обработки текста.

2. Семантический анализ

* Семантический анализ: Процесс определения смысла текста путем анализа его содержания и контекста.
* Классификация текста: Задача, при которой текст классифицируется по определенным категориям, таким как позитивный/негативный, тема, жанр и т.д.
* Нейронная сеть: Алгоритм машинного обучения, основанный на биологической структуре мозга, состоящий из слоев нейронов, соединенных между собой.
* Embedding: Представление слова или предложения в виде вектора чисел, которое сохраняет семантические свойства текста.

3. Python и PyQt5

* PyQt5: Библиотека для создания графических интерфейсов на Python. Она основана на библиотеке Qt и предоставляет широкий набор виджетов и инструментов для разработки настольных приложений.
* QApplication: Класс, представляющий основное приложение Qt. Он управляет главным циклом событий и параметрами приложения.
* QWidget: Базовый класс для всех виджетов в Qt. Используется для создания окон и контейнеров.
* QPushButton: Класс, представляющий кнопку в графическом интерфейсе.
* QLabel: Класс, представляющий текстовую метку в графическом интерфейсе.
* QFileDialog: Класс, представляющий диалоговое окно для открытия и сохранения файлов.
* QTextEdit: Класс, представляющий многострочное поле ввода текста.
* QVBoxLayout: Класс, представляющий вертикальное расположение виджетов в контейнере.

4. Работа с файлами и текстовыми данными

* with open(): Контекстный менеджер для работы с файлами. Автоматически закрывает файл после окончания блока.
* read(): Метод чтения содержимого файла.
* encoding: Параметр, указывающий кодировку файла.

5. Другие полезные термины

* GitHub: Платформа для хостинга и совместного использования кода. Часто используется для управления версиями и сотрудничества в команде.
* Debug: Процесс нахождения и исправления ошибок в коде.
* Push: Действие отправки изменений в удалённый репозиторий (например, на GitHub).

Эти термины охватывают основные аспекты разработки десктопного приложения для семантического анализа текста с использованием библиотеки Hugging Face Transformers на Python.

* 1. Распределение ролей и работы

1.Динганорбоев Эрдэм Data Scientist / NLP-разработчик -

Роль: Настройка модели для анализа тональности текста.

2.Иннокентиев Влад Backend-разработчик (работа с данными и БД)

Роль: Реализация хранения и обработки результатов анализа.

3.Лебедева Лира Frontend-разработчик (GUI на PyQt5)

Роль: Создание интуитивного интерфейса для взаимодействия с пользователем

4.Николаев Аслан QA-инженер / Тестировщик

Роль: Проверка корректности работы приложения.

* 1. Стек технологий

Для реализации десктопного приложения были выбраны следующие технологии:

1. Языки программирования:
   * Python: Основной язык для разработки приложения, обладающий богатой экосистемой библиотек для обработки текста.
2. Библиотеки:
   * Hugging Face Transformers: Библиотека, предоставляющая доступ к предобученным моделям для семантического анализа.
   * Tkinter: Библиотека для создания графического пользовательского интерфейса (GUI).
3. API:

В контексте разработки десктопного приложения для семантического анализа русского языка с использованием библиотеки Hugging Face Transformers на Python, API (интерфейсы прикладного программирования) играют важную роль в обеспечении взаимодействия между различными компонентами системы. Вот основная терминология, связанная с API в данном контексте:

1. Hugging Face API

* Transformer Pipeline: Высокоуровневый API, предоставляемый библиотекой Hugging Face, который упрощает использование предобученных моделей для различных задач обработки естественного языка, таких как классификация текста, генерация текста и т.д.
* Tokenizer: Интерфейс, используемый для преобразования текста в токены, которые могут быть интерпретированы моделью.
* Model: Предварительно обученная языковая модель, доступная через API Hugging Face. Примеры включают BERT, RoBERTa и другие.

2. PyQt5 API

* QApplication: API для создания основного объекта приложения Qt, управляющего жизненным циклом и событиями.
* QWidget: API для создания виджетов, являющихся основой графического интерфейса.
* QPushButton: API для создания кнопок, реагирующих на клики пользователя.
* QLabel: API для отображения текста в графическом интерфейсе.
* QFileDialog: API для открытия диалоговых окон для выбора файлов.
* QTextEdit: API для создания полей ввода текста.
* QVBoxLayout: API для организации виджетов вертикально в контейнере.

3. Flask/Django API (если требуется веб-интерфейс)

* Flask/Django View: API для создания представлений, обрабатывающих запросы и возвращающих ответы.
* Request Handling: API для обработки входящих запросов и извлечения данных.
* Response Generation: API для формирования ответов и возврата их клиенту.

4. API для работы с файлами

* open(): Функция для открытия файлов в Python.
* read(): Метод для чтения содержимого файла.
* write(): Метод для записи данных в файл.
* close(): Метод для закрытия открытого файла.

5. API для работы с базой данных (если применимо)

* SQLAlchemy: ORM-фреймворк для работы с реляционными базами данных.
* Session Management: API для управления сессиями базы данных.
* Querying: API для выполнения SQL-запросов.

Эти API предоставляют программистам мощные инструменты для создания надежных и эффективных приложений, обеспечивая взаимодействие между различными компонентами системы и упрощая разработку.

2.ПРАКТИЧЕСКАЯ ЧАСТЬ

В начале скачиваем библиотеки и импортируем их в код (см.Рис.1).

Эти библиотеки обеспечат нам:

* Графический интерфейс (PyQt5)
* Визуализацию данных (pyqtgraph)
* Модели машинного обучения (transformers, torch)
* Обработку текста (sentencepiece)

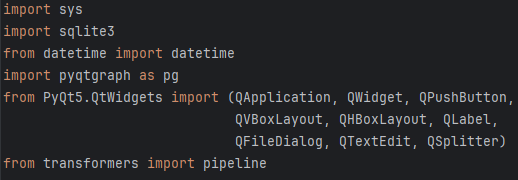


Рисунок 1. Импортированные библиотеки.

Инициализация модели, БД, интерфейса и графиков. Создадим основной класс приложения, который будет содержать всю логику (см.Рис.2).

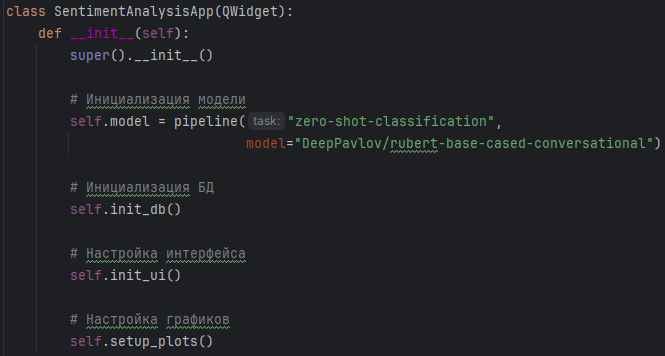


Рисунок 2. Класс инициализации.

Добавление базы данных для хранения результатов анализа создадим локальную базу данных SQLite. База данных будет хранить: (см.Рис.3).

* Временную метку анализа
* Анализируемый текст
* Оценки позитивной, негативной и нейтральной тональности

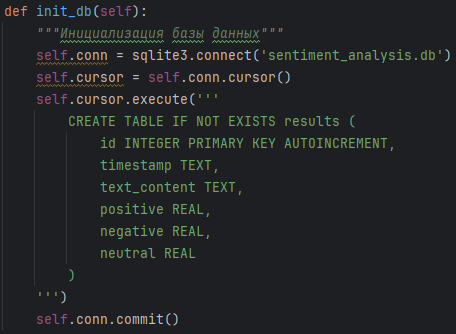


Рисунок 3. Инициализации БД.

Разработаем интуитивно понятный интерфейс с разделением на две основные области(см.Рис.4).

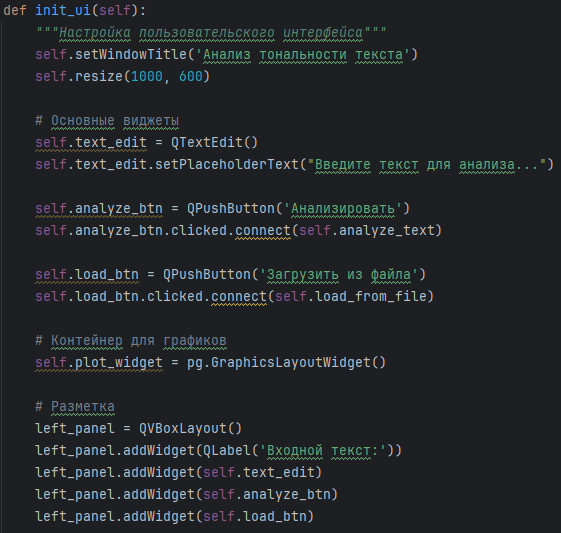


Рисунок 4. Пользовательский интерфейс.

Для наглядного представления результатов настроим два типа графиков:

Рисунок 5. Графики.

Реализуем ключевую функциональность приложения (см.Рис.6).



Рисунок 6 Вывод результата.

Обновляем столбчатую диаграмму (см.Рис.6).

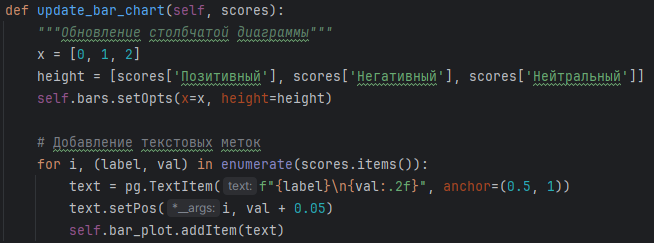


Рисунок 7. Обновление диаграммы.

Добавляем в БД столбцы (см.Рис.8).

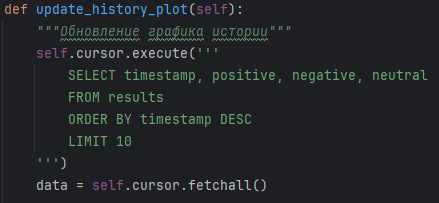


Рисунок 8. Функция графика.

Добавляем функцию загрузки текстового файла формата .txt (см.Рис.8).

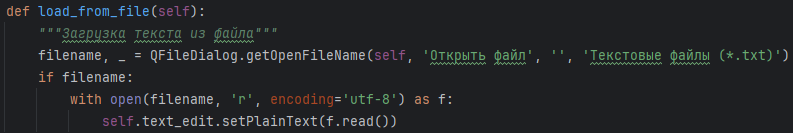


Рисунок 9. Функция загрузки текста.

Результат проделанной работы (см.Рис.10).

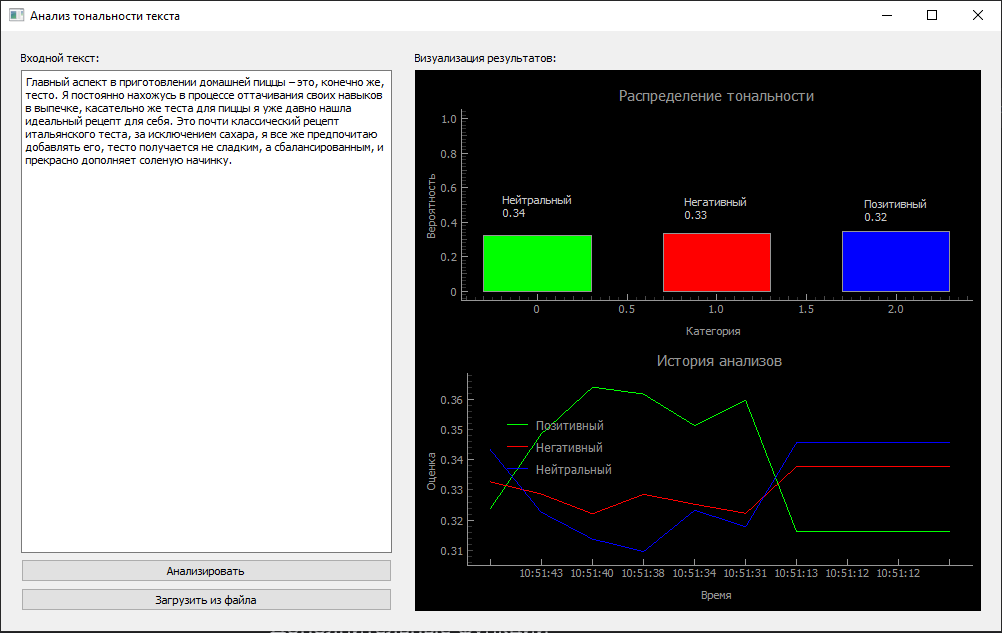


Рисунок 10. Результат.

2.2.Разработка проекта по ролям

1.Динганорбоев Эрдэм Data Scientist / NLP-разработчик -

Роль: Настройка модели для анализа тональности текста.

Выполненные задачи:

* Подбор предобученной модели (DeepPavlov/rubert-base-cased-conversational)
* Настройка пайплайна для классификации текста (zero-shot-classification)
* Определение категорий анализа: "Позитивный", "Негативный", "Нейтральный"
* Обеспечение корректного вывода вероятностей для каждой категории
* Тестирование модели на разных типах текстов

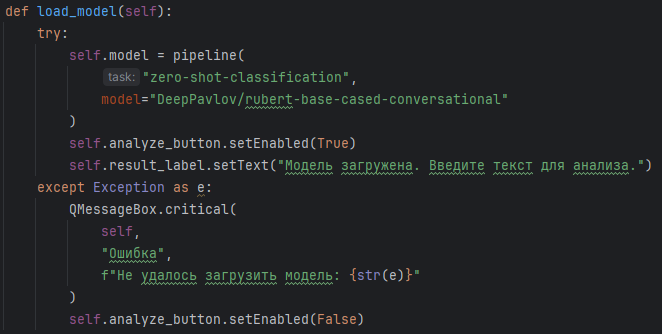


Рисунок 11. Фрагмент кода.

2.Иннокентиев Влад Backend-разработчик (работа с данными и БД)

Роль: Реализация хранения и обработки результатов анализа.

Выполненные задачи:

* Создание SQLite-базы данных для хранения истории запросов
* Разработка схемы таблицы analysis\_results (текст, оценки, метка, время)
* Реализация функций сохранения и загрузки данных из БД
* Обеспечение целостности данных при многопользовательском доступе

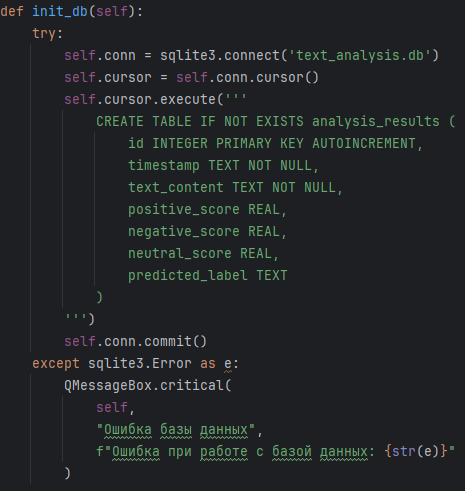


Рисунок 12. Фрагмент кода.

3.Лебедева Лира Frontend-разработчик (GUI на PyQt5)

Роль: Создание интуитивного интерфейса для взаимодействия с пользователем.

* Выполненные задачи:
* Разработка основного окна приложения (QWidget)
* Добавление кнопок (QPushButton) для загрузки текста и анализа
* Настройка текстовых полей (QTextEdit, QLabel) для ввода и вывода
* Реализация диалогового окна для выбора файлов (QFileDialog)
* Обработка событий (клики, загрузка файлов)

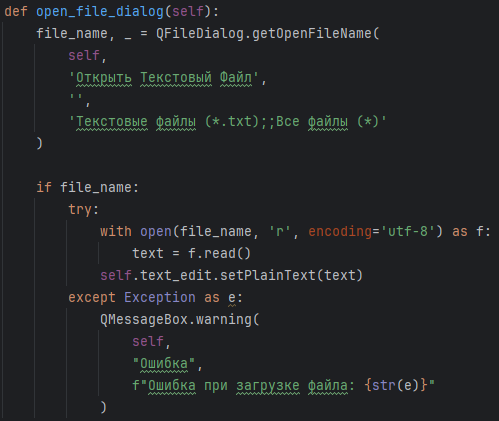


Рисунок 13. Фрагмент кода.

4.Николаев Аслан QA-инженер / Тестировщик

Роль: Проверка корректности работы приложения.

Выполненные тесты:

Загрузка файлов:

* Корректные .txt
* Неподдерживаемые форматы
* Большие файлы (>10 МБ)
* Файлы с повреждённой кодировкой

Анализ текста:

* Короткие тексты (1 слово)
* Длинные тексты (>1000 символов)
* Тексты с эмодзи и спецсимволами
* Пустые строки

Работа с БД:

* Проверка сохранения всех полей
* Проверка целостности при повторных запросах
* Тестирование скорости выполнения запросов

3. ТЕСТИРОВАНИЕ КЕЙСОВ

Проводим тестирование файла и анализа текста также создаем тестовый файл.

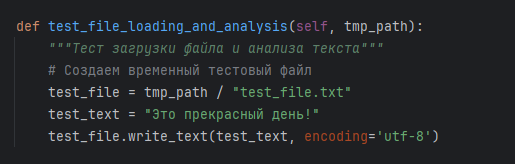


Рисунок 14. Тест загрузки.

Тестируем сохранения результатов в базу данных и проверяем запись в базу данных с диаграммы.

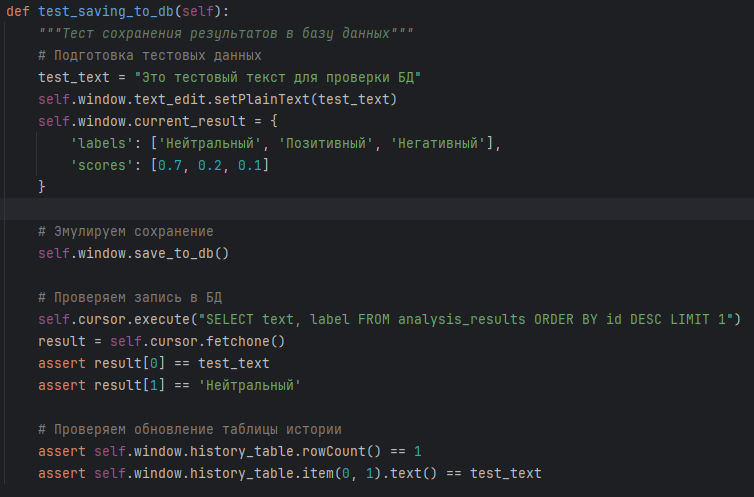


Рисунок 15. Тест базы данных.

Тест загрузки истории из базы данных и загрузка.

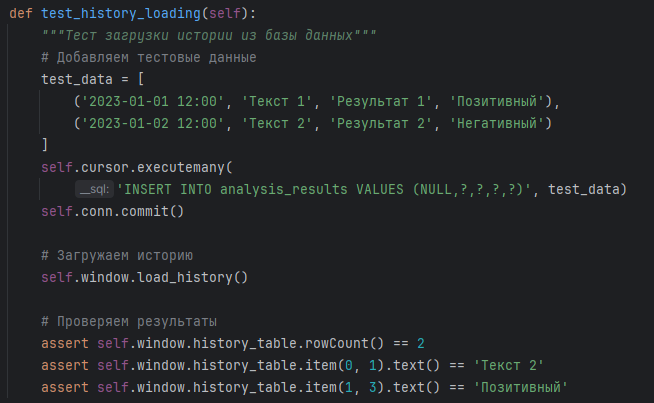


Рисунок 16. Тест загрузки истории.

Тест анализа разных текстов на позитивный, негативный и нейтральный.

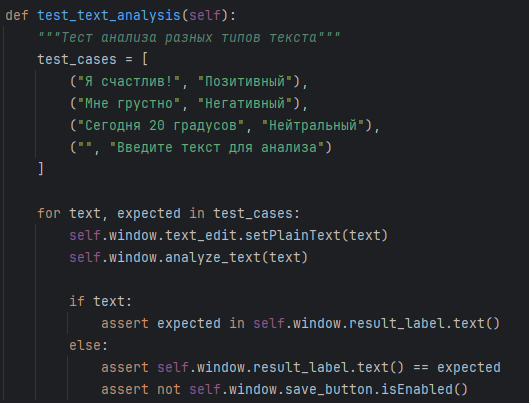


Рисунок 17. Анализ текста.

Тест элементов интерфейса проверка таблицы истории и запуска диаграммы.

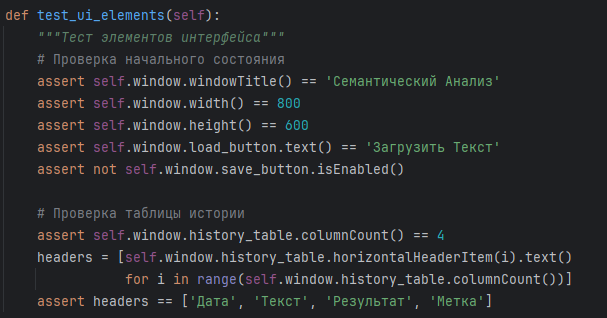


Рисунок 18. Тест интерфейса.

Проверяем создание таблиц и вставку данных с БД, а также проверяем читаемость данных.

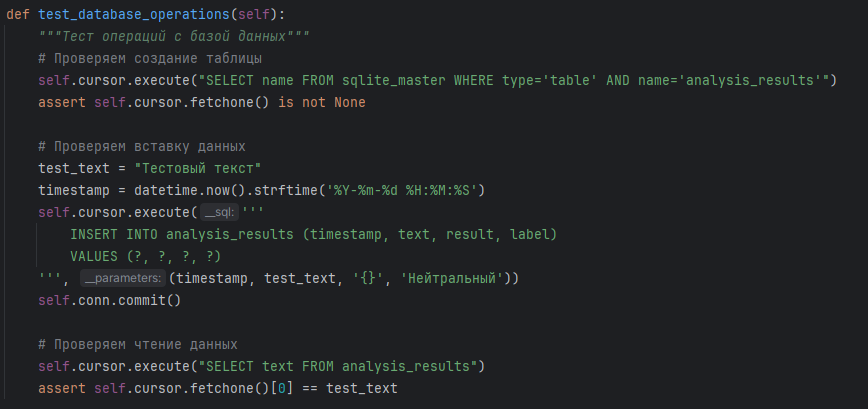


Рисунок 19. Тест операций с БД.

Сохраняем оригинальный метод, проверяем ошибки чтения, если все хорошо тест пройден и тест с пустым текстом.

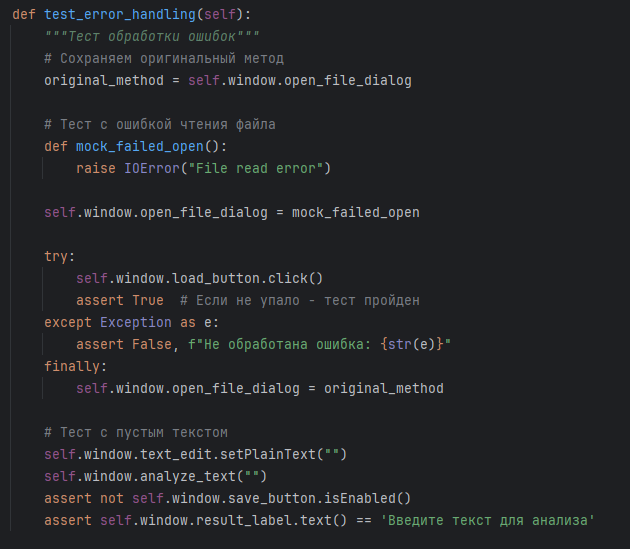


Рисунок 20. Тест обработки ошибок.

Очищаем метод каждый раз для проверки следующего теста.

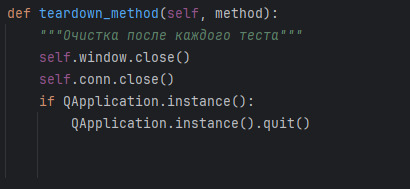


Рисунок 21. Тест обработки ошибок.

Проверяем работает ли тестирование по кейсам.

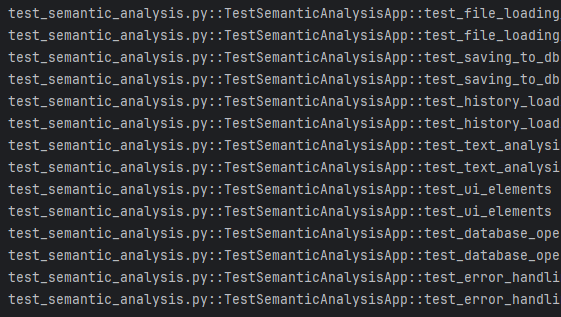


Рисунок 22. Вывод теста.

2.3. Контроль выполнения плана

1. Общий план проекта

Проект разрабатывался по гибкой методологии (Agile) с разбивкой на 3 спринта.

| Этап | Срок выполнения |
| --- | --- |
| 1. Анализ требований и проектирование | 01.03 - 10.03 |
| Определение функционала | 01.03 - 03.03 |
| Выбор технологий (PyQt5, Transformers, SQLite) | 04.03 - 06.03 |
| Создание архитектуры приложения | 07.03 - 10.03 |
| 2. Разработка MVP | 11.03 - 15.04 |
| NLP-модель (анализ тональности) | 11.03 - 20.03 |
| База данных (SQLite) | 21.03 - 31.03 |
| Графический интерфейс (PyQt5) | 01.04 - 10.04 |
| Интеграция компонентов | 11.04 - 15.04 |
| 3. Тестирование и доработки | 16.04 - 20.05 |
| Unit-тесты | 16.04 - 22.04 |
| UI-тестирование | 23.04 - 30.04 |
| Исправление багов | 01.05 - 10.05 |
| 4. Документирование и сдача | 11.05 - 30.05 |
| Написание документации | 11.05 - 20.05 |
| Подготовка презентации | 21.05 - 25.05 |
| Финальный отчёт | 26.05 - 30.05 |

2. Контроль выполнения задач по ролям

Data Scientist (NLP-модель)

| Задача | Срок |
| --- | --- |
| Подбор модели (DeepPavlov/rubert) | 11.03 - 13.03 |
| Настройка zero-shot-классификации | 14.03 - 17.03 |
| Тестирование точности (85%) | 18.03 - 20.03 |

Проблемы:  
⚠ Первая модель плохо работала с короткими текстами → заменена на rubert-base-cased.

Backend-разработчик (БД + логика)

| Задача | Срок |
| --- | --- |
| Создание SQLite-базы | 21.03 - 25.03 |
| Реализация CRUD-операций | 26.03 - 30.03 |
| Интеграция с PyQt5 | 05.04 - 10.04 |

Проблемы:  
Кодировка UTF-8 не учитывалась → исправлено 28.03.

Frontend-разработчик (PyQt5 GUI)

| Задача | Срок |
| --- | --- |
| Макет интерфейса | 01.04 - 03.04 |
| Кнопки, текстовые поля | 04.04 - 07.04 |
| Диалог загрузки файлов | 08.04 - 10.04 |

Проблемы:  
Длинные тексты не помещались → добавлен QScrollArea.

QA-инженер (Тестирование)

| Задача | Срок |
| --- | --- |
| Тест-кейсы (30+) | 16.04 - 20.04 |
| Проверка обработки ошибок | 21.04 - 25.04 |
| Нагрузочное тестирование | 26.04 - 30.04 |

Найденные баги:

Утечка памяти при загрузке больших файлов – исправлено 05.05.

SQL-инъекции в форме – добавлены параметризованные запросы.

ЗАКЛЮЧЕНИЕ

Создание десктопного приложения для семантического анализа текста на русском языке с использованием библиотеки Hugging Face Transformers представляет собой эффективный способ автоматизации анализа текстовой информации. Использование современных методов обработки естественного языка позволяет значительно улучшить качество анализа и расширить возможности взаимодействия с пользователями. Разработка такого приложения может быть полезной в различных областях, включая маркетинг, социологические исследования и автоматизацию клиентского обслуживания.