Министерство образования Республики Беларусь

Учреждение образования

“Белорусский государственный университет

информатики и радиоэлектроники”

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

Отчёт по дисциплине

«Естественно-языковой интерфейс интеллектуальных систем»

Лабораторная работа №2.

Построение и использование корпусов текстов естественного языка

Выполнил:

Дичковский В А

студент группы

221701

Проверил:

Крапивин Ю.Б.

Минск 2024

Цели работы:

1. изучить принципы построения корпусов текстов, виды разметки и способы аннотирования, инструменты работы с корпусами текстов, построить корпус текстов и разработать корпусный менеджер.

Задание

1. Сформировать электронный корпус текстов по выбранной предметной области
2. . 2. Используя результаты лабораторной работы №1 (возможность получения лингвистических сведений для произвольной лексемы естественного языка) разработать корпусный менеджер, обеспечивающий базовую функциональность работы с созданным корпусом текстов.

**Вариант 6** Русский Литература

Программа "Корпусный менеджер" предназначена для работы с текстовыми документами на русском языке. Она позволяет создавать и управлять корпусом текстов, выполнять лингвистический анализ, а также осуществлять поиск по лексемам, словоформам и грамматическим признакам. Приложение имеет графический интерфейс, реализованный с использованием библиотеки Tkinter, и базу данных SQLite для хранения информации.

Пользователь может загружать текстовые файлы в форматах TXT, PDF или DOCX.

Текст документа отображается постранично. Пользователь может листать страницы, кликать на слова для их анализа. При клике на слово программа:

* Находит соответствующую лемму и грамматические признаки в базе данных.
* Отображает контекст употребления слова (конкорданс) с настраиваемым количеством токенов слева и справа.

Поиск поддерживает два режима:

* По лемме
* По словоформе

Фильтры позволяют уточнить запрос по:

* Части речи (существительное, глагол и т.д.).
* Грамматическим признакам (падеж, число, род, время и др.).

Результаты поиска выводятся в виде списка с указанием документа, предложения и контекста. Для каждого найденного токена отображаются его грамматические признаки, переведённые на русский язык

В разделе "Отчёты" пользователь может просмотреть:

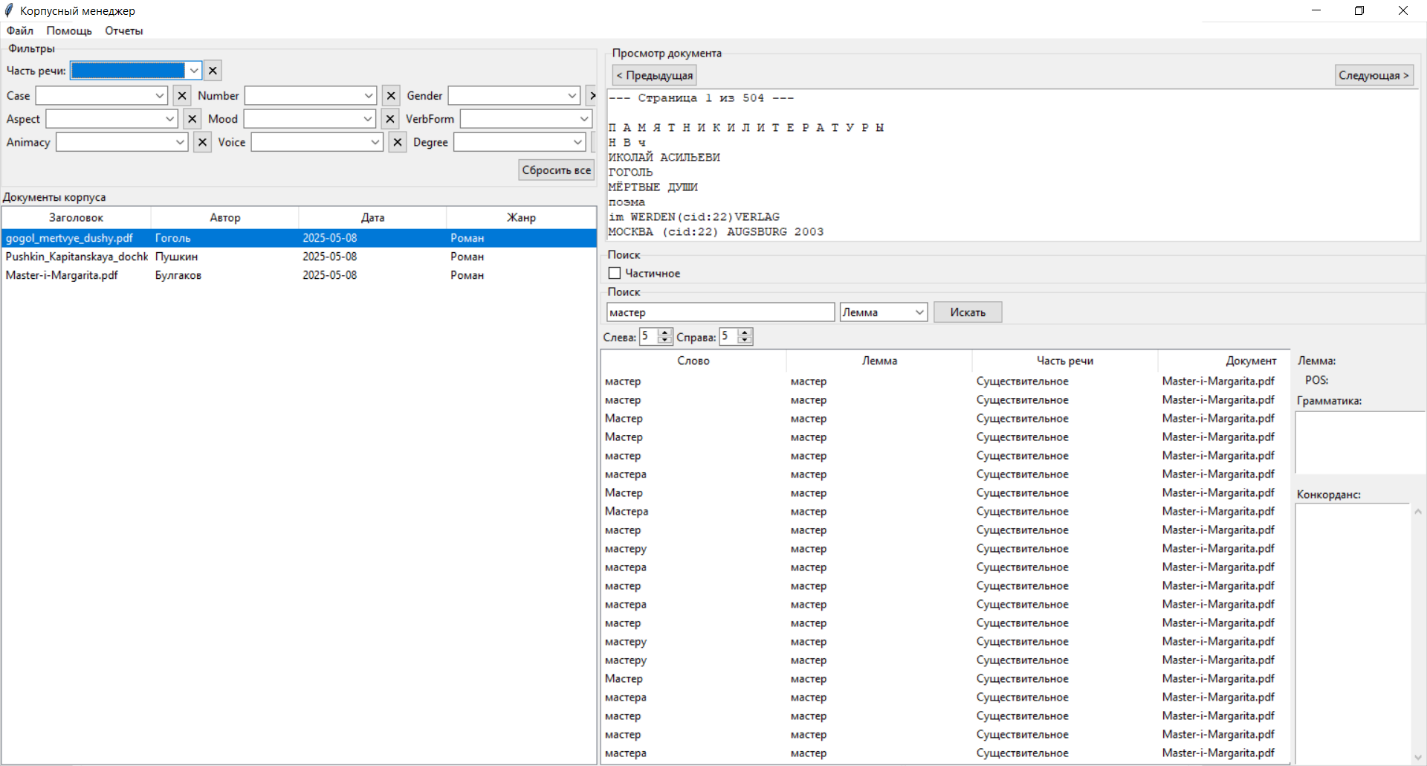
* Время обработки каждого документа в зависимости от количества страниц.
* График, иллюстрирующий зависимость времени обработки от объёма текста (с экспоненциальной аппроксимацией).
* Таблицу с детализацией по каждому документу.

Программа разделена на модули по принципу MVC (Model-View-Controller):

* Модели (models/) отвечают за работу с базой данных и NLP-обработку.
* Контроллеры (controllers/) управляют бизнес-логикой (добавление документов, поиск).
* Представления (views/) реализуют графический интерфейс.

Класс RussianTranslator преобразует коды грамматических признаков (например, "Gen" → "Родительный падеж") для удобства пользователя. Это позволяет работать с фильтрами на русском языке, даже если данные хранятся в стандартизированном формате.

**Интерфейс программы:**



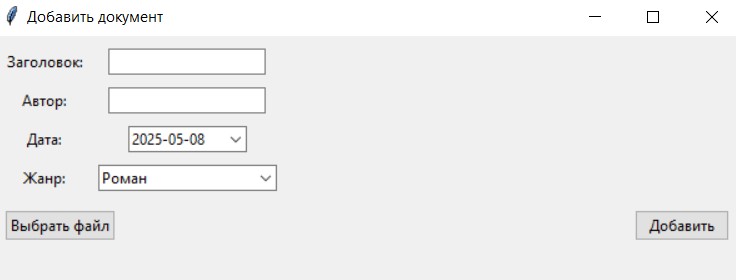
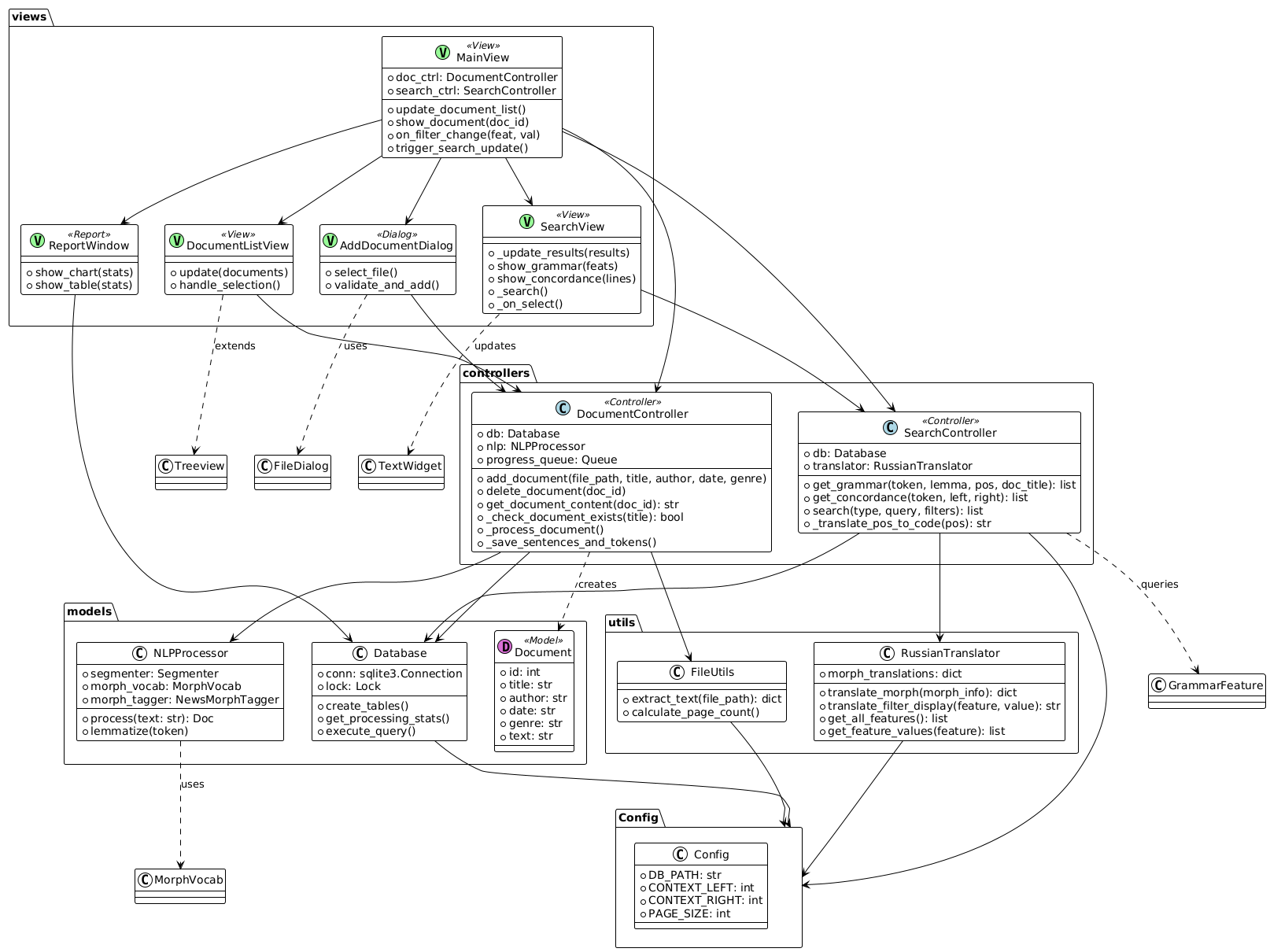


Диаграмма классов:



В проекте используются следующие библиотеки и фреймворки:

Стандартные библиотеки Python:

1. sqlite3  
   Работа с SQLite-базой данных (модель models/database.py).
2. threading  
   Многопоточность для обработки документов (контроллер controllers/document\_controller.py).
3. queue  
   Очереди для передачи сообщений между потоками (контроллер controllers/document\_controller.py).
4. time  
   Измерение времени обработки документов (контроллер controllers/document\_controller.py).
5. dataclasses  
   Создание классов данных (модель models/document.py).
6. typing  
   Аннотации типов (в контроллерах и моделях).

Внешние библиотеки:

1. tkinter  
   Графический интерфейс (основной фреймворк для всех views/\*.py).
2. ttk  
   Расширение tkinter для создания виджетов (все представления).
3. tkcalendar  
   Календарь для выбора даты (диалог добавления документа views/dialogs.py).
4. pdfplumber  
   Извлечение текста из PDF (утилита utils/file\_utils.py).
5. python-docx  
   Обработка DOCX-файлов (утилита utils/file\_utils.py).
6. natasha  
   Обработка естественного языка на русском (модель models/nlp\_processor.py).  
   Используются модули:
   * Segmenter (сегментация текста),
   * MorphVocab, NewsMorphTagger (морфологический анализ),
   * NewsEmbedding (векторизация).
7. matplotlib  
   Построение графиков статистики (отчет views/report\_view.py).  
   Используется FigureCanvasTkAgg для интеграции с tkinter

**Алгоритм обработки нового документа:**

**НАЧАЛО**

1. **Проверка наличия документа с указанным названием**
   * Если документ с таким названием уже существует в базе данных:
     + Завершить обработку с ошибкой "Документ с таким названием уже существует".
   * Если документ не существует:
     + Перейти к пункту 2.
2. **Извлечение текста из файла**
   * Использовать утилиту extract\_text для получения текста и количества страниц.
   * Если текст не извлечен:
     + Завершить обработку с ошибкой "Не удалось извлечь текст".
   * Если текст получен:
     + Перейти к пункту 3.
3. **Обработка текста NLP-моделью**
   * Выполнить сегментацию текста на предложения.
   * Провести морфологический анализ токенов (определение части речи, лемматизация, грамматические признаки).
4. **Сохранение метаданных документа в БД**
   * Записать в таблицу documents:
     + Название, автора, дату, жанр.
     + Исходный текст, время обработки, количество страниц.
   * Получить doc\_id (ID созданной записи).
   * Перейти к пункту 5.
5. **Сохранение предложений в БД**
   * Для каждого предложения из NLP-результата:
     + Добавить запись в таблицу sentences с doc\_id и текстом предложения.
   * Перейти к пункту 6.
6. **Получение ID сохраненных предложений**
   * Запросить из таблицы sentences все ID, связанные с doc\_id.
   * Перейти к пункту 7.
7. **Обработка предложений и токенов**
   * Для каждого предложения (по порядку):
     + Если ID предложения не совпадает с индексом (коррупция данных):
       - Пропустить обработку токенов для этого предложения.
     + Для каждого токена в предложении:
       - **Лемматизация**: Преобразовать токен в лемму.
       - **Сохранение токена**:
         * Записать в таблицу tokens лемму, часть речи, позиции в тексте.
         * Получить token\_id (ID созданной записи).
       - **Извлечение грамматических признаков**:
         * Преобразовать признаки в строку формата ключ=значение.
         * Для каждого признака:

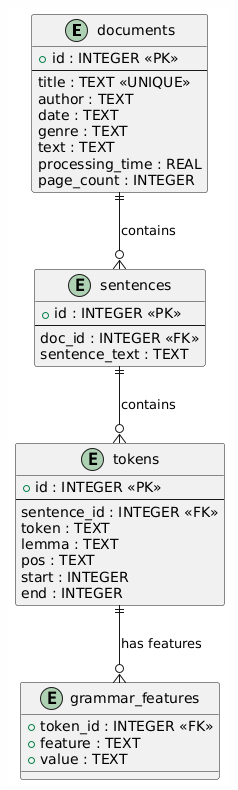
Записать в таблицу grammar\_features связь с token\_id.

* + - Перейти к следующему токену
  + Перейти к следующему предложению

1. **Завершение обработки**
   * Отправить уведомление "Документ успешно добавлен".
   * Вызвать update\_callback для обновления интерфейса.

**КОНЕЦ**

**Схема базы данных**



Пример xml

<?xml version="1.0" encoding="utf-8"?>

<document id="1">

<title>Матер и Маргарита</title>

<author>Михаил Булгаков</author>

<date>2025-05-08</date>

<genre>Роман</genre>

<annotations>

<sentence id="1">

<text>Дверь открылась, и капельдинер втащил толстую пачку только что напечатанных дополнительных афиш</text>

<token id="1">

<text>Дверь</text>

<lemma>дверь</lemma>

<pos>NOUN</pos>

<start>0</start>

<end>5</end>

</token>

<token id="2">

<text>открылась</text>

<lemma>открыться</lemma>

<pos>VERB</pos>

<start>6</start>

<end>14</end>

</token>

<token id="3">

<text>,</text>

<lemma>,</lemma>

<pos>PUNCT</pos>

<start>14</start>

<end>15</end>

</token>

<token id="4">

<text>и</text>

<lemma>и</lemma>

<pos>CCONJ</pos>

<start>16</start>

<end>17</end>

</token>

<token id="5">

<text>капельдинер</text>

<lemma>капельдинер</lemma>

<pos>NOUN</pos>

<start>18</start>

<end>29</end>

</token>

<token id="6">

<text>втащил</text>

<lemma>втащить</lemma>

<pos>VERB</pos>

<start>30</start>

<end>37</end>

</token>

<token id="7">

<text>толстую</text>

<lemma>толстый</lemma>

<pos>ADJ</pos>

<start>38</start>

<end>45</end>

</token>

<token id="8">

<text>пачку</text>

<lemma>пачка</lemma>

<pos>NOUN</pos>

<start>46</start>

<end>51</end>

</token>

<token id="9">

<text>только</text>

<lemma>только</lemma>

<pos>ADV</pos>

<start>52</start>

<end>57</end>

</token>

<token id="10">

<text>что</text>

<lemma>что</lemma>

<pos>SCONJ</pos>

<start>58</start>

<end>61</end>

</token>

<token id="11">

<text>напечатанных</text>

<lemma>напечатать</lemma>

<pos>VERB</pos>

<start>62</start>

<end>73</end>

</token>

<token id="12">

<text>дополнительных</text>

<lemma>дополнительный</lemma>

<pos>ADJ</pos>

<start>74</start>

<end>88</end>

</token>

<token id="13">

<text>афиш</text>

<lemma>афиша</lemma>

<pos>NOUN</pos>

<start>89</start>

<end>93</end>

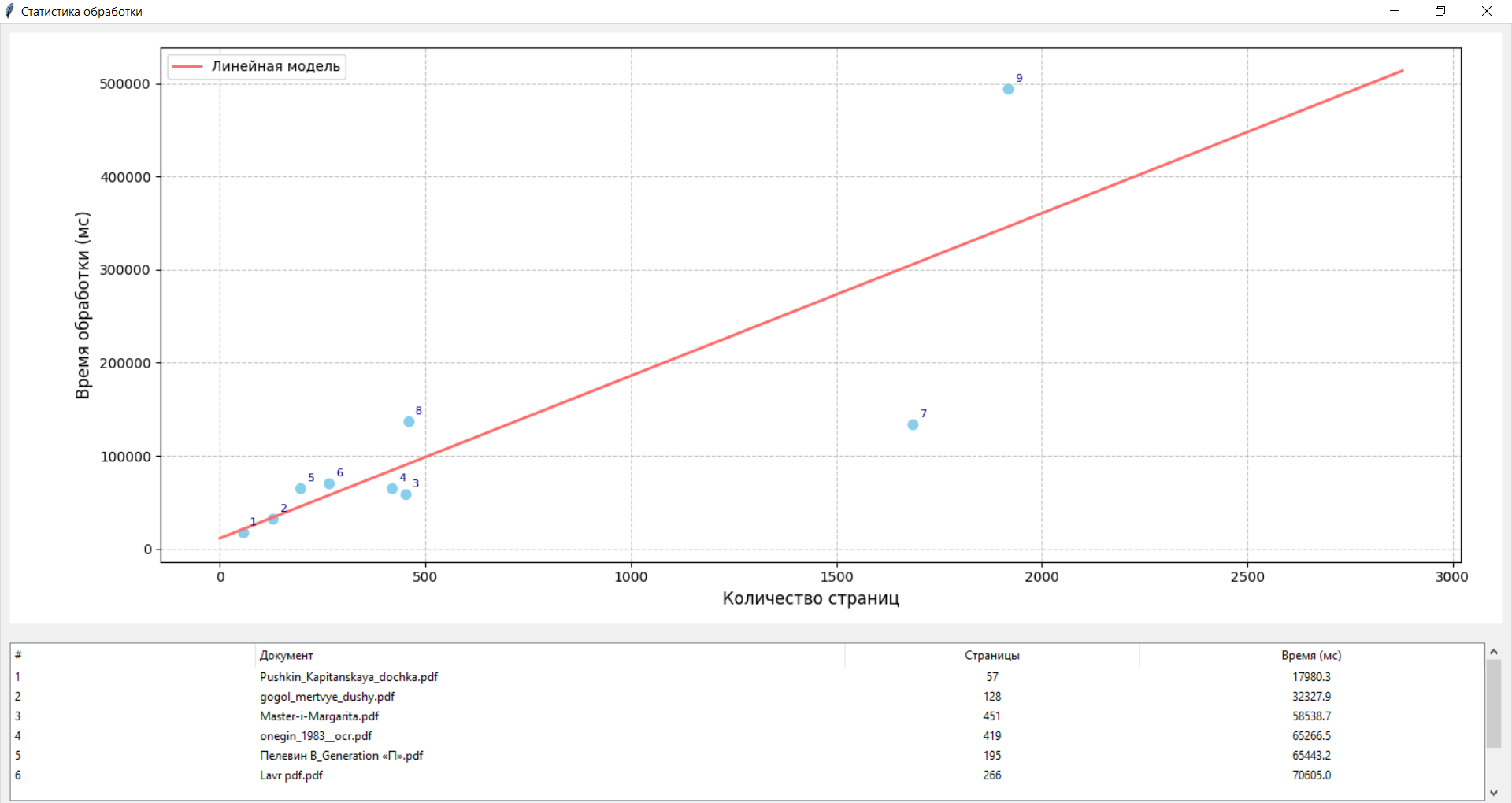
</token>

</sentence>

</annotations>

</document>

График времени обработки текстов от количества страниц



**Вывод**

Программа сочетает функциональность корпусного менеджера и лингвистического анализатора. Она упрощает работу с большими коллекциями текстов, позволяя быстро находить нужные слова, анализировать их употребление и визуализировать данные. Использование асинхронной обработки и продуманный интерфейс делают её удобной для пользователей работающих с русскоязычными текстами.