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 №1.

Разработка автоматизированной системы

формирования словаря естественного языка

Выполнил:

Дичковский В А

студент группы

221701

Проверил:

Крапивин Ю.Б.

Минск 2024

**Цель работы:**

Освоить принципы разработки прикладных сервисных программ для решения задачи автоматического лексического и лексико-грамматического анализа текста естественного языка.

**Задачи лабораторной работы:**

1. Познакомиться с назначением, структурой и функциональностью, предоставляемой базовым ЛП для решения задачи автоматического лексического и лексико-грамматического анализа, ТЕЯ.

2. Закрепить навыки программирования при решении задач автоматической обработки ТЕЯ.

**Вариант 11:** Текст на русском языке в формате PDF, Задание 4

Выходные данные – перечень лексем с дополнительной информацией согласно заданию:

**Задание 4.**

Список слов, упорядоченный по алфавиту и включающий только лексемы с дополнительно оформленными записями о месте и роли данного слова в составе предложения.

К такой информации относится описание того, каким членом предложения может быть данное слово и в какой форме (падеж, число, время и т.п.).

Например, если это существительное в именительном падеже, то оно может выступать в роли подлежащего; если это существительное в родительном падеже, то оно может быть дополнением; если это прилагательное, то оно может быть определением и т.п.

**Описание:**

Данная программа позволяет проанализировать pdf файл с текстом на русском языке.

Программа использует скрипт на python в котором синтаксический роль слови и морфологическая информация определяются с помощью библиотеки Natasha.

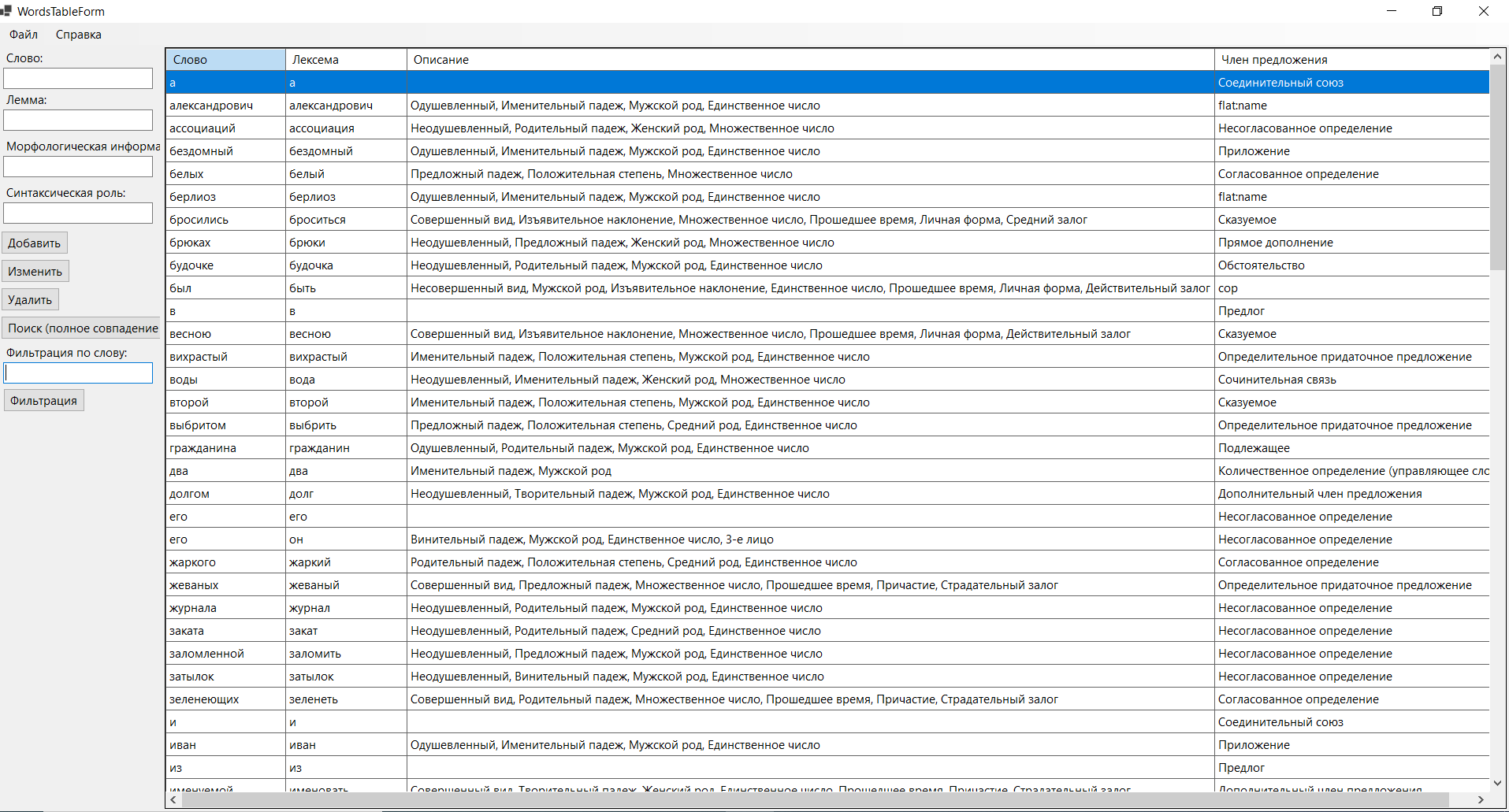
Чтение pdf файлов осуществляется также внутри python скрипта. Сформированный словарь можно сохранить в формате json, есть возможность загрузить готовый словарь либо отредактировать существующий.

В программе реализован поиск по полному совпадению слова, фильтрация по началу слова, а также фильтрация по морфологическому описанию, синтаксической роли и лексеме с использованием регулярных выражений.

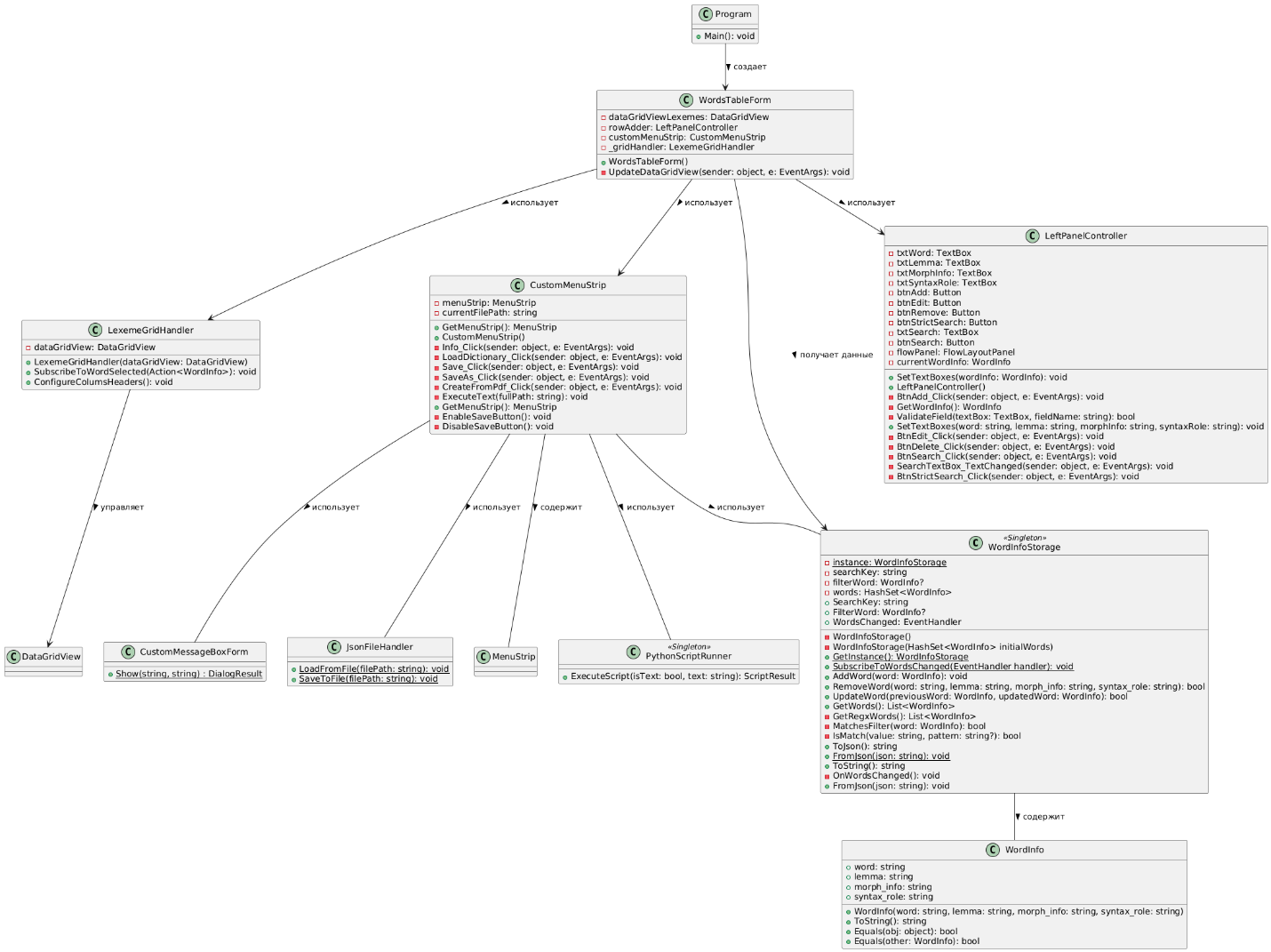
Есть возможность отредактировать все поля с информацией о любом слове и сохранить результат.

Также можно удалить нужное слово.

**Интерфейс:**

****

**Диаграмма классов:**



**Описание python скрипта для анализа текстов:**

Алгоритм, реализованный в скрипте, предназначен для анализа текста на русском языке, включая извлечение морфологической и синтаксической информации. Он использует библиотеки Natasha и PyMuPDF для обработки текста и работы с PDF-документами.

Входные данные

Скрипт принимает два аргумента из командной строки:

1. **arg\_type**: Определяет тип входных данных:
   * 1: Анализ PDF-документа.
   * 2: Анализ текстовой строки.
2. **second\_arg**: Путь к PDF-файлу или текстовая строка в зависимости от значения первого аргумента.

Выходные данные

Результатом работы скрипта является JSON-строка, содержащая информацию о каждом токене в тексте, включая:

* **word**: Исходное слово в нижнем регистре.
* **lemma**: Лемма слова.
* **morph\_info**: Морфологическая информация (падеж, число, род и т.д.).
* **syntax\_role**: Синтаксическая роль слова в предложении.

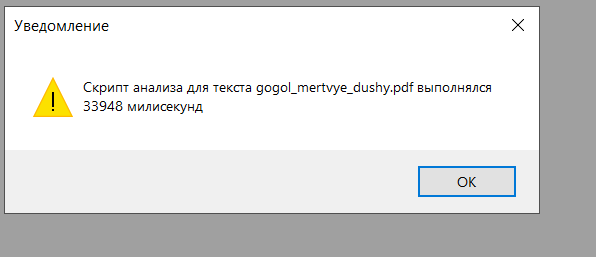
Используемые библиотеки и их функции:

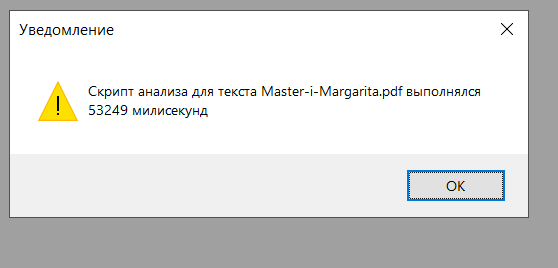
1. **Natasha**:
   * Segmenter: Для сегментации текста на предложения и токены.
   * MorphVocab: Для работы с морфологической информацией.
   * NewsEmbedding: Для загрузки предобученных эмбеддингов.
   * NewsMorphTagger: Для морфологического тегирования слов.
   * NewsSyntaxParser: Для синтаксического анализа.
   * Doc: Основной класс для работы с текстом.
2. **PyMuPDF (fitz)**:
   * Используется для извлечения текста из PDF-документов.
3. **json**: Для сериализации данных в формат JSON.
4. **sys**: Для работы с аргументами командной строки.

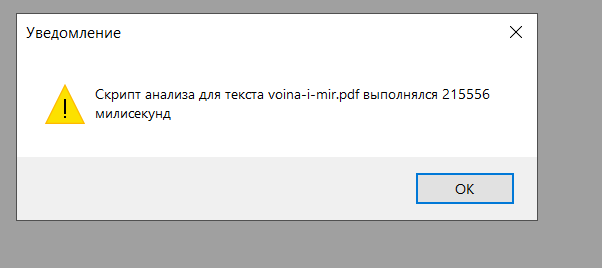
**Функции**

1. **translate\_morph(morph\_info)**:
   * Переводит морфологическую информацию на русский язык, используя словари перевода.
2. **translate\_syntax(syntax\_role)**:
   * Переводит синтаксическую роль слова на русский язык.
3. **analyze\_text(text)**:
   * Основная функция анализа текста. Она выполняет следующие шаги:
     + Сегментация текста на предложения и токены.
     + Тегирование токенов по частям речи и морфологии.
     + Синтаксический анализ предложений.
     + Сбор информации о каждом токене и формирование JSON-выхода.
4. **extract\_text\_from\_pdf(pdf\_path)**:
   * Извлекает текст из PDF-документа, открывая его с помощью PyMuPDF и собирая текст со всех страниц.
5. **main**:
   * Обрабатывает входные аргументы, вызывает функции для извлечения текста (из PDF или из строки) и анализирует полученный текст.

Скорость работы:







**Алгоритм функции Analize Text:**

**Начало**

1.1. Создаются экземпляры необходимых классов:  
   1.1.1. **Segmenter** – для разделения входного текста на предложения и токены.  
   1.1.2. **MorphVocab** – словарь для поддержки лемматизации токенов.  
   1.1.3. **NewsEmbedding** – модель эмбеддингов для векторного представления слов.  
   1.1.4. **NewsMorphTagger** – морфологический таггер, использующий эмбеддинги для определения грамматических признаков.  
   1.1.5. **NewsSyntaxParser** – синтаксический парсер, анализирующий синтаксическую структуру предложений.  
   1.1.6. **RussianTranslator** – объект, который содержит словари для перевода технических обозначений морфологии и синтаксиса на понятные русские описания.

**Предобработка и анализ текста**2.1. Создается объект **Doc**, в который передается исходный текст.  
2.2. Применяется сегментация текста:  
   2.2.1. Метод **segment()** объекта Doc разбивает текст на предложения и токены с использованием Segmenter.  
2.3. Выполняется морфологическая разметка:  
   2.3.1. Метод **tag\_morph()** определяет грамматические признаки каждого токена с помощью NewsMorphTagger.  
2.4. Выполняется синтаксический анализ:  
   2.4.1. Метод **parse\_syntax()** строит синтаксическое дерево предложений, определяя связи между токенами с помощью NewsSyntaxParser.

**Обработка каждого токена (основной цикл)**3.1. Инициализируется пустой список **words\_data**, в который будут сохраняться результаты анализа каждого слова.  
3.2. Итерация по предложениям документа:  
   3.2.1. Для каждого предложения запускается вложенный цикл по токенам, содержащимся в этом предложении.  
   3.2.2. **Проверка токена на соответствие условиям обработки:**     3.2.1. Токен обрабатывается далее только если состоит исключительно из русских букв

3.3. **Обработка токена, удовлетворяющего условиям:**   3.3.1. Выполняется лемматизация токена с использованием метода **lemmatize()** и объекта MorphVocab.  
   3.3.2. Извлекаются следующие данные:  
     3.3.2.1. **Исходный текст токена** (приводится к нижнему регистру).  
     3.3.2.2.Получаем **Лемму** токена.  
     3.3.2.3. Получаем **Морфологические признаки** (атрибут **feats** токена).  
     3.3.2.4. Получаем **Синтаксическуюсвязь** (атрибут **rel** токена).

3.4. **Перевод и форматирование данных токена:**     3.4.1. Морфологические признаки переводятся в понятный формат:  
       3.4.1.1. С помощью метода **translate\_morph()** объекта RussianTranslator каждый признак заменяется на соответствующее русское описание.  
       3.4.1.2. Полученные переводы объединяются в строку, где признаки разделяются запятой.

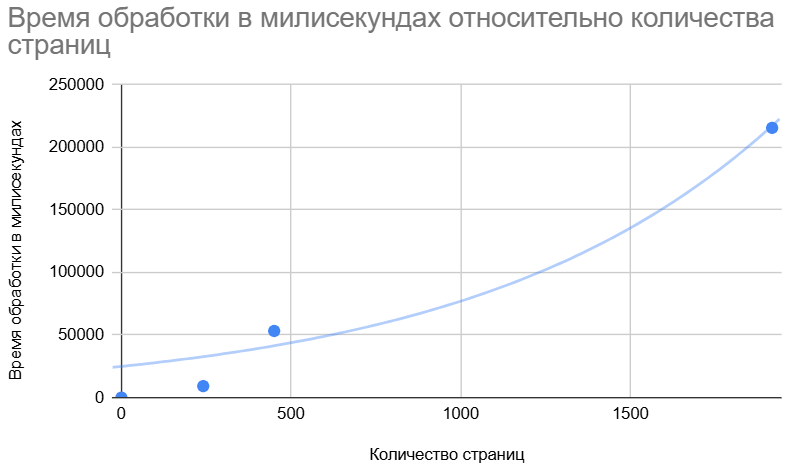
3.4.2. Синтаксическая связь токена переводится с помощью метода **translate\_syntax()**.

3.5. **Формирование структуры данных для токена:**     3.5.1. Создается словарь **word\_data** с ключами:  
       - **"word"** – исходный текст токена (в нижнем регистре).  
       - **"lemma"** – лемматизированная форма токена.  
       - **"morph\_info"** – строка с переведенными морфологическими признаками.  
       - **"syntax\_role"** – переведенная синтаксическая связь.      3.5.2. Словарь **word\_data** добавляется в список **words\_data**.  
3.6. **Переход к следующему токену:**     После завершения обработки текущего токена управление возвращается к началу вложенного цикла для обработки следующего токена в текущем предложении.

3.7. **Переход к следующему предложению:**     После обработки всех токенов в одном предложении, управление переходит к следующему предложению в документе.

**Формирование и возврат итогового результата**4.1. После завершения обработки всех предложений и токенов, список **words\_data** содержит данные по каждому обработанному слову.  
4.2. Список преобразуется в формат JSON с помощью функции **json.dumps()** (с параметрами ensure\_ascii=False и форматированием отступов для удобства чтения).  
4.3. JSON-строка возвращается как итоговый результат работы функции.

Конец



**Структура хранения данных:**

{

"word": "александр",

"lemma": "александр",

"morph\_info": "Одушевленный, Именительный падеж, Мужской род, Единственное число",

"syntax\_role": "подлежащее"

}

**Вывод:**

Программа выполняет задание, описанное в условии лабораторной работы.

Программа учитывает корректный и некорректный формат файлов. Она не прекращает работать в случае неправильных данных, а просто сообщает об этом пользователю.

При программировании был применён паттерн Singleton.

После изучения разработанной структуры хранения данных я пришёл к выводу, что было бы рациональнее использовать табличную базу данных в связке с ORM например Entity Framework.

  Разработанный скрипт на python это мощный инструмент для анализа русского языка с использованием современных библиотек NLP и обработки документов.

Интерфейс, разработанный на C# Windows Forms удобен для модификации и расширения. Код задокументирована и это позволяет легко его расширять.