Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

ОТЧЁТ

по лабораторным работам № 3, № 4

по дисциплине

Вариант 1

Естественно-языковой интерфейс интеллектуальных систем

“Синтаксический анализ текстов естественного языка. Семантико-синтаксический анализ текстов естественного языка”

| Выполнили | студенты группы 221701:  Абушкевич А. А.  Робилко Т. М. |
| --- | --- |
| Проверил | Крапивин Ю. Б. Н.В |
|  |  |

Минск 2025

**Лабораторная работа № 3:**

**Цель**: освоить принципы разработки прикладных сервисных программ для решения задачи анализа текста естественного языка.

**Постановка задачи**:требуется спроектировать и программно реализовать структуры хранения данных, алгоритмы их обработки (в виде подключаемой библиотеки функций), необходимые в рамках следующих базовых требований к разрабатываемому приложению:

* Входные данные – текст заданного естественного языка;
* Выходные данные – структуры, полученные при проведении автоматического синтаксического анализа предложений входного текста согласно варианта задания;
* Взаимодействие с пользователем посредствам графического интерфейса (интерфейс должен быть интуитивно-понятным и дружественным пользователю);
* Наличие системы средств помощи пользователю;
* Обеспечение возможности построения, сохранения, просмотра, редактирования, пополнения, фильтрации и поиска по заданному условию, документирования автоматически получаемого словаря либо заданной его части;
* Поддержка различных форматов представления входных данных.

Вариант 1. Язык текста - русский. Формат входного документа - TXT.

**Средства разработки**

Приложение было разработано в клиент-серверной архитектуре. Сервер реализовывался на языке Python, клиент — на C#. Для языка Python использованы следующие библиотеки:

* библиотека Natasha (для обработки текста естественного языка);
* библиотека enum (для определения структуры данных);
* библиотека dataclasses (для создания read-only структур данных);
* библиотека typing (для использования аннотированных типов данных);
* библиотека logging (для фиксирования данных в работе веб-ресурсов);
* библиотека pydantic (для определения моделей данных и правил их проверки);
* библиотека datetime (для работы с реальным временем);
* фреймворк FastApi (для реализации серверной части приложения);
* фреймворк Avalonia для С# (для реализации клиентной части приложения);
* Система контроля версий Git;

Подробное описание функционала библиотек представлено далее.

**Лабораторная работа № 4:**

**Цель**: освоить принципы разработки прикладных сервисных программ для решения задачи автоматического семантико-синтаксического анализа текста естественного языка.

**Постановка задачи**:требуется спроектировать и программно реализовать структуры хранения данных, алгоритмы их обработки (в виде подключаемой библиотеки функций), необходимые в рамках следующих базовых требований к разрабатываемому приложению:

* Входные данные – текст заданного естественного языка;
* Выходные данные – структуры, полученные при проведении автоматического синтаксического анализа предложений входного текста согласно варианта задания;
* Взаимодействие с пользователем посредствам графического интерфейса (интерфейс должен быть интуитивно-понятным и дружественным пользователю);
* Наличие системы средств помощи пользователю;
* Обеспечение возможности построения, сохранения, просмотра, редактирования, пополнения, фильтрации и поиска по заданному условию, документирования автоматически получаемого словаря либо заданной его части;
* Поддержка различных форматов представления входных данных.

Вариант 1. Язык текста - русский. Формат входного документа - TXT.

**Средства разработки**

Средства разработки для приложения ЛР4 представлены в разделе “Лабораторная работа №3”.

**Алгоритм работы приложения:**

* Открытие файла:

1. Выбрать пункт меню “Открыть файл”;
2. Выбрать файл из проводника;
3. Чтение файла;

5.1. Если файл имеет неверный формат, перейти к пункту 5;

5.2. Если файл имеет верный формат, перейти к пункту 4;

1. Отображение содержимого файла в приложении;
2. Конец.

* Удаление файла:

1. Выбор из списка файлов конкретного файла;
2. Нажать на значок крестика;
3. Файл удаляется из списка доступных;
4. Конец.

* Сохранение файла:

1. Открытие файла;
2. Выполнить операции над текстом;
3. Выбрать пункт меню “Сохранить файл”;
4. Сохранение файла;
5. Конец.

* Редактирование файла:

1. Выбор конкретного файла;
2. Перейти в рабочую область;
3. Поставить курсор на область с текстом;
4. Редактировать текст;
5. Сохранить изменения;
6. Конец.

* Деление на предложения:

1. Выбор конкретного файла;
2. Перейти в рабочую область;
3. Нажать кнопку “Разделить на предложения”;
4. Отправка текста в сервер;
5. Вызов функции разделения текста на предложения;

5.1. Вызов функции sentenize библиотеки natasha для разделения на предложения;

5.2. Текст делится на предложения;

5.3. Функция завершает свою работу и возвращает список строковых объектов;

1. Вернуть список предложений в представление;
2. Конец.

* Морфологический разбор:

1. Выбор конкретного файла;
2. Перейти в рабочую область;
3. Нажать кнопку “Разделить на предложения”;
4. Перейти к конкретному предложению;
5. Выбрать кнопку “Морфологический разбор”;
6. Отправка текста предложения в сервер;
7. Вызов функции выполнения морфологического разбора;

7.1. Разбиение входного текста на токены

7.2. Определить с помощью функции tag\_morph библиотеки natasha часть речи токена;

7.3. Определить с помощью библиотеки natasha грамматические характеристики слова;

7.4. Произвести лемматизацию слова с помощью функции lemmatize библиотеки natasha;

7.5. Функция завершает свою работу и возвращает токены;

1. Вернуть из функции в представление токены части речи, леммы и позиции;
2. Конец.

* Синтаксический разбор:

1. Выбор конкретного файла;
2. Перейти в рабочую область;
3. Нажать кнопку “Разделить на предложения”;
4. Перейти к конкретному предложению;
5. Выбрать кнопку “Синтаксический разбор”;
6. Отправка текста предложения в сервер;
7. Вызов функции выполнения синтаксического разбора;

7.1. Взять сведения из морфологического разбора: часть речи и грамматические характеристики;

7.2. С помощью функции parse\_syntax библиотеки natasha определить синтаксические связи между словами в предложении;

7.3. Построить синтаксическое дерево зависимостей;

7.4. Присвоить каждому слову токен члена предложения;

7.5. Функция завершает свою работу и возвращает токены члена предложения;

1. Вернуть из функции в представление токены члена предложения, позиции и отношения между членами предложения;
2. Конец.

* Семантический разбор:

1. Выбор конкретного файла;
2. Перейти в рабочую область;
3. Нажать кнопку “Разделить на предложения”;
4. Перейти к конкретному предложению;
5. Выбрать кнопку “Семантический разбор”;
6. Отправка текста предложения на сервер;
7. Вызов функции выполнения семантического разбора;

7.1. Выполнить морфологический разбор;

7.2. Выполнить синтаксический разбор;

7.3. Определить именованные сущности с помощью функции tag\_net библиотеки natasha;

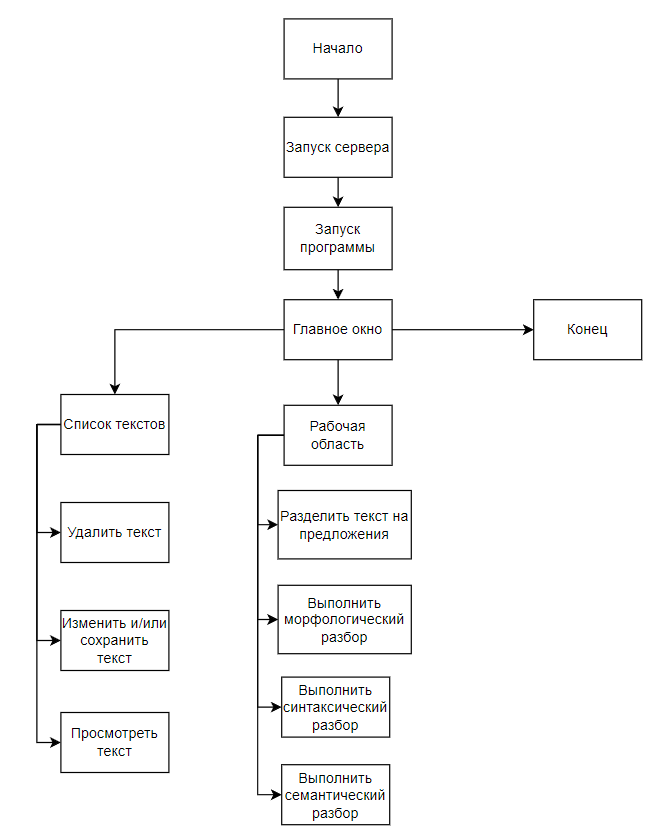
7.4. Распознать и классифицировать сущности организаций, имен собственных и локаций;

7.5. Сформировать токены для распознанных сущностей;

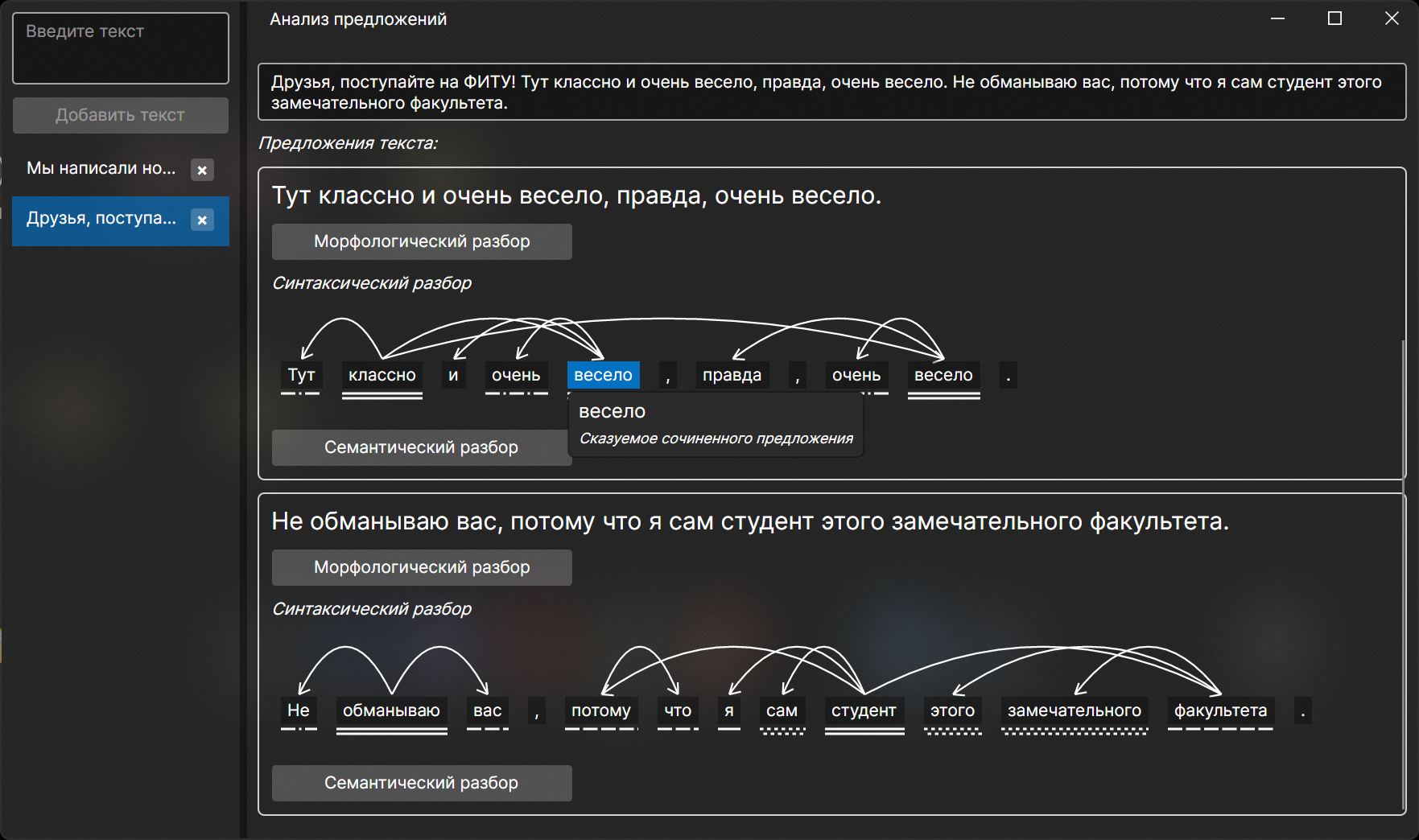
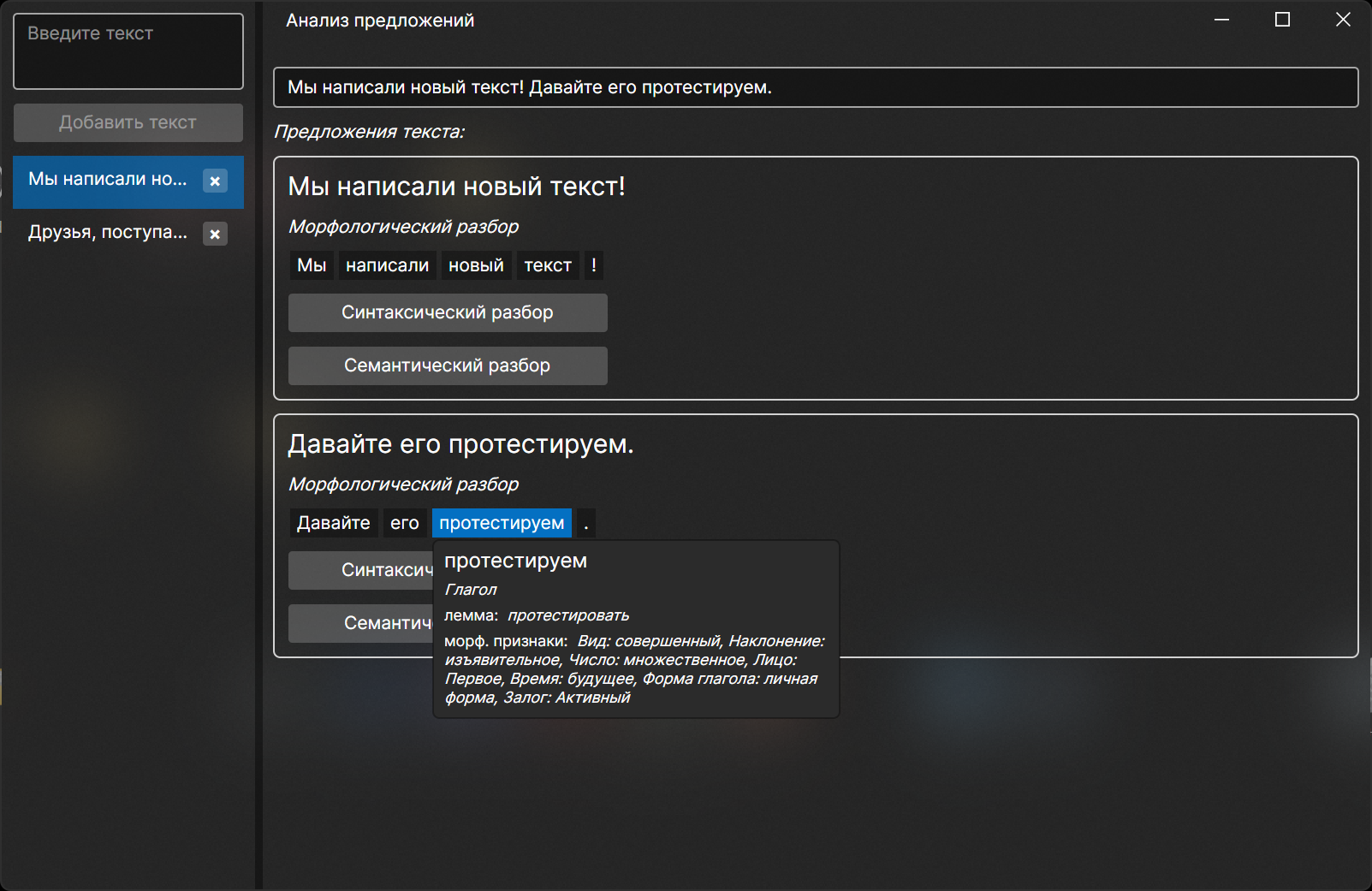
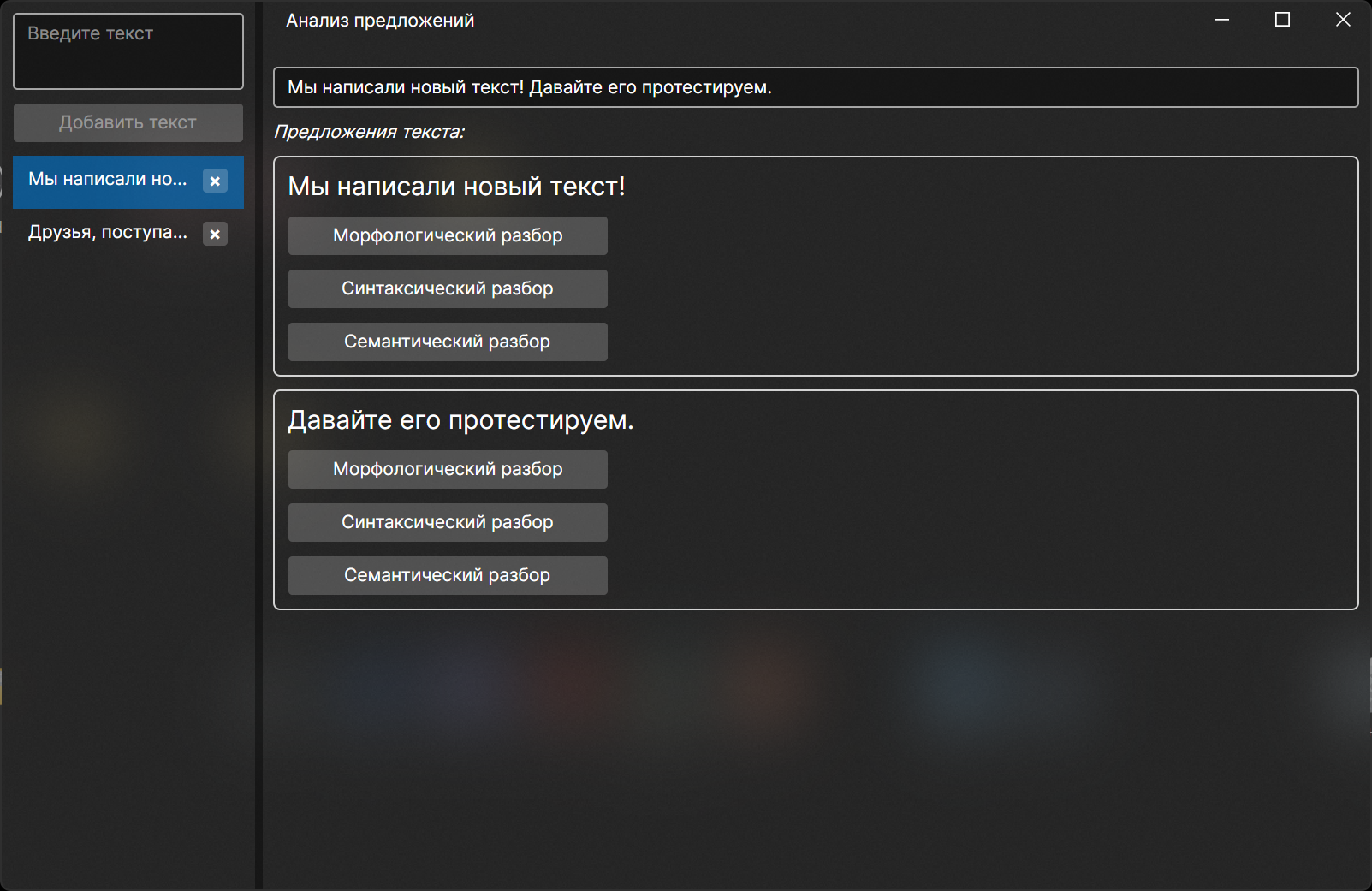
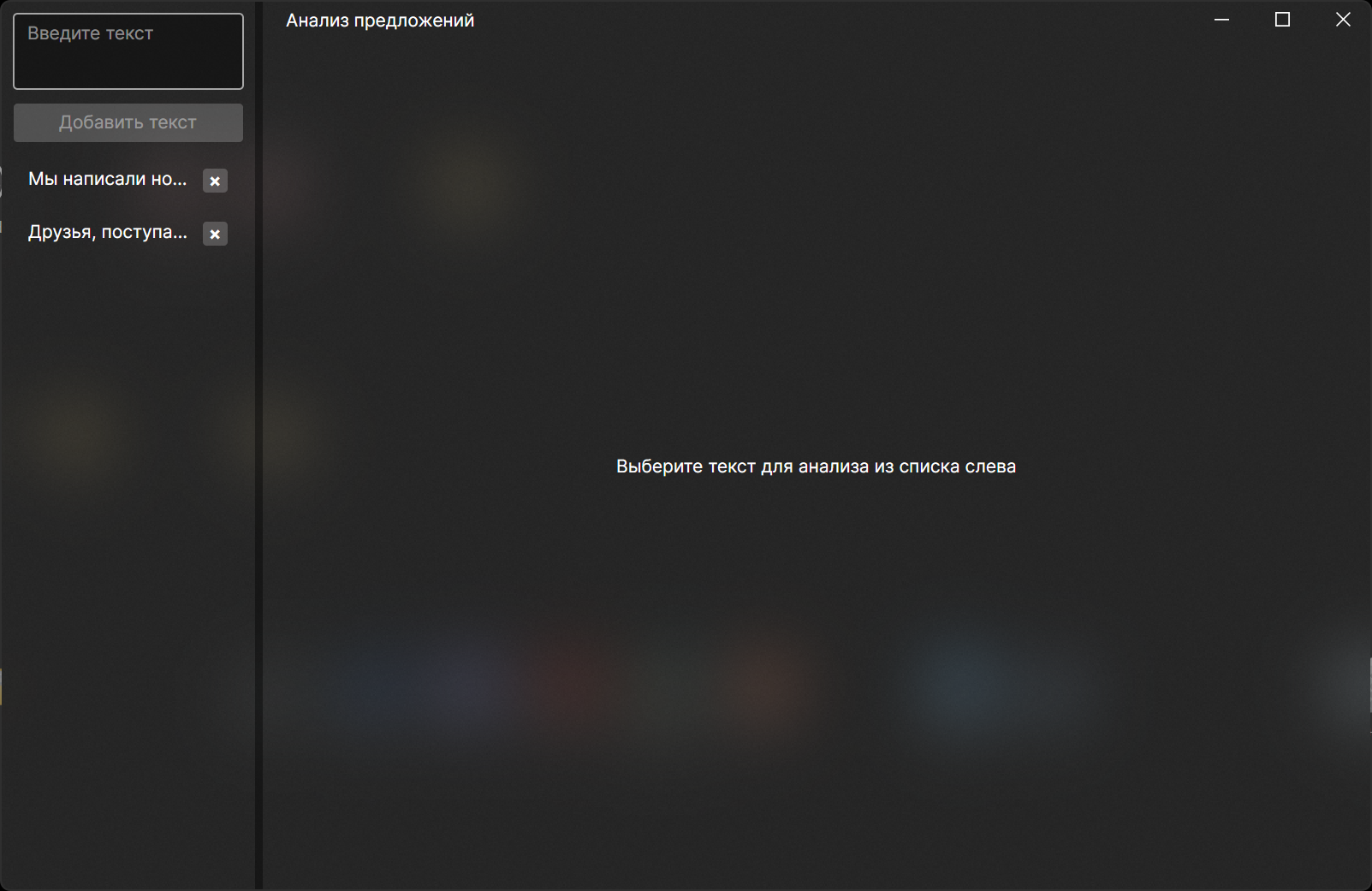
7.6. Функция завершает свою работу и возвращает список токенов для распознанных сущностей;

1. Вернуть из функции в представление токены именных частей речи и описание именной части речи;
2. Конец.

**Структурно-функциональная схема разработанного приложения:**

****

**Интерфейс приложения:**

****

**Описание структур хранения данных:**

* Класс SyntaxRelation:

Enum-класс для представления перечисления, содержащего члены предложений.

* Класс PartOfSpeech:

Enum-класс для представления перечисления, содержащего части речи.

* Класс Syntax:

Хранит синтаксическую информацию о токене:

* id - позиция в предложении;
* head\_id - ссылка на головной элемент в синтаксической структуре;
* relation - тип синтаксической связи;
* Класс Morphology:

Содержит морфологический анализ предложения:

* pos - часть речи;
* lemma - нормальная форма слова;
* morph\_info - морфологические признаки;
* Класс SentenceToken:

Представляет токен в предложении:

* start/end\_idx - позиции в исходной строке;
* syntax - синтаксические характеристики;
* morphology - морфологические характеристики;
* Класс SentenceSyntax:

Представляет синтексическую структуру всего предложения:

* список Syntax;
* Класс SentenceMorphology:

Морфологический анализ всего предложения:

* список Morphology;
* Класс SentenceSemantics:

Хранит семантическую информацию:

* entities - распознанные именованные сущности;
* object\_description - семантическая информация об объекте;
* Класс Sentence:

Объединяет всю информацию о предложении:

* text - исходный текст;
* tokens - список токенов;
* syntax - синтаксический анализ;
* semantics - семантическая информация.

**Описание использованных библиотек:**

1. **Библиотека natasha:**

Библиотека Natasha решает базовые задачи обработки естественного русского языка: сегментация на токены и предложения, морфологический и синтаксический анализ, лемматизация, извлечение именованных сущностей. Для новостных статей качество на всех задачах сравнимо или превосходит существующие решения. Библиотека поддерживает Python 3.5+ и PyPy3, не требует GPU, зависит только от NumPy.

Она основана на принципах машинного обучения и использует различные модели для обработки текстов.

Для использования в работе была выбрана за удобство использования и достойные результаты в обработке текстов русского языка.

1. **Библиотека enum:**

Модуль enum содержит в себе тип для перечисления значений с возможностью итерирования и сравнения. Его можно использовать для создания понятных обозначений вместо использования чисел (для которых приходится помнить, какое число что обозначает) или строк (в которых легко опечататься и не заметить). В рамках данного приложения библиотека использована для представления перечислений частей речи и членов предложения.

1. **Библиотека dataclasses:**

Библиотека dataclasses в Python предназначена для упрощения создания классов, которые в основном используются для хранения данных. Она появилась в Python 3.7 и с тех пор стала стандартным инструментом для описания простых «структур данных». В данной лабораторной работе применялась в качестве представления моделей для аннотирования типов.

1. **Библиотека typing:**

Библиотека typing — это стандартный модуль Python, предназначенный для аннотации типов переменных, аргументов и возвращаемых значений функций. Она не влияет на выполнение кода, но улучшает читаемость, делает код более надёжным и помогает инструментам статической проверки типов, таким как mypy, Pyright, Pylance, и т.п.

1. **Библиотека logging:**

Стандартный модуль Python для гибкой организации логирования (записи событий, сообщений, ошибок) в приложениях. В данной ЛР применялась для логгирования информации в консоль для отладки.

1. **Библиотека pydantic:**

Сторонняя библиотека для валидированной и типизированной модели данных. Часто используется в FastAPI, для работы с конфигурационными файлами и API. Класс BaseModel определяет модели с полями и аннотацией типов.

1. **Библиотека datetime:**

Стандартный модуль для работы с датой и временем. Применялась для проверки корректности работы сервиса.

1. **Фреймворк FastAPI:**

Асинхронный веб‑фреймворк для создания API на Python на базе Starlette и Pydantic. Из ключевых особенностей можно выделить:

1. высокая производительность;
2. авто-документация;
3. типобезопасность;
4. асинхронность;
5. простая система внедрения зависимостей;

В данной лабораторной работе выступает в роли сервера.

1. **Фреймворк Avalonia UI**

Свободный и открытый .NET‑фреймворк для построения десктопных приложений с единой кодовой базой под Windows, Linux и macOS. В данном приложении использовался для реализации клиента, который через HttpClient или WebSockets обращается к API, получает/отправляет данные, отображает их в UI.

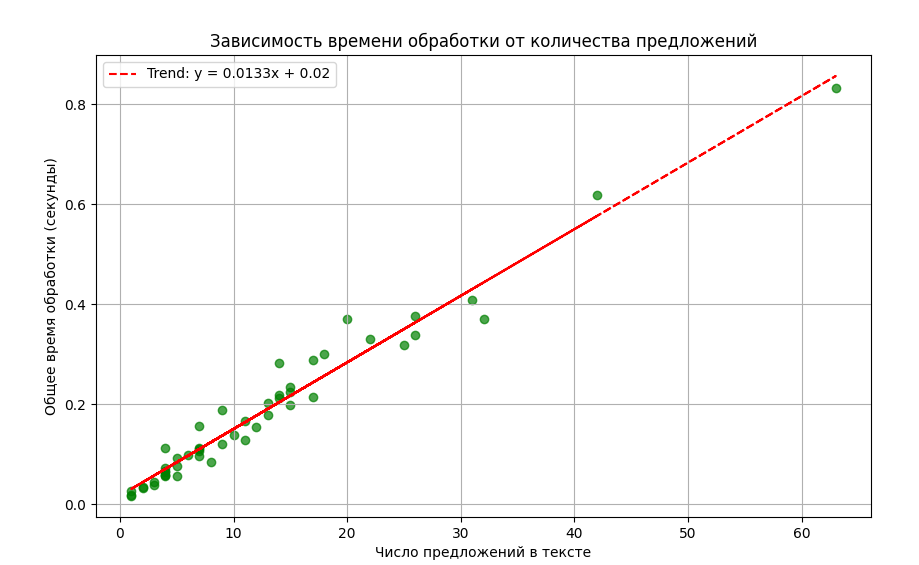
1. **Библиотека beautifulsoup4**

Библиотека упрощает скрапинг веб-страниц. В лабораторной работе использована для извлечения семантической информации из толковых онлайн-словарей.

**Оценка производительности приложения:**

* Оценка производительности серверной части:

График зависимости времени обработки от количества предложений в тексте представлен на рисунке ниже. В обработку в данном случае входил морфологический, синтаксический и семантический разборы. Также, на графике определена линия тренда и задающая её функция. Тестирования производилось на наборе данных из 50-ти текстов с разным количеством предложений. Аппаратное обеспечение - процессор Intel Core i5-11300H 3.1 ГГц.

****

* Оценка производительности клиентской части:

Для оценки клиентской части были использованы встроенные средства разработчика фреймворка Avalonia. В частности, были получены оценки минимального, среднего и максимального времени компоновки и отрисовки интерфейса.



Был проведен стресс-тест на тексте в 19 предложений (2269 символов).

При нормальной работе приложения (последовательный разбор всех видов для произвольных предложений) время компоновки не превышает 50 мс. Время отрисовки не превышает 30 мс. Среднее время компоновки 14 мс, среднее время отрисовки - 7 мс, что обеспечивает среднюю частоту кадров равную 47 кадров в секунду.

Лучшие результаты характерны для сценариев, когда длина разбираемого текста минимальна. Это обусловлено множеством графических элементов, отвечающих за синтаксический и другие разборы предложений.

В случае разбивки текста на предложения без разбора предложений время компоновки и отрисовки не превышает средние значения, что подтверждает утверждение выше.

Отдельно стоит отметить случай переключения между несколькими текстами, когда компоновка интерфейса и создание всех его элементов производится с нуля. В таком случае время компоновки составило рекордные 680 мс.

Также важно отметить, что на приближенных к реальным ситуациях (объём текста в несколько предложений, разбор отдельных элементов) показатели близки к средним. Общая производительность удовлетворяет пользовательским ожиданиям и не вызывает дискомфорта при пользовании приложением.

**Выводы:**

В ходе выполнения лабораторных работы было спроектировано и разработано полноценное клиент-серверное приложение, представляющее собой морфологический и семантико-синтаксический анализатор предложений текста. Пользователю предоставляется функционал добавления, редактирования, удаления текста, а также реализованы виды разборов: морфологический, синтаксический и семантический. Приложение состоит из двух составляющих - клиент и сервер - которые реализованы на C# и Python соответственно, с использованием фреймворков и дополнительных модулей. Приложение имеет перспективы развития, заключающиеся во внедрении в комплексную языковую систему, или существование самостоятельно как отдельный как веб-сервис.