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 №4

“Семантико-синтаксический анализ текстов естественного языка”

Выполнили студенты группы 221702: Целуйко Д.А.

Гринь Н.А.

Проверил: Крапивин Ю.Б.

Минск 2024

Лабораторная работа №4

**Цель работы:**

Освоить принципы разработки прикладных сервисных

программ для решения задачи автоматического семантико-синтаксического

анализа текста естественного языка.

**Задачи лабораторной работы (Вариант 12):**

1. Познакомиться с назначением, структурой и функциональностью, предоставляемой базовым ЛП для решения задачи автоматического синтаксического анализа ТЕЯ.

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

**Язык текста:**

Английский

**Формат входного документа:**

HTML

**Используемые средства разработки:**

Для решения поставленной задачи было разработано десктоп-приложение на языке программирования **Python** с использованием библиотеки **Tkinter**, а также некоторые другие библиотеки

**Tkinter —** это пакет для Python, предназначенный для работы с библиотекой Tk. Библиотека Tk содержит компоненты графического интерфейса пользователя (graphical user interface – GUI). Эта библиотека написана на языке программирования Tcl.

**Spacy (spaCy)** — это программная библиотека с открытым исходным кодом для расширенной обработки естественного языка, написанная на языках программирования Python и Cython. Были догружены следующие ресурсы:

* **spacy.load('en\_core\_web\_sm') -** легкая английская языковая модель, оптимизированная для использования центральным процессором. Эта модель позволяет выполнять присвоение частей речи, синтаксический анализ зависимостей и лемматизацию токенов во входном тексте. Разработано компанией Explosion AI.

**NLTK (Natural Language Toolkit) -** пакет библиотек и программ для символьной и статистической обработки естественного языка, написанных на языке программирования Python.

Были догружены следующие ресурсы:

* **wordnet -** это основной компонент, используемый для получения определений, синонимов и антонимов, которые требовалось добавить в новые столбцы таблицы.
* **punkt** - Хотя конкретно это приложение полагается на spaCy для большей части токенизации и разделения предложений, многие функции NLTK (включая некоторые потенциальные взаимодействия с WordNet или анализ его определений) зависят от punkt. Его загрузка часто необходима для корректной работы NLTK, даже если он не вызывается напрямую в нашем текущем коде.
* **averaged\_percepetron\_tagger** **-** Поиск в WordNet сильно зависит от части речи слова (например, "bank" как существительное и "bank" как глагол — это разные значения). Хотя мы в основном используем POS-теги spaCy и преобразуем их в теги WordNet (\_map\_spacy\_pos\_to\_wordnet), наличие собственного теггера NLTK гарантирует, что NLTK сможет выполнить разметку, если это потребуется для внутренних операций. Это стандартная практика при использовании интерфейса WordNet в NLTK.
* **omw-1.4 -** Даже при использовании в основном английского WordNet через NLTK, omw-1.4 часто предоставляет обновленные соответствия и обеспечивает лучшую совместимость и доступ к связанным структурам данных внутри модуля wordnet NLTK. Обычно рекомендуется загружать его вместе с wordnet для достижения наилучших результатов.

**Json** — используется в качестве стандартного текстового формата для импорта/экспорта информации об анализируемых словах и их морфологических и синтаксических данных.

Система работает со следующими частями речи:

1. **VERB**: глагол
2. **PRON**: местоимение
3. **ADJ**: прилагательное
4. **NOUN**: существительное
5. **ADV**: наречие
6. **ADP**: предлог
7. **DET**: артикль/детерминатив
8. **PROPN**: имя собственное
9. **CCONJ**: сочинительный союз
10. **AUX**: вспомогательный глагол
11. **SCONJ**: подчинительный союз
12. **NUM**: числительное
13. **PART**: частица
14. **INTJ**: междометие

Система учитывает следующие синтаксические отношения между членами предложения:  
1**. ROOT (Корень)**: Основной элемент предложения, от которого зависят другие слова.

* + Пример: "The quick brown fox **jumps**." (Здесь "jumps" — корень.)

1. **nsubj (Подлежащее)**: Основное подлежащее предложения.
   * Пример: "**The cat** is sleeping." (Здесь "The cat" — подлежащее.)
2. **dobj (Прямое дополнение)**: Объект, на который непосредственно воздействует глагол.
   * Пример: "She ate **the apple**." (Здесь "the apple" — прямое дополнение.)
3. **iobj (Косвенное дополнение)**: Объект, который получает действие, но не является прямым объектом.
   * Пример: "He gave **her** a book." (Здесь "her" — косвенное дополнение.)
4. **pobj (Объект предлога)**: Объект, управляемый предлогом.
   * Пример: "She sat **on the chair**." (Здесь "the chair" — объект предлога "on".)
5. **subj (Именная часть)**: Подлежащее в именной части предложения.
   * Пример: "**The book** on the table is interesting." (Здесь "The book" — именная часть.)
6. **csubj (Подлежащее придаточного предложения)**: Подлежащее в придаточном предложении.
   * Пример: "I think **she is smart**." (Здесь "she" — подлежащее придаточного предложения.)
7. **attr (Атрибут)**: Атрибут или определение, относящийся к подлежащему.
   * Пример: "The book is **interesting**." (Здесь "interesting" — атрибут.)
8. **agent (Агент)**: Выполняет действие в пассивной конструкции.
   * Пример: "The cake was eaten **by her**." (Здесь "her" — агент.)
9. **advcl (Наречное дополнение)**: Наречное дополнение, модифицирующее глагол.
   * Пример: "He left **because he was tired**." (Здесь "because he was tired" — наречное дополнение.)
10. **advmod (Наречие)**: Наречие, модифицирующее глагол, прилагательное или другое наречие.
    * Пример: "She runs **quickly**." (Здесь "quickly" — наречие.)
11. **amod (Определение)**: Прилагательное, модифицирующее существительное.
    * Пример: "The **big** dog." (Здесь "big" — определение.)
12. **appos (Приложение)**: Приложение, добавляющее информацию к существительному.
    * Пример: "My friend, **a doctor**, lives nearby." (Здесь "a doctor" — приложение.)
13. **aux (Вспомогательный глагол)**: Вспомогательный глагол, помогающий основному глаголу.
    * Пример: "She **is** eating." (Здесь "is" — вспомогательный глагол.)
14. **auxpass (Вспомогательный глагол в пассиве)**: Вспомогательный глагол в пассивной конструкции.
    * Пример: "The cake **was** eaten." (Здесь "was" — вспомогательный глагол в пассиве.)
15. **cc (Сочинительный союз)**: Союз, соединяющий элементы на одном уровне.
    * Пример: "apples **and** oranges." (Здесь "and" — сочинительный союз.)
16. **ccomp (Дополнение придаточного предложения)**: Придаточное предложение, выполняющее роль дополнения.
    * Пример: "I know **that she is smart**." (Здесь "that she is smart" — дополнение придаточного предложения.)
17. **compound (Сложное слово)**: Часть составного слова.
    * Пример: "coffee **cup**." (Здесь "cup" — часть сложного слова.)
18. **conj (Соединительный элемент)**: Элемент, соединенный с другим элементом.
    * Пример: "I like apples **and** oranges." (Здесь "oranges" — соединительный элемент.)
19. **dative (Дательный падеж)**: Косвенное дополнение, получающее действие.
    * Пример: "She gave **him** a book." (Здесь "him" — дательный падеж.)
20. **dep (Неопределенная зависимость)**: Неопределенная зависимость, не подходящая под другие категории.
    * Пример: Может использоваться в сложных конструкциях, где зависимость не ясна.
21. **det (Определитель)**: Определитель, уточняющий существительное.
    * Пример: "**The** book." (Здесь "The" — определитель.)
22. **expl (Вводное слово)**: Вводное слово, не несущее смысловой нагрузки.
    * Пример: "**There** is a book." (Здесь "There" — вводное слово.)
23. **intj (Междометие)**: Междометие, выражающее эмоцию.
    * Пример: "**Oh**, I see." (Здесь "Oh" — междометие.)
24. **mark (Маркер)**: Маркер, вводящий придаточное предложение.
    * Пример: "I think **that** she is smart." (Здесь "that" — маркер.)
25. **meta (Мета-модификатор)**: Модификатор, изменяющий структуру предложения.
    * Пример: "She is **not** coming." (Здесь "not" — мета-модификатор.)
26. **neg (Отрицание)**: Отрицание, изменяющее смысл предложения.
    * Пример: "She is **not** coming." (Здесь "not" — отрицание.)
27. **nmod (Именное дополнение)**: Дополнение, модифицирующее существительное.
    * Пример: "The book **on the table**." (Здесь "on the table" — именное дополнение.)
28. **npadvmod (Именная группа как наречие)**: Именная группа, выполняющая роль наречия.
    * Пример: "She left **this morning**." (Здесь "this morning" — именная группа как наречие.)
29. **nummod (Числительное)**: Числительное, модифицирующее существительное.
    * Пример: "**Three** books." (Здесь "Three" — числительное.)
30. **oprd (Объектное дополнение)**: Объектное дополнение, связанное с подлежащим.
    * Пример: "She considers him **a friend**." (Здесь "a friend" — объектное дополнение.)
31. **parataxis (Паратаксис)**: Паратаксис, соединяющий предложения без явного союза.
    * Пример: "He came; **she left**." (Здесь "she left" — паратаксис.)
32. **pcomp (Дополнение предлога)**: Дополнение, управляемое предлогом.
    * Пример: "She is afraid **of spiders**." (Здесь "of spiders" — дополнение предлога.)
33. **poss (Притяжательное дополнение)**: Притяжательное дополнение, указывающее на владение.
    * Пример: "**Her** book." (Здесь "Her" — притяжательное дополнение.)
34. **preconj (Предшествующий союз)**: Союз, предшествующий соединенному элементу.
    * Пример: "She is **both** smart **and** kind." (Здесь "both" — предшествующий союз.)
35. **predet (Пред-определитель)**: Определитель, предшествующий основному определителю.
    * Пример: "**All the** books." (Здесь "All" — пред-определитель.)
36. **prep (Предлог)**: Предлог, управляющий объектом.
    * Пример: "She sat **on** the chair." (Здесь "on" — предлог.)
37. **prt (Частица)**: Частица, изменяющая значение глагола.
    * Пример: "She gave **up**." (Здесь "up" — частица.)
38. **punct (Пунктуация)**: Пунктуация, структурирующая предложение.
    * Пример: "Hello, world!" (Здесь "," и "!" — пунктуация.)
39. **quantmod (Квантификатор)**: Квантификатор, указывающий на количество.
    * Пример: "**Many** books." (Здесь "Many" — квантификатор.)
40. **relcl (Относительное дополнение)**: Придаточное предложение, модифицирующее существительное.
    * Пример: "The book **that she read**." (Здесь "that she read" — относительное дополнение.)
41. **xcomp (Открытое дополнение придаточного предложения)**: Придаточное предложение, выполняющее роль открытого дополнения.
    * Пример: "She started **to read**." (Здесь "to read" — открытое дополнение придаточного предложения.)

**Алгоритм Морфологического анализа:**

1. **Начало алгоритма**: Получен результат обработки текста (проанализированный документ spaCy). Начинается итерация для заполнения таблицы результатов.
2. **Шаг 1**: Взять следующий токен
3. **Шаг 2**: Для текущего токена в проанализированном документе:
4. **Шаг 2.1**: Проверить, является ли токен релевантным (не пробел, не помечен как игнорируемый вручную). Если не релевантен, перейти к следующему токену (вернуться к Шагу 1).
5. **Шаг 2.2**: Получить текст токена (словоформу).
6. **Шаг 2.3**: Получить базовую форму слова (лемму) из результатов анализа spaCy.
7. **Шаг 2.4**: Проверить, есть ли ручное переопределение для леммы этого токена. Если есть, использовать переопределенное значение вместо базового.
8. **Шаг 2.5**: Получить тег части речи из результатов анализа spaCy.
9. **Шаг 2.6**: Преобразовать тег части речи в читаемое название (если есть словарь перевода).
10. **Шаг 2.7**: Проверить, есть ли ручное переопределение для части речи этого токена. Если есть, использовать переопределенное значение вместо полученного на предыдущем шаге.
11. **Шаг 2.8**: Получить набор морфологических признаков из результатов анализа spaCy.
12. **Шаг 2.9**: Преобразовать набор морфологических признаков в одну строку (например, "Число:Ед, Падеж:Им").
13. **Шаг 2.10**: Проверить, есть ли ручное переопределение для морфологии этого токена. Если есть, использовать переопределенное значение вместо сгенерированной строки.
14. **Шаг 2.11**: Подготовить полученные данные (словоформа, лемма, часть речи, морфология) для добавления в строку таблицы.
15. **Шаг 2.12**: Перейти к выполнению **Алгоритма "Синтаксический анализ и отображение"** для этого же токена (чтобы получить данные для следующего столбца).
16. **Конец алгоритма** (для морфологической части обработки одного токена).

**Алгоритм "Синтаксический анализ и отображение"**

1. **Начало алгоритма**: Получены морфологические данные для текущего токена.
2. **Шаг 1**: Получить метку синтаксической зависимости (например, "nsubj", "dobj", "ROOT") из результатов анализа spaCy для текущего токена.
3. **Шаг 2**: Проверить, есть ли ручное переопределение для метки зависимости этого токена. Если есть, использовать переопределенное значение вместо полученного на предыдущем шаге.
4. **Шаг 3**: Подготовить полученную метку зависимости для добавления в строку таблицы.
5. **Шаг 4**: Перейти к выполнению Алгоритма "Семантический анализ и отображение" для этого же токена (чтобы получить данные для следующих столбцов).
6. **Конец алгоритма** (для синтаксической части обработки одного токена).

**Алгоритм "Семантический анализ и отображение (на основе WordNet)"**

1. **Начало алгоритма**: Получены синтаксические данные для текущего токена.
2. **Шаг 1**: Получить лемму токена (оригинальную или переопределенную, используется результат Шага 1.4 из морфологического алгоритма).
3. **Шаг 2**: Получить оригинальный тег части речи spaCy (не переопределенный и не переведенный) для текущего токена.
4. **Шаг 3**: Установить значения по умолчанию ("N/A") для синонимов, антонимов и определения.
5. **Шаг 4**: Проверить, подходит ли полученная часть речи для поиска в WordNet (например, не является ли пунктуацией, символом, пробелом и т.д.). Если не подходит, перейти на Шаг 13.
6. **Шаг 5**: Попытаться преобразовать тег части речи spaCy в соответствующий тег WordNet (например, NOUN, VERB, ADJ, ADV).
7. **Шаг 6**: Если преобразование не удалось (тег не соответствует основным типам WordNet), перейти на Шаг 13.
8. **Шаг 7**: Попытаться найти наборы синонимов (синсеты) в WordNet для полученной леммы и преобразованного тега части речи.
9. **Шаг 8**: Если синсеты не найдены, перейти на Шаг 13.
10. **Шаг 9**: Взять первый (наиболее частый) синсет из найденных.
11. **Шаг 10**: Извлечь определение из этого синсета. Если определения нет, использовать "N/A". Сохранить как результат для определения.
12. **Шаг 11**: Извлечь несколько первых лемм (слов) из этого синсета. Сформировать строку синонимов (через запятую), исключив саму исходную лемму. Если синонимы не найдены, использовать "N/A". Сохранить как результат для синонимов.
13. **Шаг 12**: Попытаться найти антонимы для первой леммы этого синсета. Сформировать строку антонимов (через запятую), взяв несколько первых. Если антонимы не найдены, использовать "N/A". Сохранить как результат для антонимов.
14. **Шаг 13**: Подготовить полученные данные (синонимы, антонимы, определение) для добавления в строку таблицы.
15. **Шаг 14**: Добавить все собранные данные (морфологические, синтаксические, семантические) для текущего токена как новую строку в таблицу результатов.
16. **Шаг 15**: Сохранить связь между ID добавленной строки таблицы и индексом обработанного токена.
17. **Шаг 16**: Перейти к следующему токену в исходном документе (вернуться к Шагу 1 в морфологическом алгоритме). Если токены закончились, конец алгоритма (для всего процесса заполнения таблицы).
18. **Конец алгоритма** (для семантической части обработки одного токена).

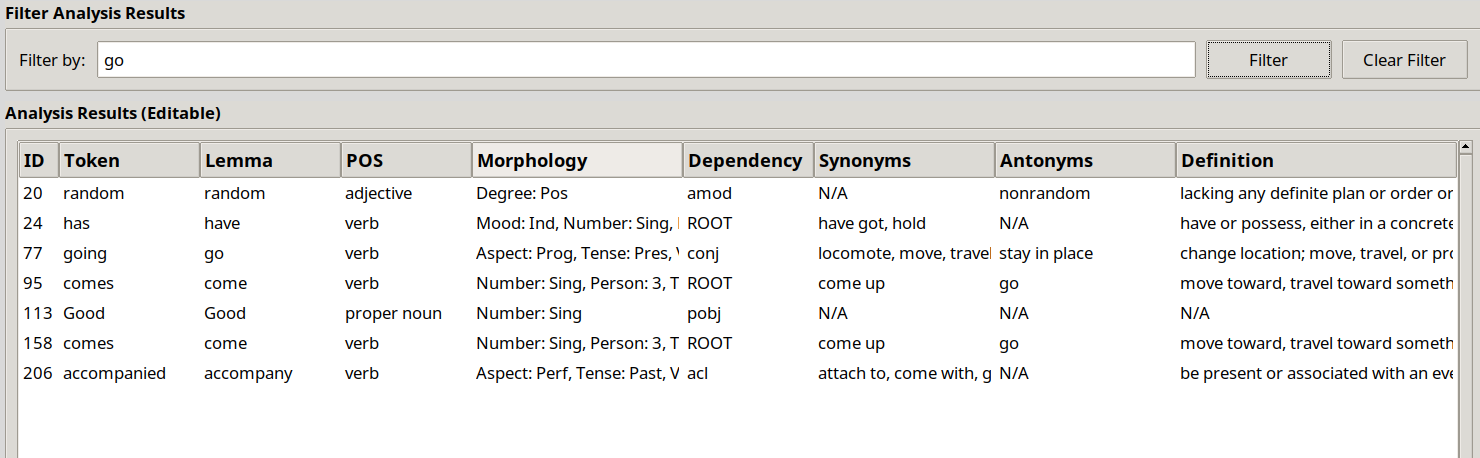
**Скриншоты разработанной системы:**



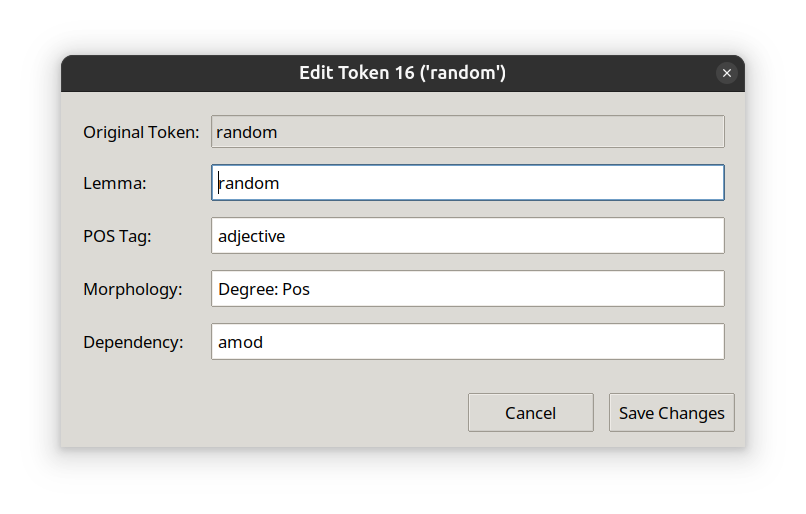
*Рис. 1- начальное окно приложения*

****

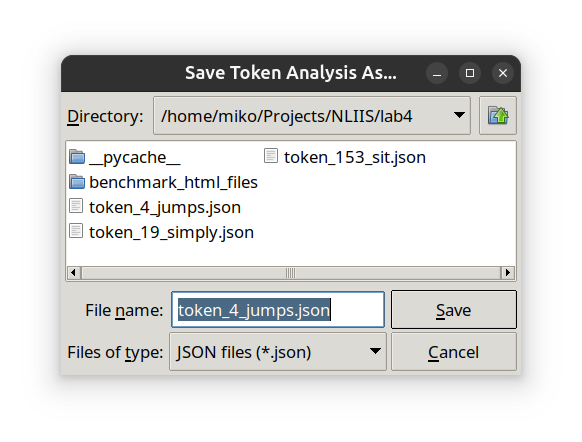
*Рис. 2 – результат анализа файла*

**

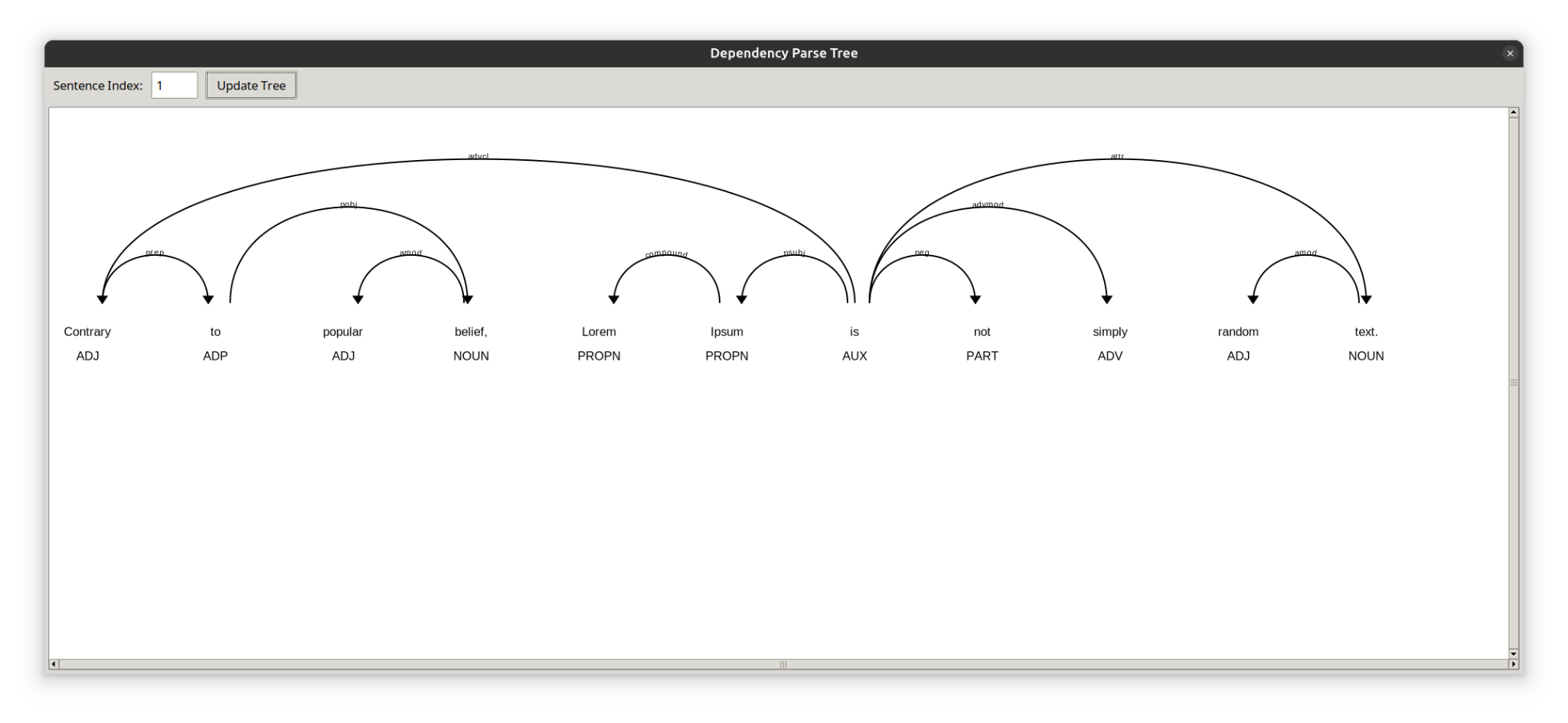
*Рис. 2 – фильтрация (поиск) по слову “go”*

****

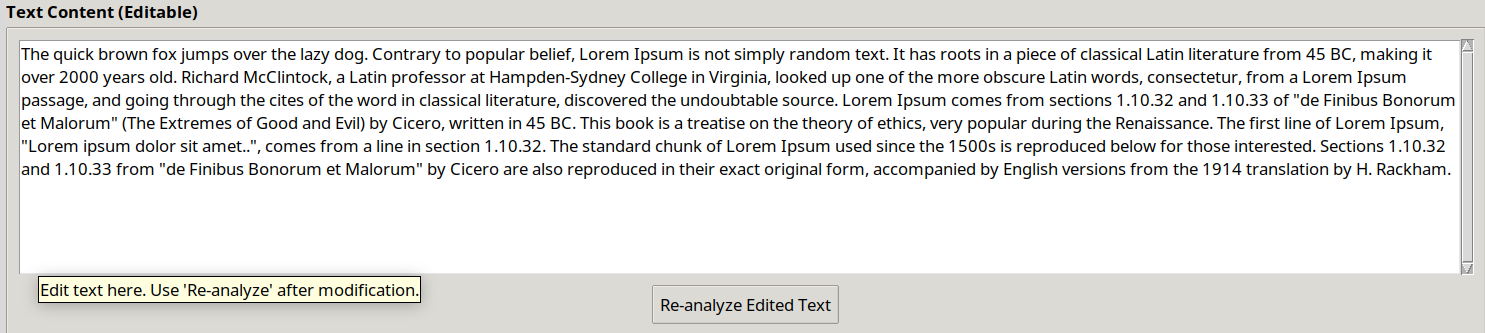
*Рис. 3 – Окно редактирования слова*

****

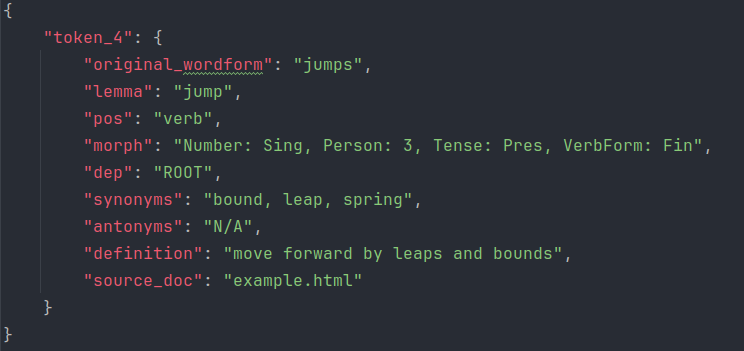
*Рис. 4 – страница экспорта слова “jumps” в json файл*



*Рис. 5 – синтаксическое дерево предложения “The quick brown fox jumps over the lazy dog.”*

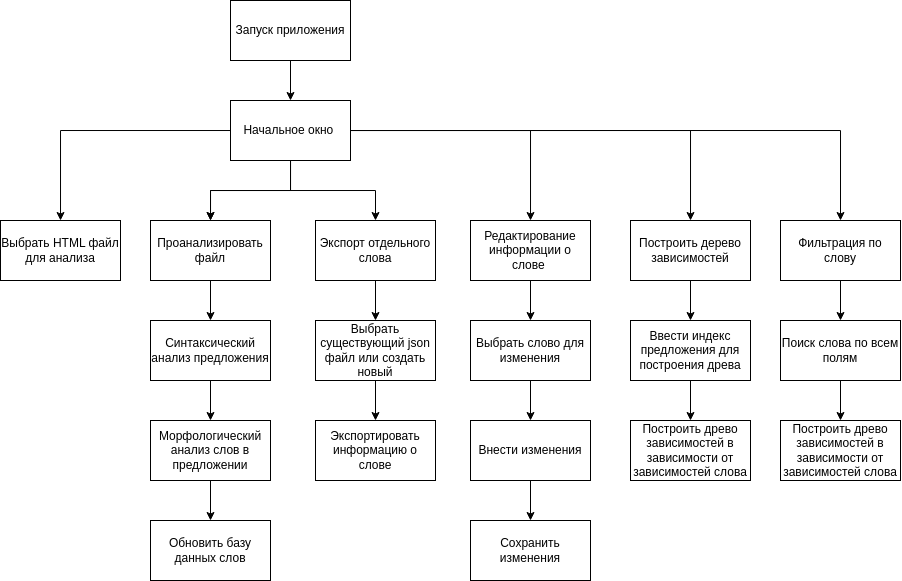
****

*Рис. 6 – пример окна с помощью к таблице*

**

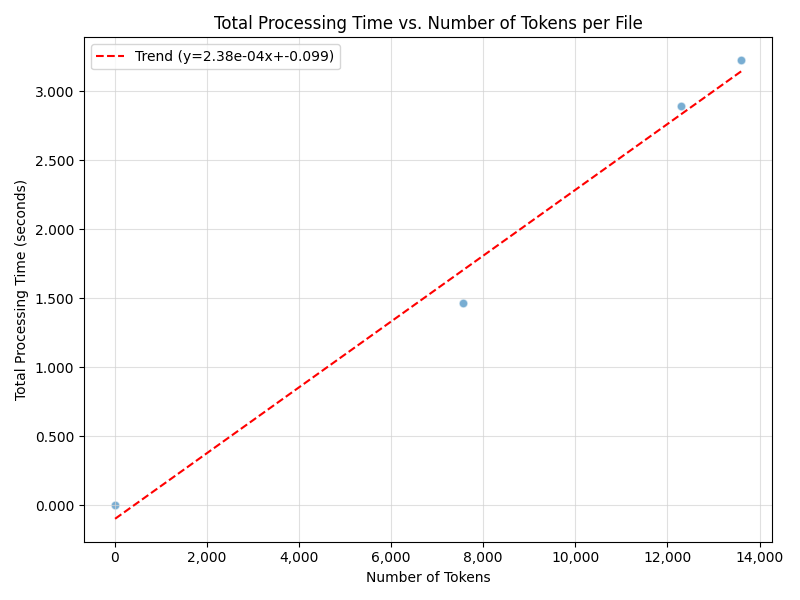
*Рис. 7 – Содержимое файла new.json с экспортированным словом jumps*

**Структурно-функциональная схема разрабатываемого приложения:**

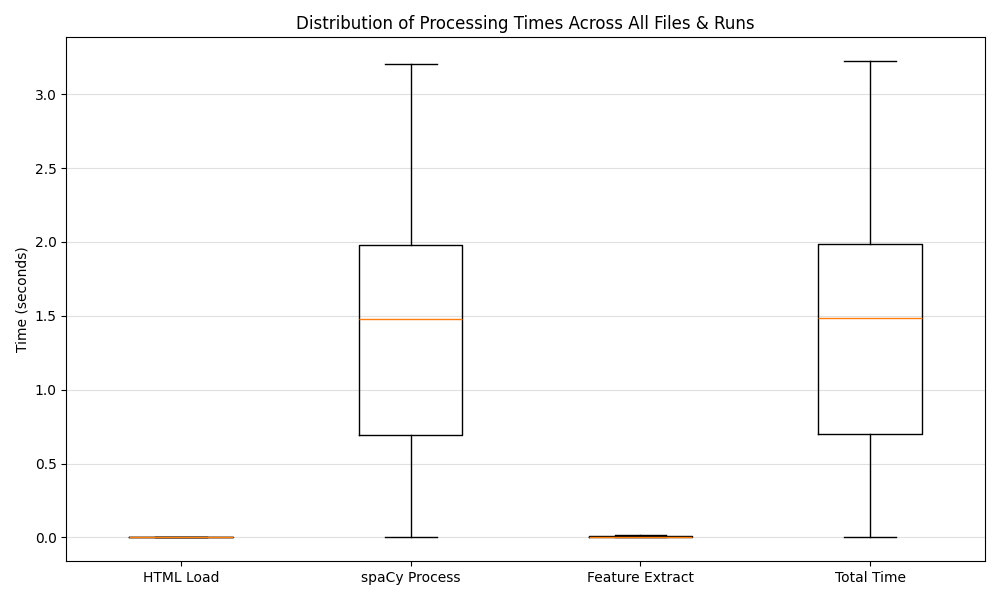


*Рис. 8 - структурно-функциональная схема приложения*

**График оценки быстродействия приложения:**



*Рис. 9 - график быстродействия обработки токенов. Зависимость времени от количества токенов*

**

*Рис. 10 - график распределения времени на работу компонентов*

**Выводы по работе и по перспективам развития приложения:**

Разработанное приложение представляет собой систему, которая позволяет проводить синтаксический анализ каждого предложения заданного текста, а также построить синтаксическое дерево для каждого предложения. В рамках перспективы развития приложения можно выделить возможность сделать приложение многопользовательским.