Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра интеллектуальных информационных технологий

Отчёт
по курсу «**Естественно-языковой интерфейс интеллектуальных систем**»

Лабораторная работа №3
«Семантико-синтаксический анализ текстов естественного языка»

Выполнили студенты группы 121701:	Чвилёв И.А. Воронцов Р.Г. Силибин С.
Проверил:	Крапивин Ю.Б.

Минск 2024

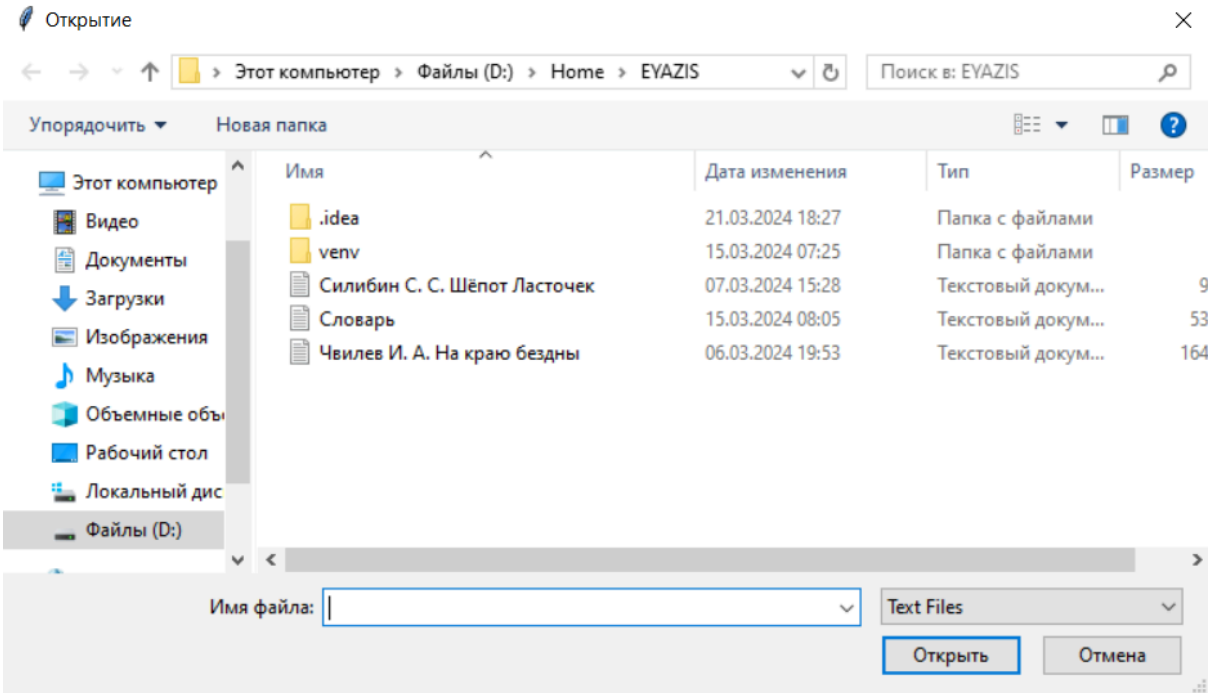
Цель работы: Освоить принципы разработки прикладных сервисных программ для решения задачи автоматического семантико-синтаксического анализа текста естественного языка.

Задание:

1. Познакомиться с назначением, структурой и функциональностью, предоставляемой базовым ЛП для решения задачи автоматического семантико-синтаксического анализа ТЕЯ.
2. Закрепить навыки программирования при решении задач автоматической обработки ТЕЯ.

Алгоритм работы программы:

1) Открывается диалог выбора файла на диске
   а) Пользователь выбирает текстовый файл, переход к шагу 2

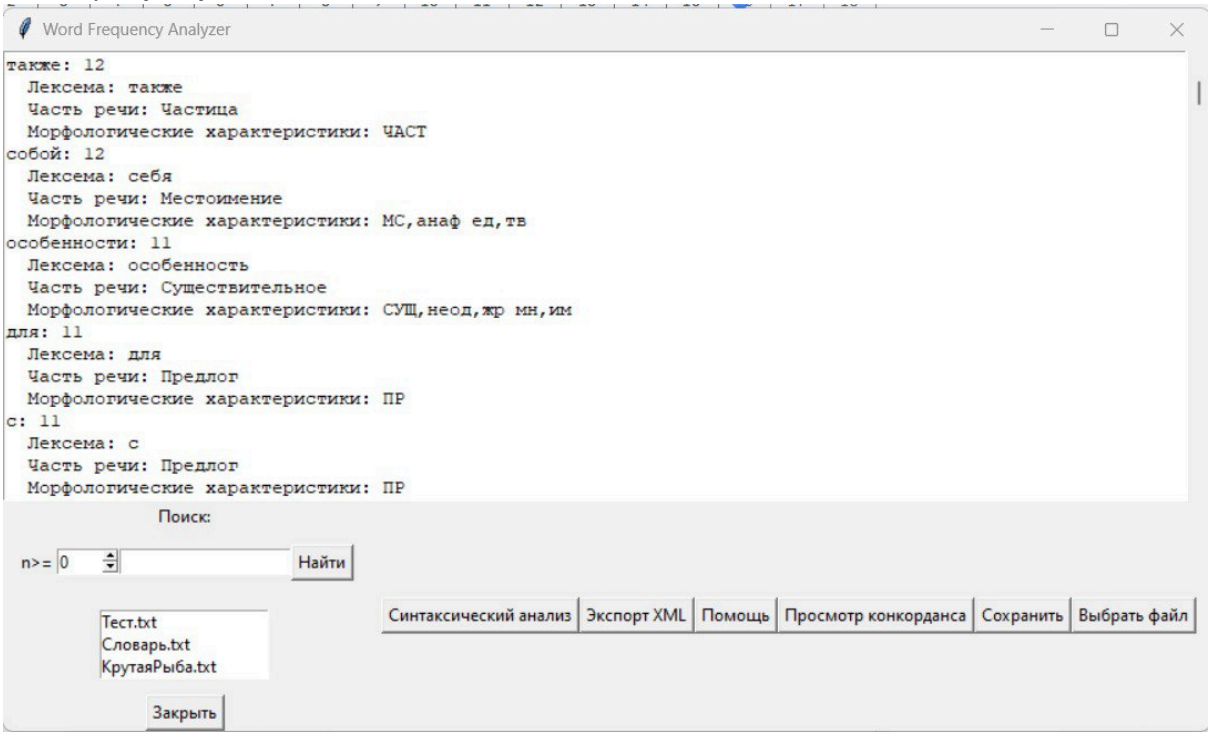


2) Программа проходит по всему тексту, получает все слова из файла, подсчитывает их частоту, производит морфологический анализ и создает описание для каждого слова.

```
def parse_text(self, file_paths):
    words = []
    for file_path in file_paths:
        with open(file_path, "r", encoding="utf-8") as file:
            content = file.read()
            file_words = re.findall(r"\w+", content.lower())
            words.extend(file_words)

    self.word_freq = Counter(words)

    for word in self.word_freq:
        parsed_word = self.morph.parse(word)[0]
        word_desc = {
            "wordform": word,
            "lexeme": parsed_word.normal_form,
            "pos": convert_tags_to_russian(parsed_word.tag.POS),
            "morphological_properties": parsed_word.tag.cyr_repr,
        }
        self.word_desc[word] = word_desc

    self.word_freq = dict(sorted(self.word_freq.items(), key=lambda item: item[1], reverse=True))
    self.update_text_box()
```

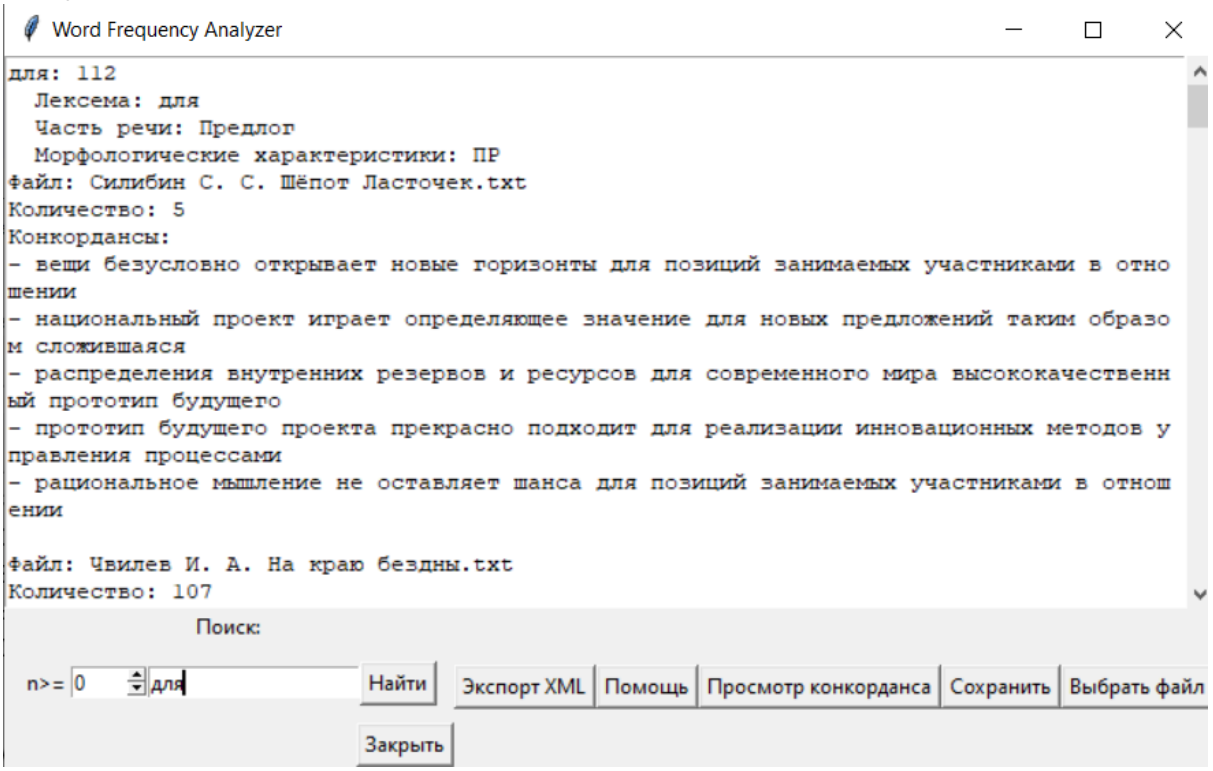
3) Программа формирует основное окно интерфейса и выводит сформированный словарь. Формируются поля для поиска, фильтрации по длине слова, просмотра конкорданса, синтаксического анализа, выбора файла (файлов), завершения работы, вкладка помощи, сохранения текста, экспорта.

3.1) Программа ожидает действия пользователя.

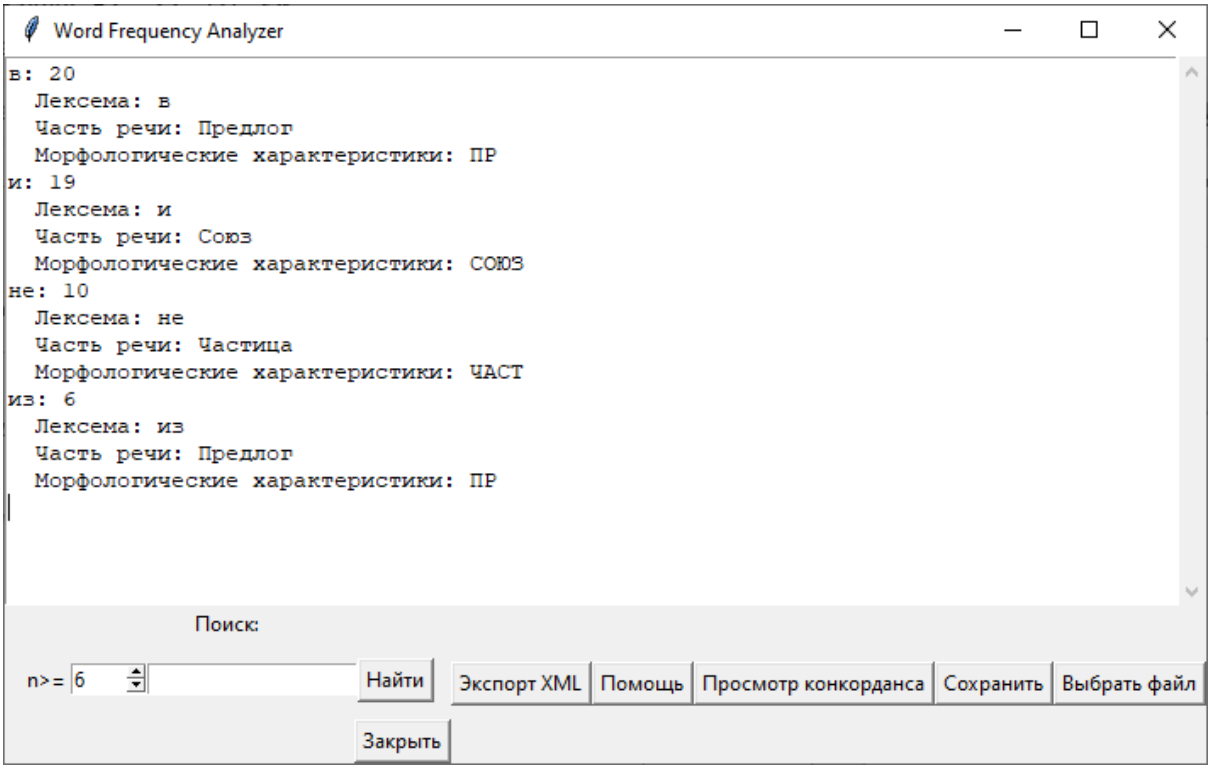


а) Пользователь вводит запрос в строку поиска и нажимает “Найти”. Программа переходит к шагу 4.
б) Пользователь выбирает минимальное значение n, введя число в поле слева от строки поиска и нажимает “Найти”. Программа переходит к шагу 5.
г) Пользователь нажимает “Сохранить”. Программа переходит к шагу 6.
д) Пользователь нажимает на кнопку “Помощь”. Программа переходит к шагу 7.
е) Пользователь нажимает на кнопку “Экспорт XML”. Программа переходит к шагу 8.
ж) Пользователь нажимает на кнопку “Синтаксический анализ”. Программа переходит к шагу 9.
з) Пользователь закрывает программу. Программа переходит к шагу 10.

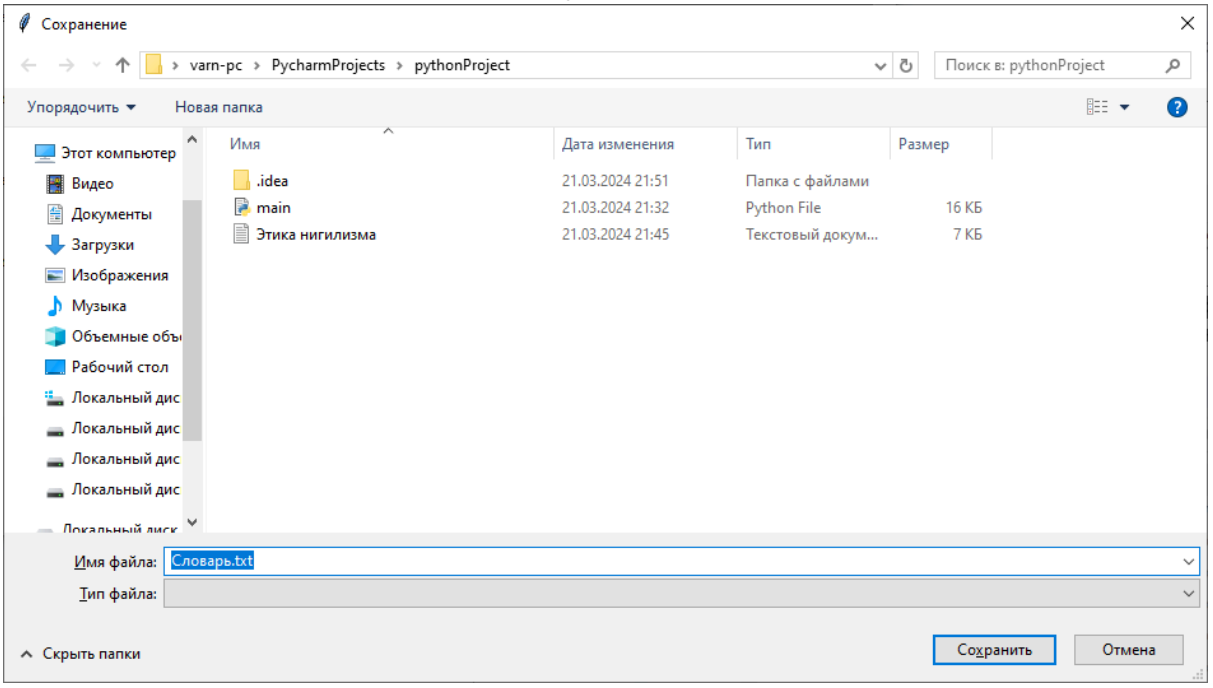
4) Список выведенных слов обновляется, включая только те, которые содержат значение поиска. Программа возвращается к шагу 3.1.



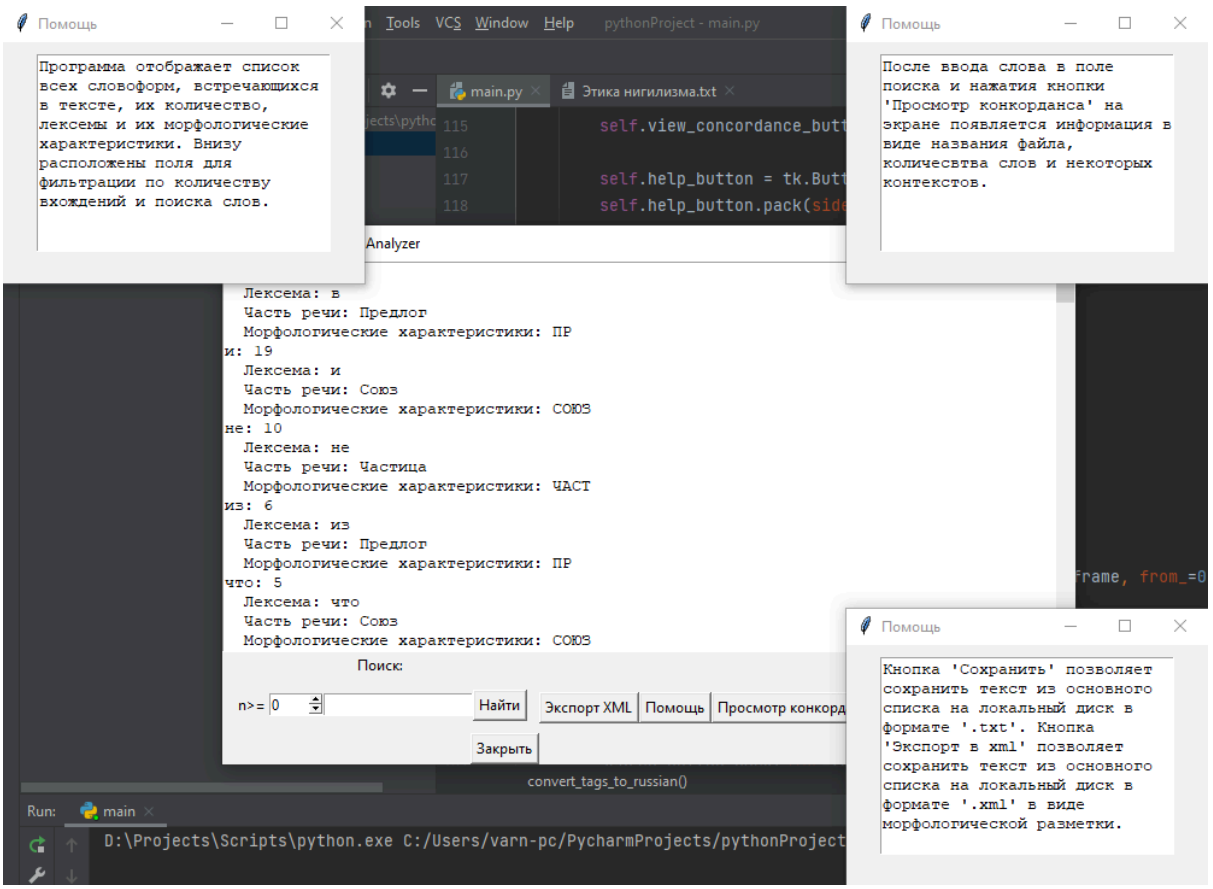
5) Список выведенных слов обновляется, включая только те, число которых больше или равно значению в поле “n>=”. Программа возвращается к шагу 3.1



6) Программа предлагает пользователю выбрать имя файла и расположение на диске, куда этот файл будет сохранён. Затем сохраняет и возвращается к шагу 3.1



7) Открываются 3 окна-подсказки, объясняющие пользователю что, где и для чего расположено. Закрыть их можно повторным нажатием на “Помощь”. Программа возвращается к шагу 3.1

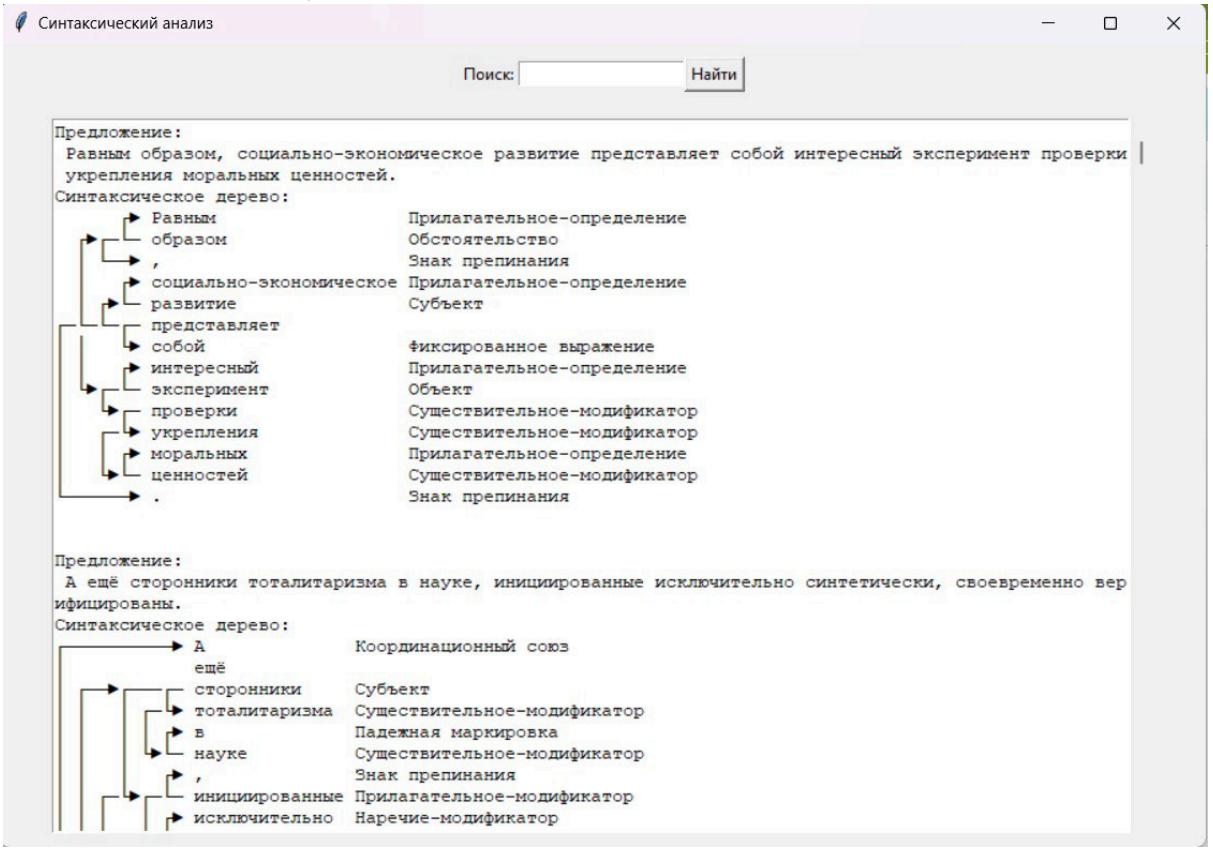


8) Результаты работы программы экспортируются в файл формата XML.

```xml
<?xml version='1.0' encoding='windows-1251'?>
<text>
<w>банальные<ana lemma="банальный" pos="Прилагательное (полное)" gram="ПРИЛ,кач мн,им" />
</w>
<w>но<ana lemma="но" pos="Союз" gram="СОЮЗ" />
</w>
<w>неопровержимые<ana lemma="неопровержимый" pos="Прилагательное (полное)" gram="ПРИЛ,кач мн,им" />
</w>
<w>выводы<ana lemma="вывод" pos="Существительное" gram="СУЩ,неод,мр мн,вн" />
</w>
<w>а<ana lemma="а" pos="Союз" gram="СОЮЗ" />
</w>
<w>также<ana lemma="также" pos="Частица" gram="ЧАСТ" />
</w>
<w>акционеры<ana lemma="акционер" pos="Существительное" gram="СУЩ,од,мр мн,им" />
</w>
<w>крупнейших<ana lemma="крупный" pos="Прилагательное (полное)" gram="ПРИЛ,превосх,кач мн,рд" />
</w>
```

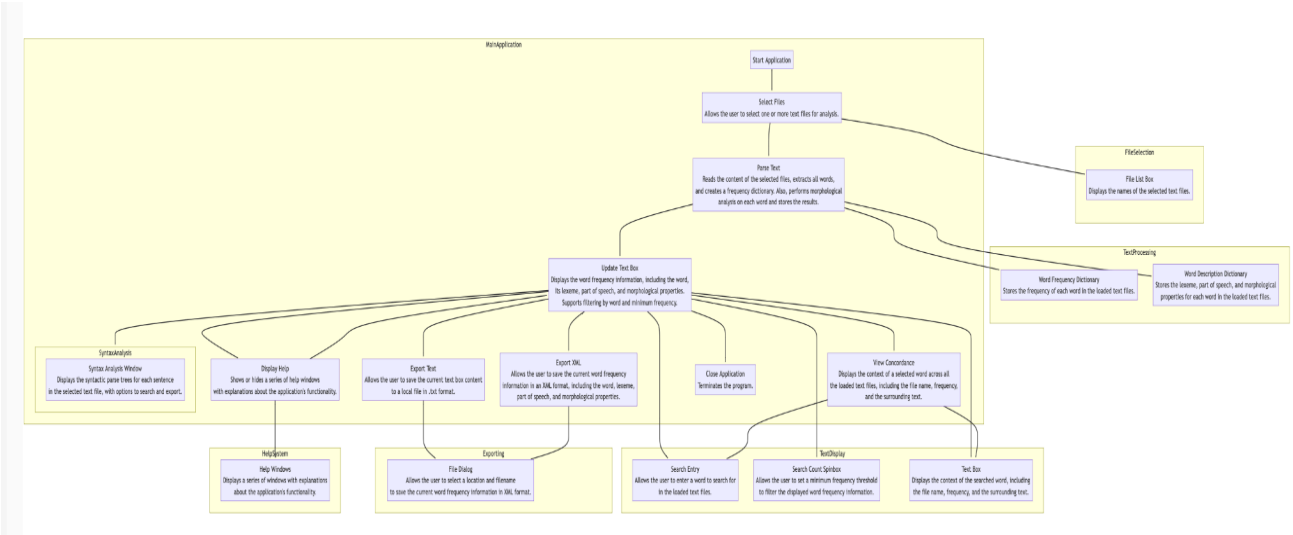
9) Программа формирует окно с синтаксическим анализом. Здесь же можно вводить запрос в строку поиска в соответствующее поле,

и, нажав на кнопку “Найти”, синтаксический анализ обновится.



10) Программа завершает работу.

Структурно-функциональная схема приложения:



Использованные структуры хранения данных:

После получения текста из выбранного файла программа инициализирует объект типа Doc, предоставленный библиотекой для синтаксического анализа русского языка Natasha.

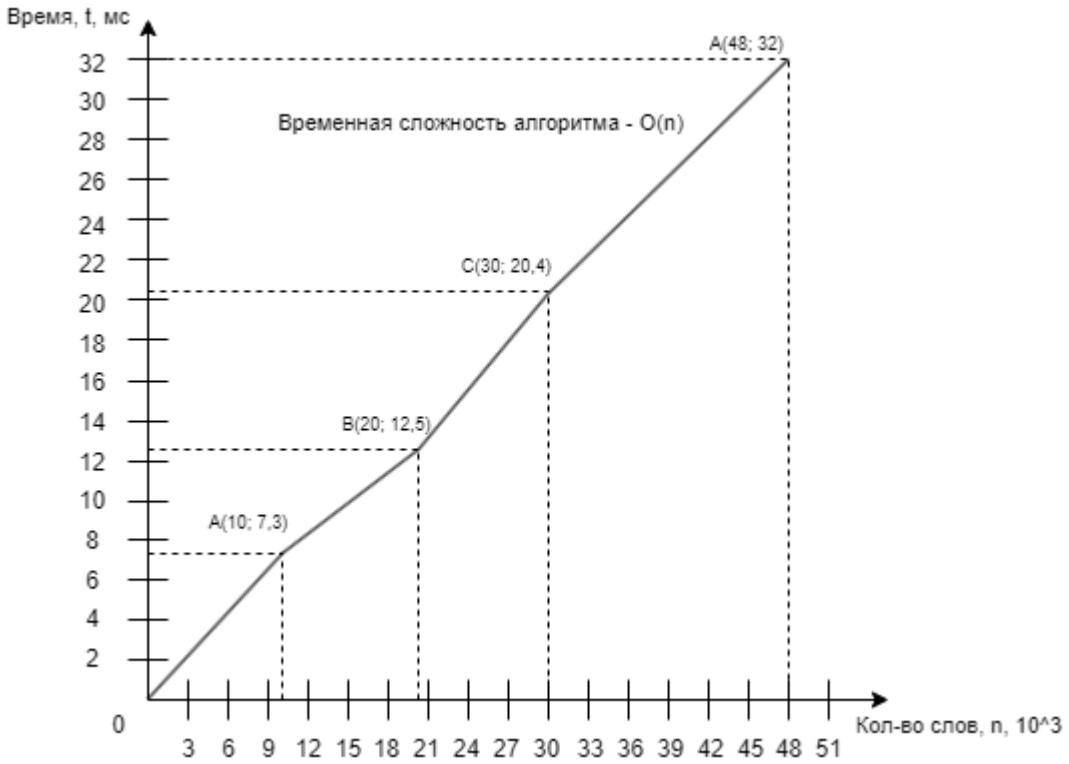
Doc - базовый класс, используемый для последующей обработки в библиотеке.
После инициализации проводим сегментацию с помощью метода segment, для которого предоставляем объект типа Segmenter. Segmenter разбивает текст на объекты Token, представляющие отдельные предложения.
После разбиения текста на токены можно провести синтаксический анализ с использованием NewsSyntaxParser, специального синтаксического парсера. Он инициализирует объекты syntax для каждого токена.

```
def syntax_analysis(self):
    segmenter = Segmenter()
    emb = NewsEmbedding()
    syntax_parser = NewsSyntaxParser(emb)

    selected_file = self.file_name_dict[self.file_listbox.get(tk.ACTIVE)]
    text = ''

    with open(selected_file, 'r', encoding='utf-8') as file:
        text = file.read()

    doc = Doc(text)
    doc.segment(segmenter)
    doc.parse_syntax(syntax_parser)

    self.create_syntax_window(doc)
```

Затем программа выводит синтаксические деревья для каждого предложения в новом окне. Текст предоставляется в наглядном формате библиотекой Natasha и доступен с помощью метода print() у объектов syntax.

```
for sentence in doc.sents:
    if search_text in sentence.text.lower():
        sentence.syntax.print()
        text_area.insert(tk.END, f'Предложение:\n {sentence.text}\n')
        text_area.insert(tk.END, 'Синтаксическое дерево:\n')
        text_area.insert(tk.END, f'{translate_text(output.getvalue())}\n')
        text_area.insert(tk.END, '\n')
        output.truncate(0)
        output.seek(0)
```

Оценка быстродействия приложения:



Оценка была проведена путем вывода времени программы после ее завершения. В зависимости от количества слов на график было нанесено несколько точек, а эти точки были соединены прямыми линиями.

Вывод: Были изучены принципы разработки прикладных сервисных программ для решения задачи автоматического синтаксического анализа текста естественного языка (ТЕЯ), в частности познакомились с назначением, структурой и функциональностью, предоставляемой базовым лингвистическим процессором (ЛП), а также закрепили навыки программирования при решении задач автоматической обработки текста естественного языка (ТЕЯ). Разработанное приложение может быть полезно для исследования языка, обучения языковым навыкам и оптимизации систем обработки языка.