Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра интеллектуальных информационных технологий

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

Лабораторная работа №2
«Построение и использование корпусов текстов естественного языка»

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

Минск 2024

Цели работы:

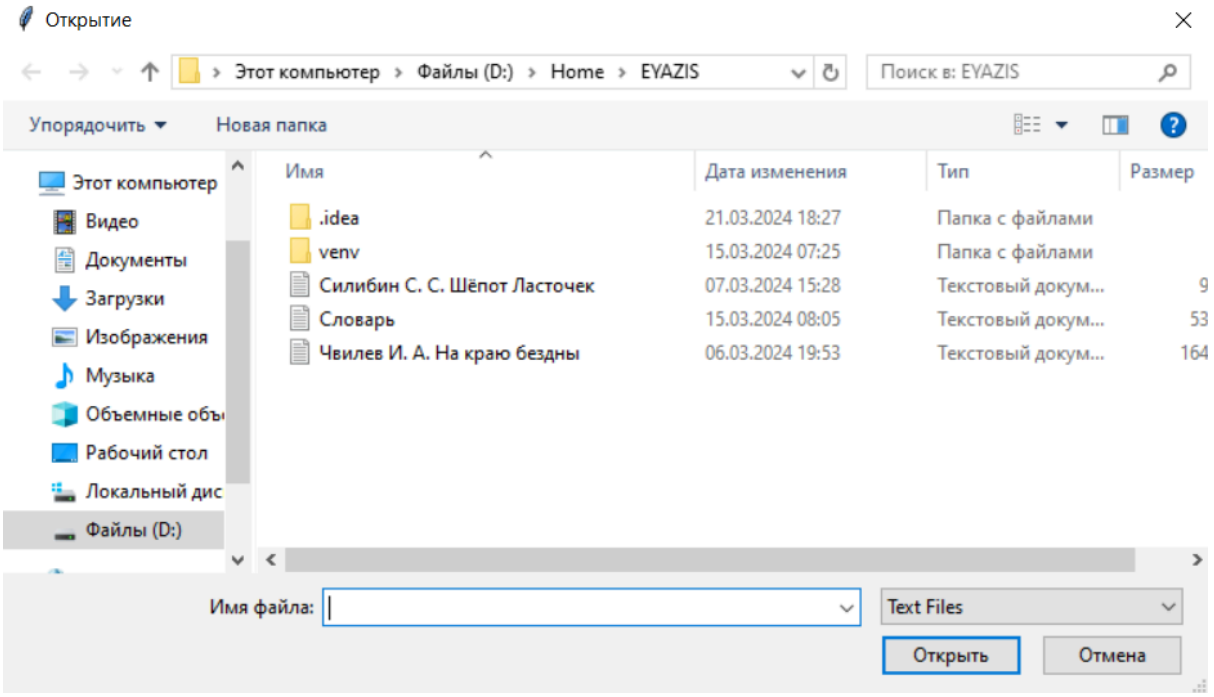
1. Изучить принципы построения корпусов текстов, виды разметки и способы аннотирования, инструменты работы с корпусами текстов
2. Построить корпус текстов и разработать корпусный менеджер.

Задание:

1. Сформировать электронный корпус текстов по выбранной предметной области.
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.
   Язык текста – русский, формат входного документа – TXT, RTF.

Алгоритм работы программы:

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

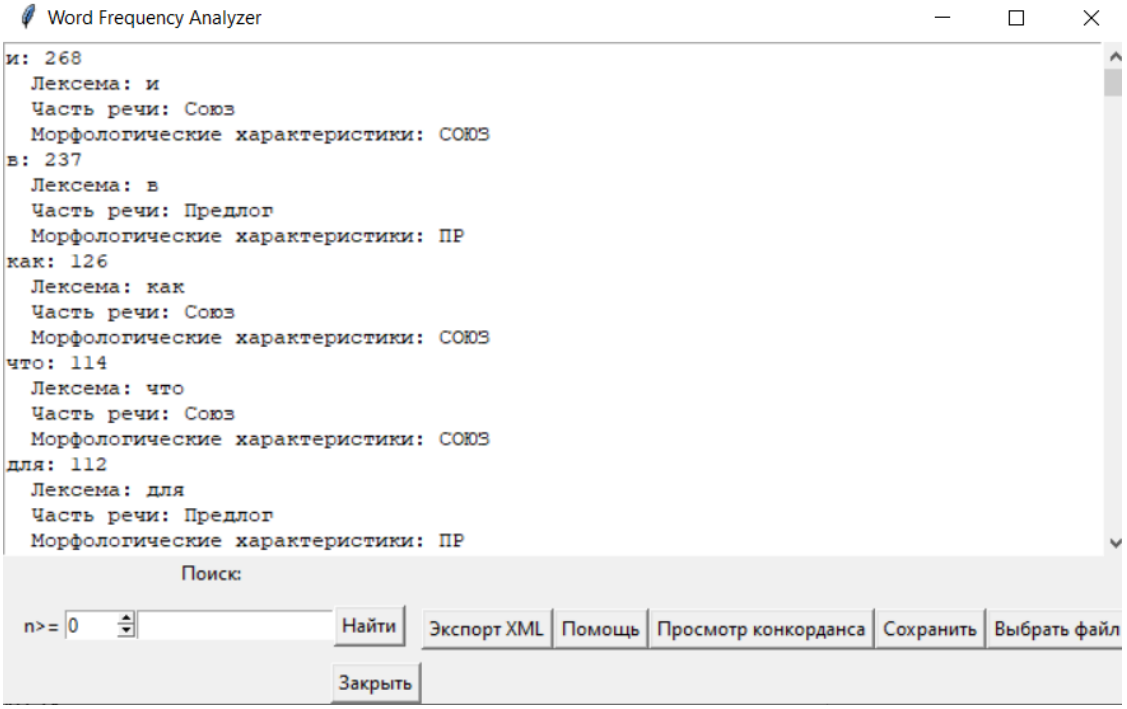


2) Программа проходит по всему тексту, получает все слова из файла, подсчитывает их частоту, производит морфологический анализ и создает описание для каждого слова.

```
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, выбора файла (файлов), завершения работы, вкладка помощи, сохранения текста, экспорта.

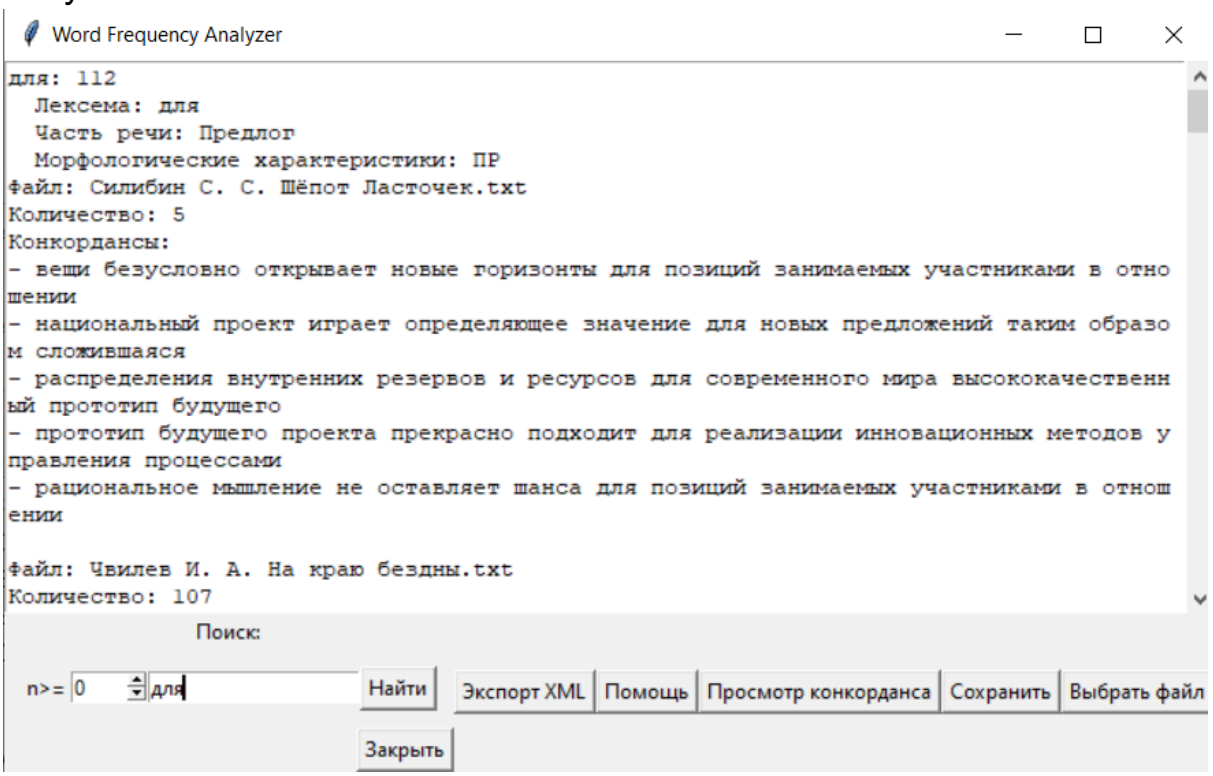
3.1) Программа ожидает действия пользователя.



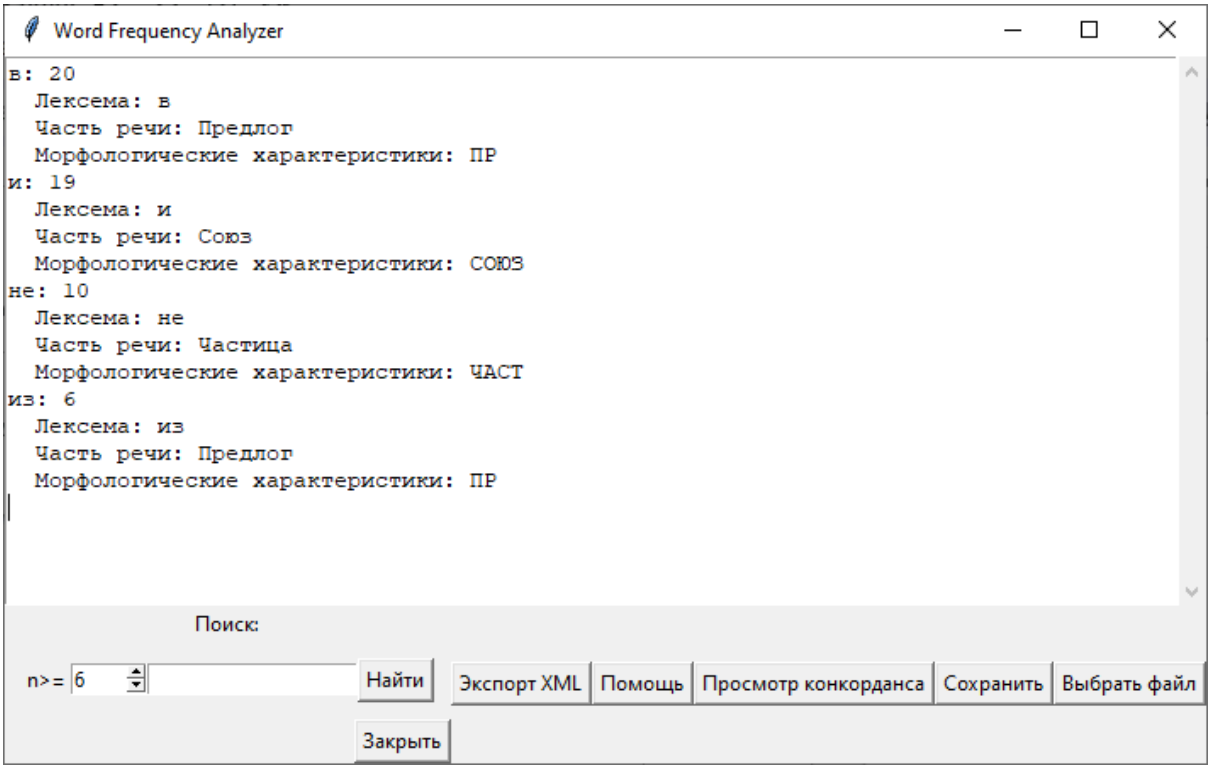
а) Пользователь вводит запрос в строку поиска и нажимает “Найти”. Программа переходит к шагу 4.

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 закрывает программу. Программа переходит к шагу 9.

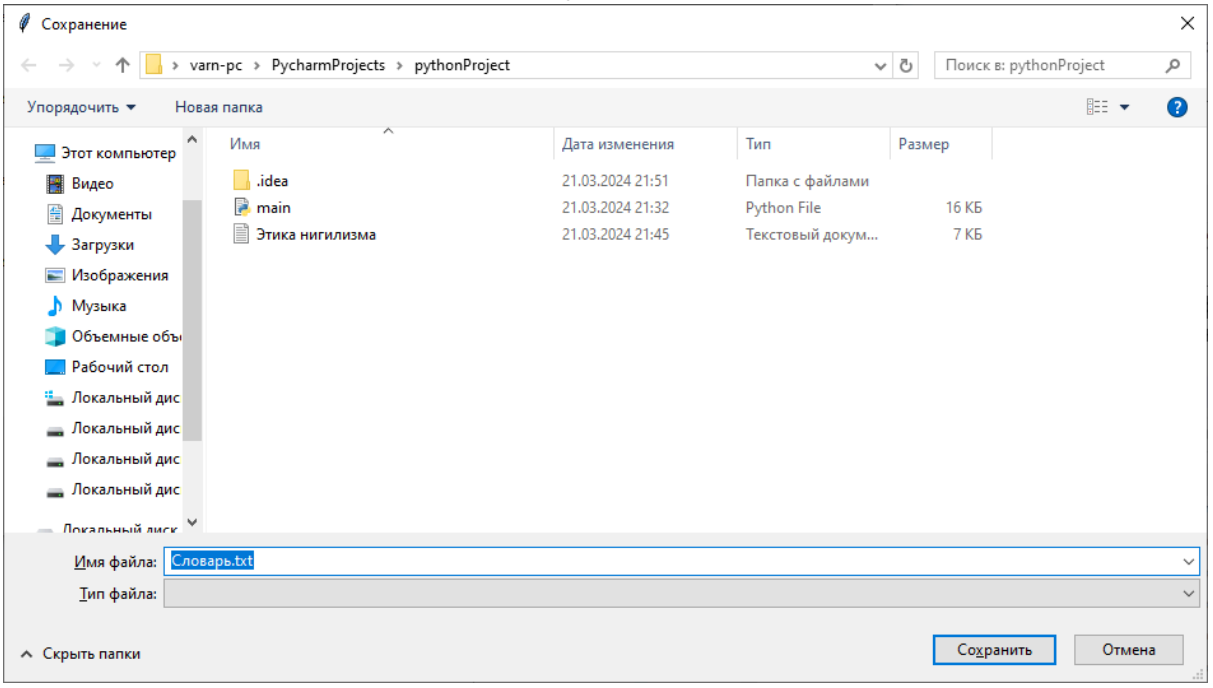
4) Список выведенных слов обновляется, включая только те, которые содержат значение поиска. Программа возвращается к шагу 3.1.



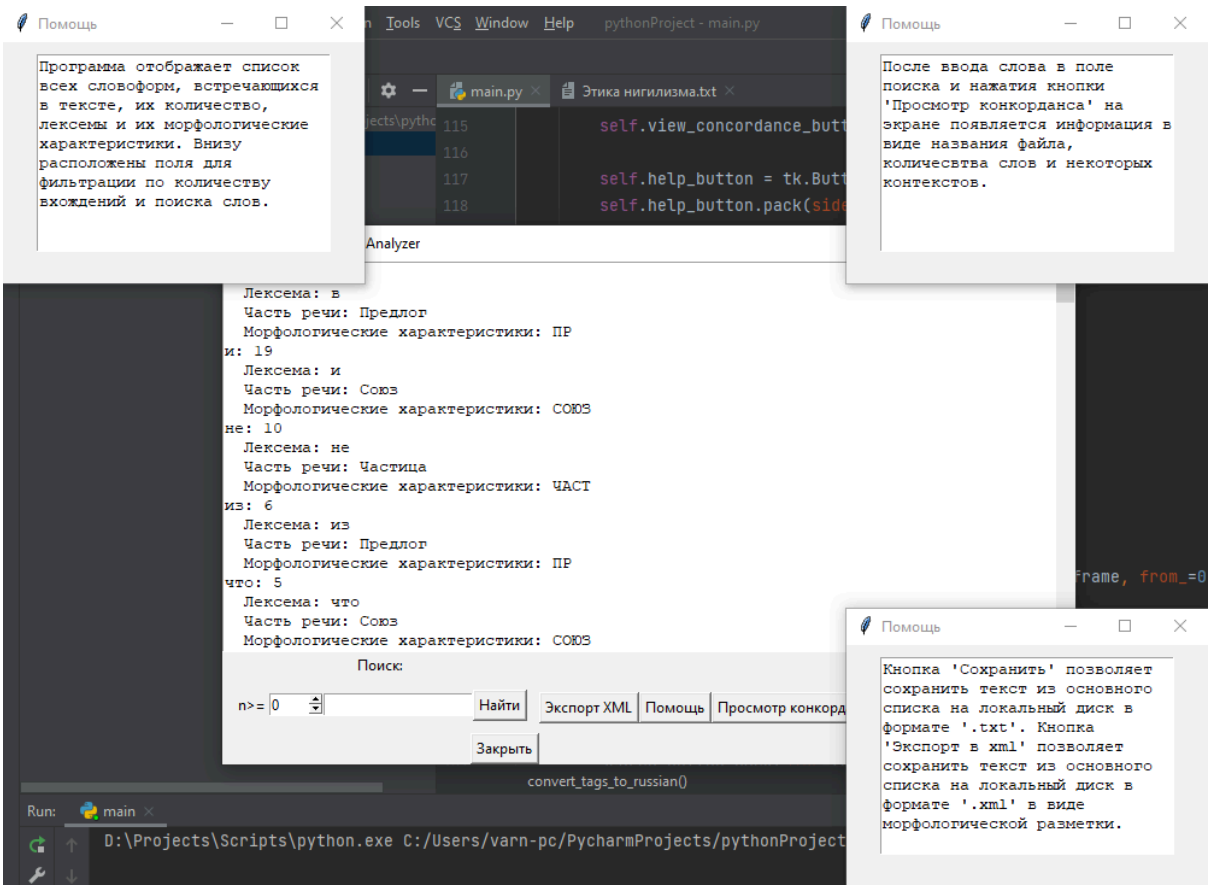
5) Список выведенных слов обновляется, включая только те, число которых больше или равно значению в поле “n>=”. Программа возвращается к шагу 3.1



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 завершает работу.

Использованные структуры хранения данных:
Хранение данных производится в самой программе на Python, в типе `defaultdict`, то есть словаре. Таких словарей 2:
word_freq:
ключ - слово из текста
значение - количество вхождений данного слова тексте
Заполняется после выбора текстового документа. Используется для отображения в основном списке.

word_desc:
ключ - слово из текста
значение - информация о слове
Заполняется после выбора текстового документа. Используется для отображения в основном списке.

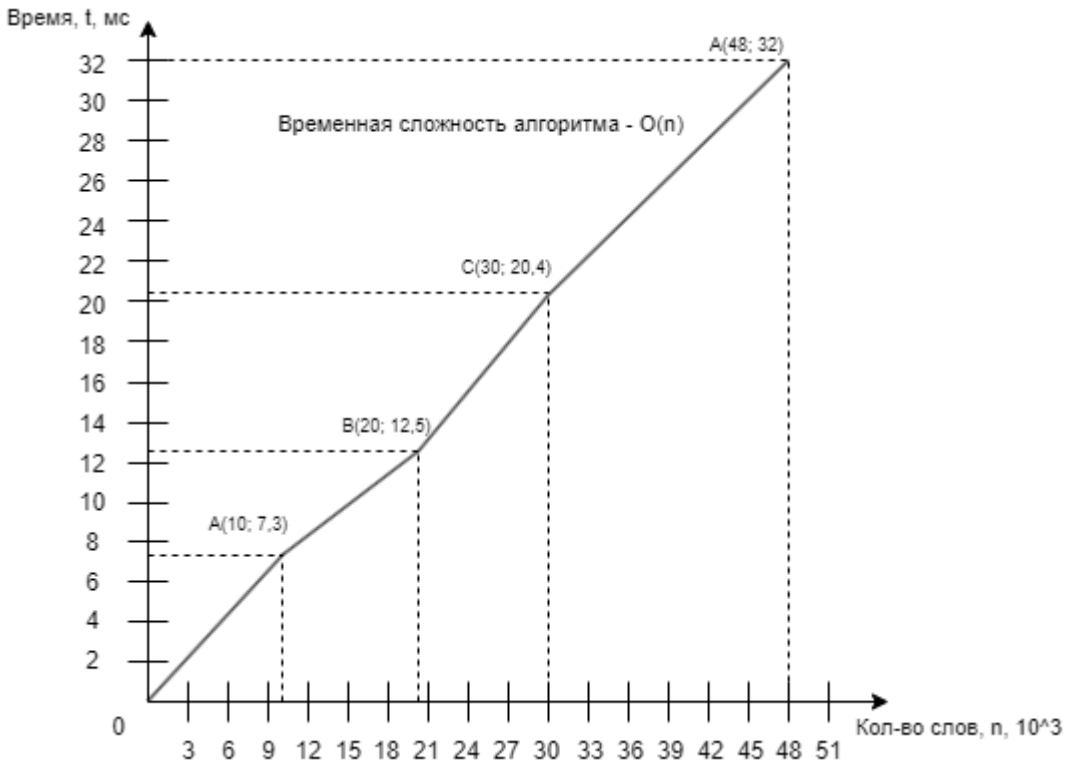
```python
def parse_text(self, file_paths):
    words = []
    for file_path in file_paths:
        with open(file_path, "r", encoding="utf-8") as file:
            content = file.read()
            file_words = re.findall(r"\w+", content.lower())
            words.extend(file_words)

    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
```

concordance:
ключ - слово из текста

значение - контексты использования этого слова
Изначально пуст, заполняется после нажатия на кнопку “Просмотр конкорданса”

```python
def view_concordance(self):
    word = self.search_entry.get().lower()
    if word:
        file_paths = self.selected_file_paths
        concordance = {}

        for file_path in file_paths:
            with open(file_path, "r", encoding="utf-8") as file:
                content = file.read()
                words = re.findall(r"\w+", content.lower())

                for i in range(len(words)):
                    if words[i] == word:
                        context = " ".join(words[max(0, i - 5):i + 6])
                        if file_path in concordance:
                            concordance[file_path].append(context)
                        else:
                            concordance[file_path] = [context]
```

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 построения корпусов текстов, видов разметки и способов аннотирования, а также инструменты работы с корпусами текстов. В результате выполнения лабораторной работы был построен корпус текстов и разработан корпусный менеджер. Разработанное приложение может быть полезно для исследования текстовых данных, выявления ключевых тем, идентификации повторяющихся паттернов или анализа языковых особенностей. Оно также может быть полезно при изучении текстов научных работ или в литературных исследованиях, так как позволяет создавать конкорданс.