Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра интеллектуальных информационных технологий

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

Лабораторная работа №6
«Диалоговая система с поддержкой естественного языка»

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

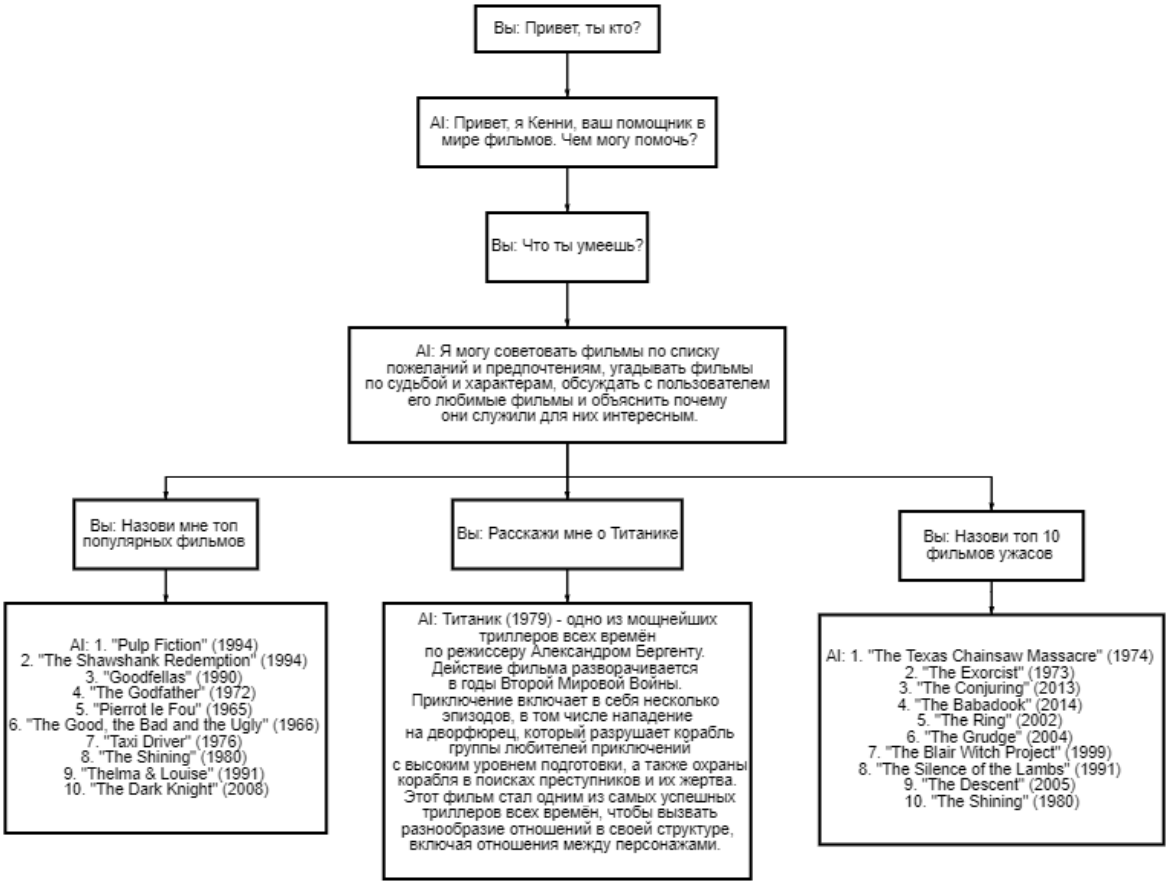
Минск 2024

Цель работы: Освоить принципы разработки диалоговых систем с поддержкой естественного языка.

Задание:

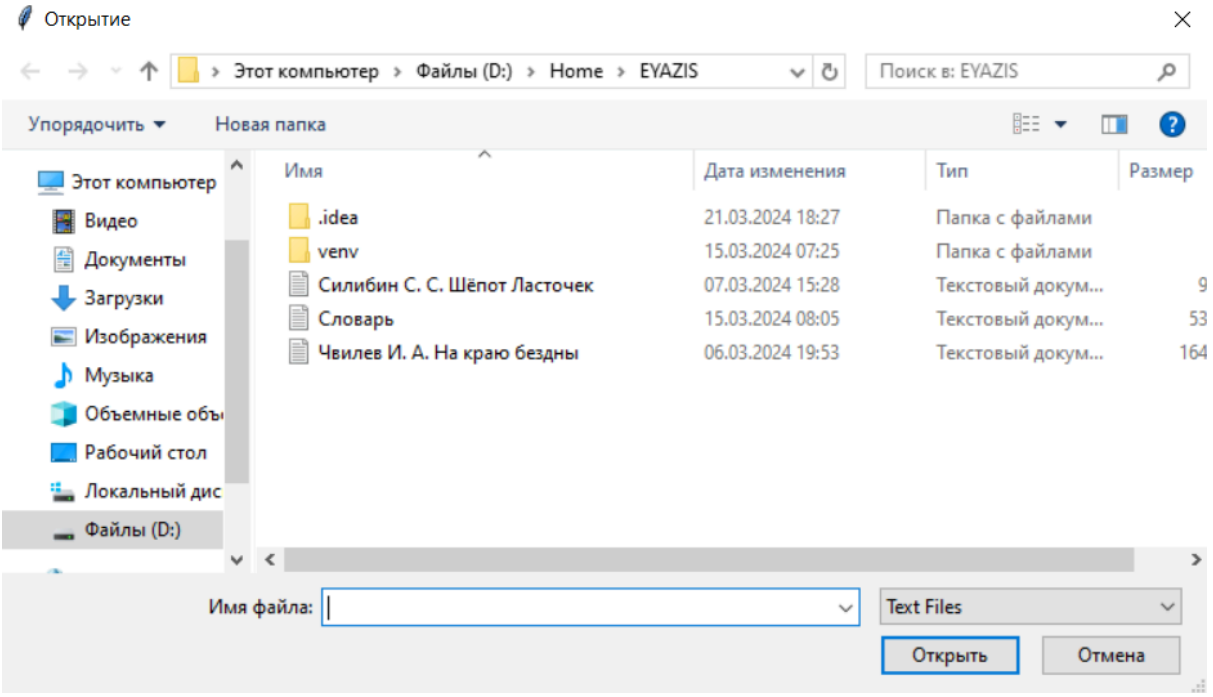
1. Изучить основы создания диалоговых систем с поддержкой естественного языка.
2. Закрепить навыки программирования при решении задач организации диалогового взаимодействия с поддержкой естественного языка.

Описание логической структуры сценария диалога:



Алгоритм работы программы:

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

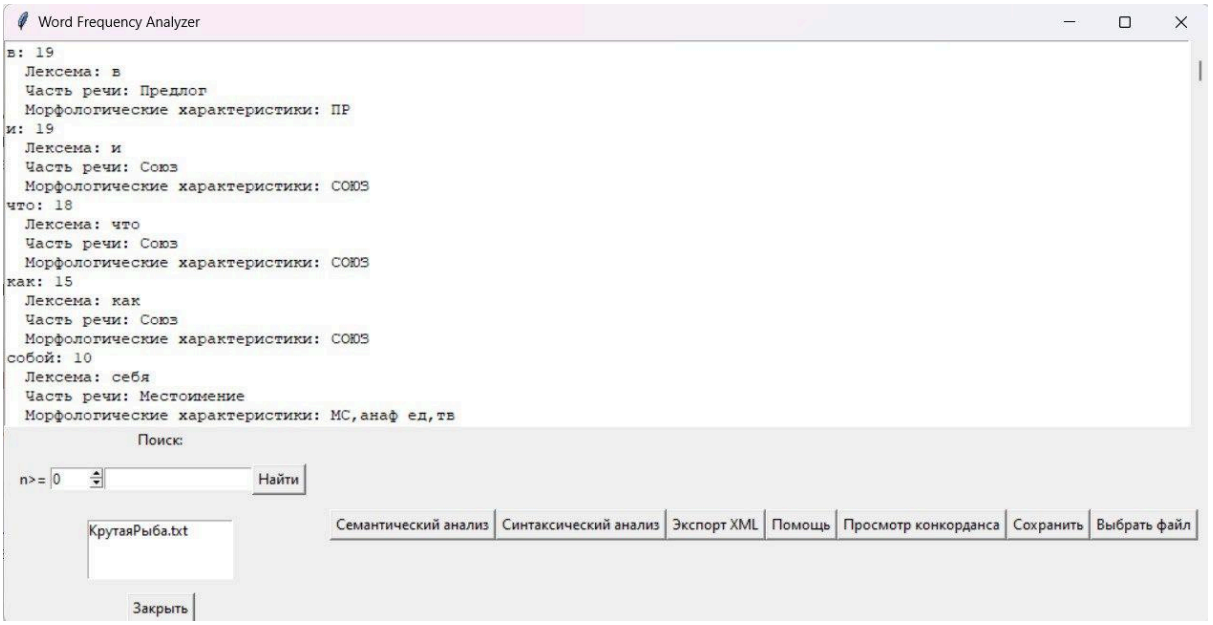


2) Программа проходит по всему тексту, получает все слова из файла, подсчитывает их частоту, производит морфологический анализ и создает описание для каждого слова.

```
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.

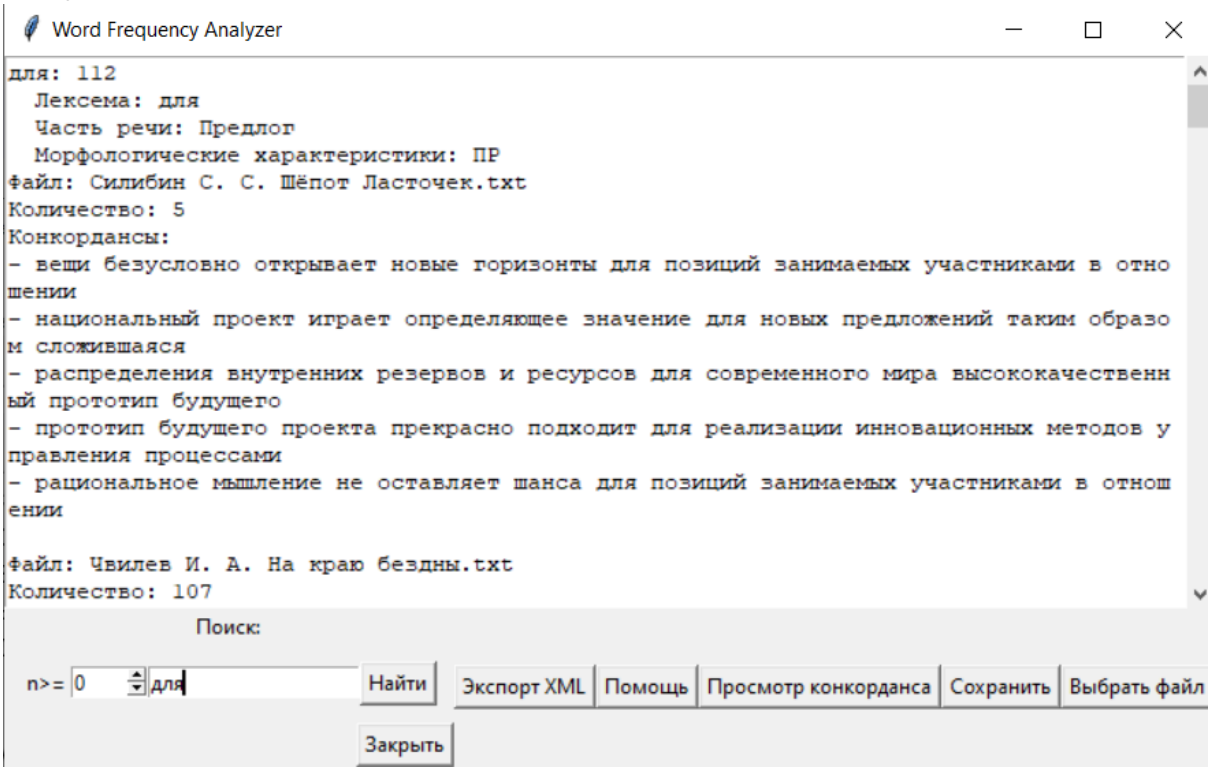
з) Пользователь нажимает на кнопку “Семантический анализ”. Программа переходит к шагу 10.

и) Пользователь нажимает на кнопку “Чат с помощником”. Программа переходит к шагу 11.

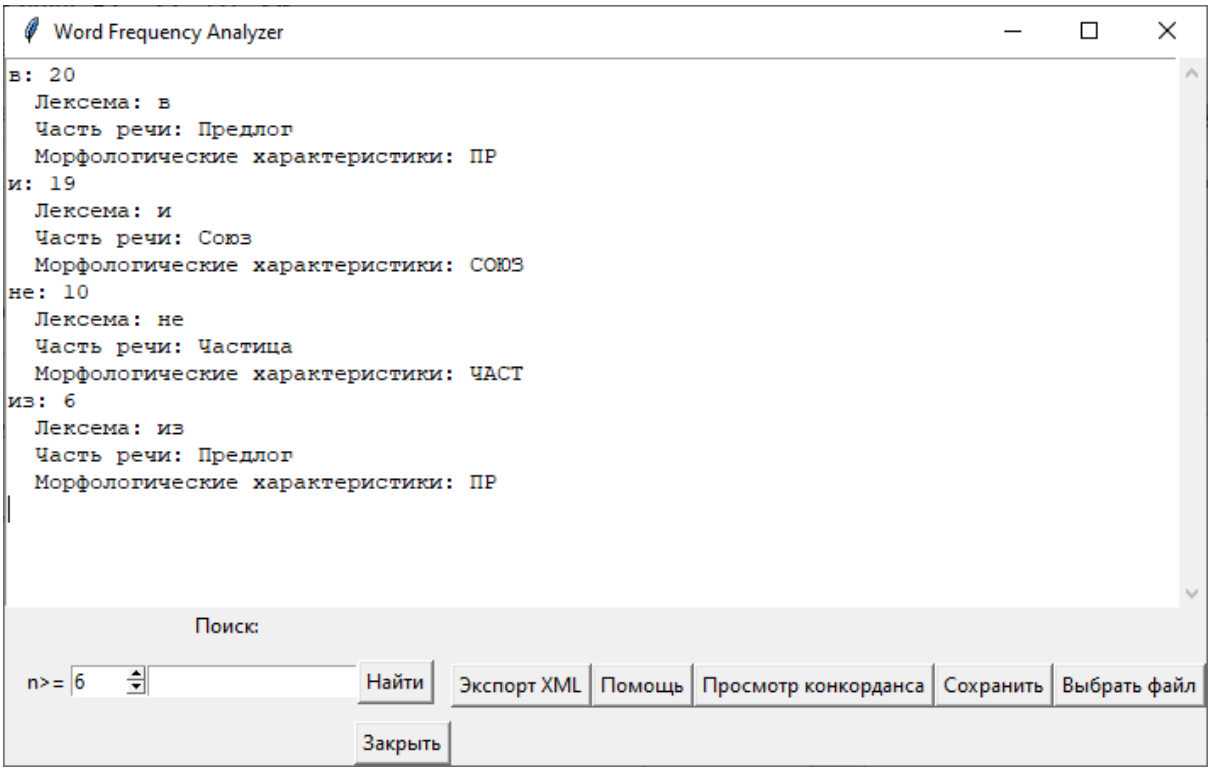
й) Пользователь закрывает программу. Программа переходит к шагу 12.

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

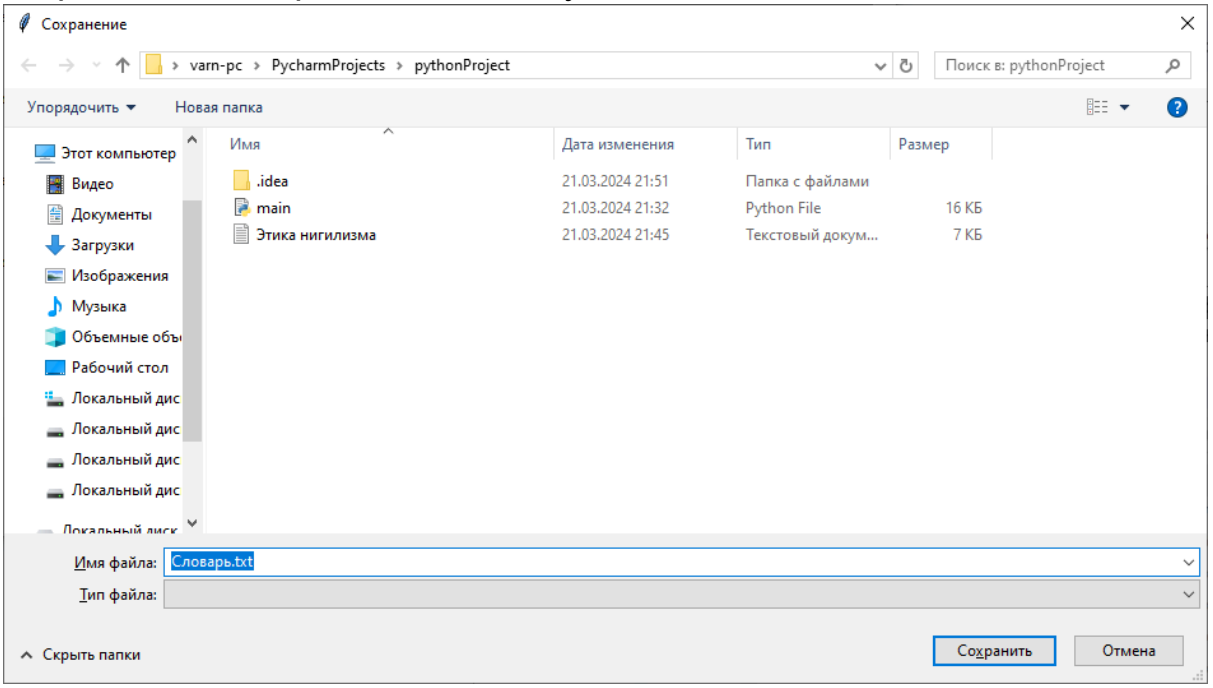
шагу 3.1.



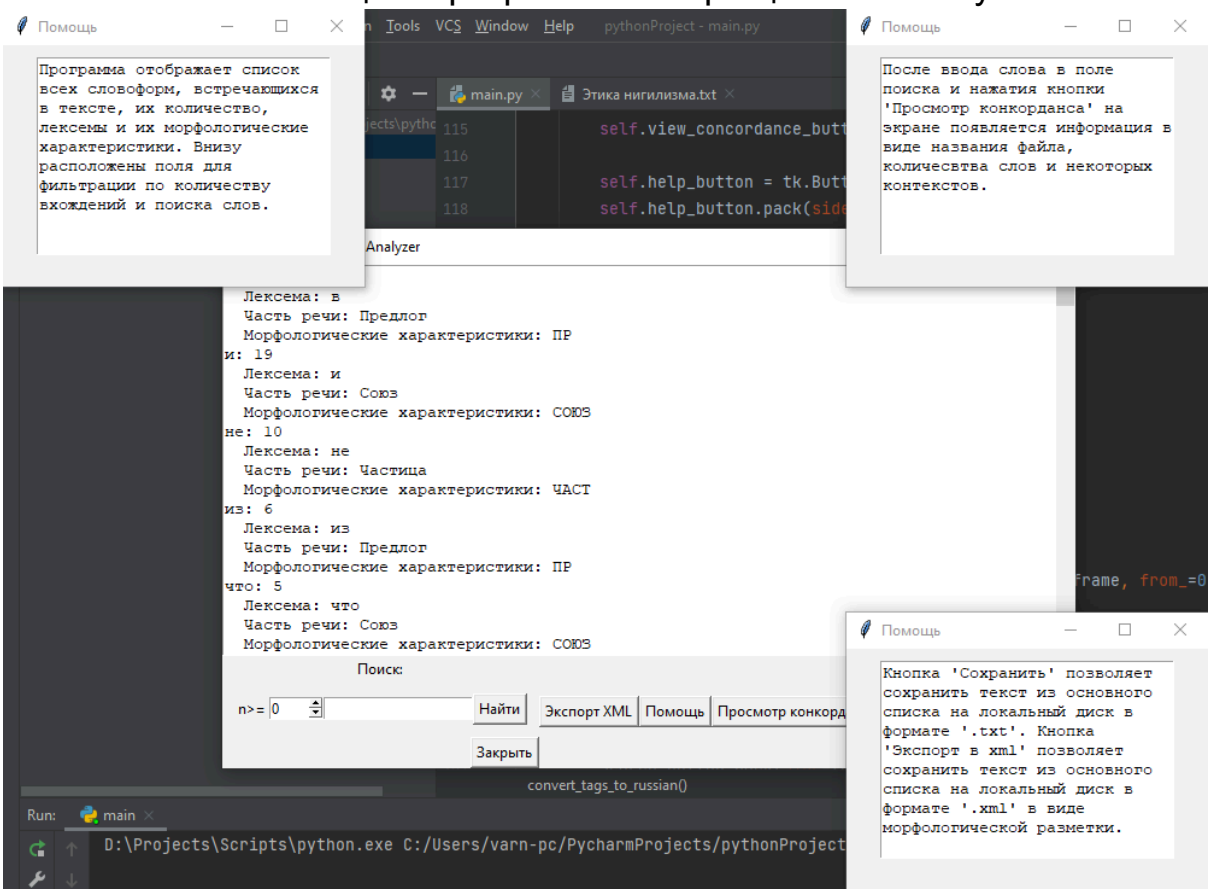
5) Список выведенных слов обновляется, включая только те, число которых больше или равно значению в поле “n>=”. Программа возвращается к шагу 3.1



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



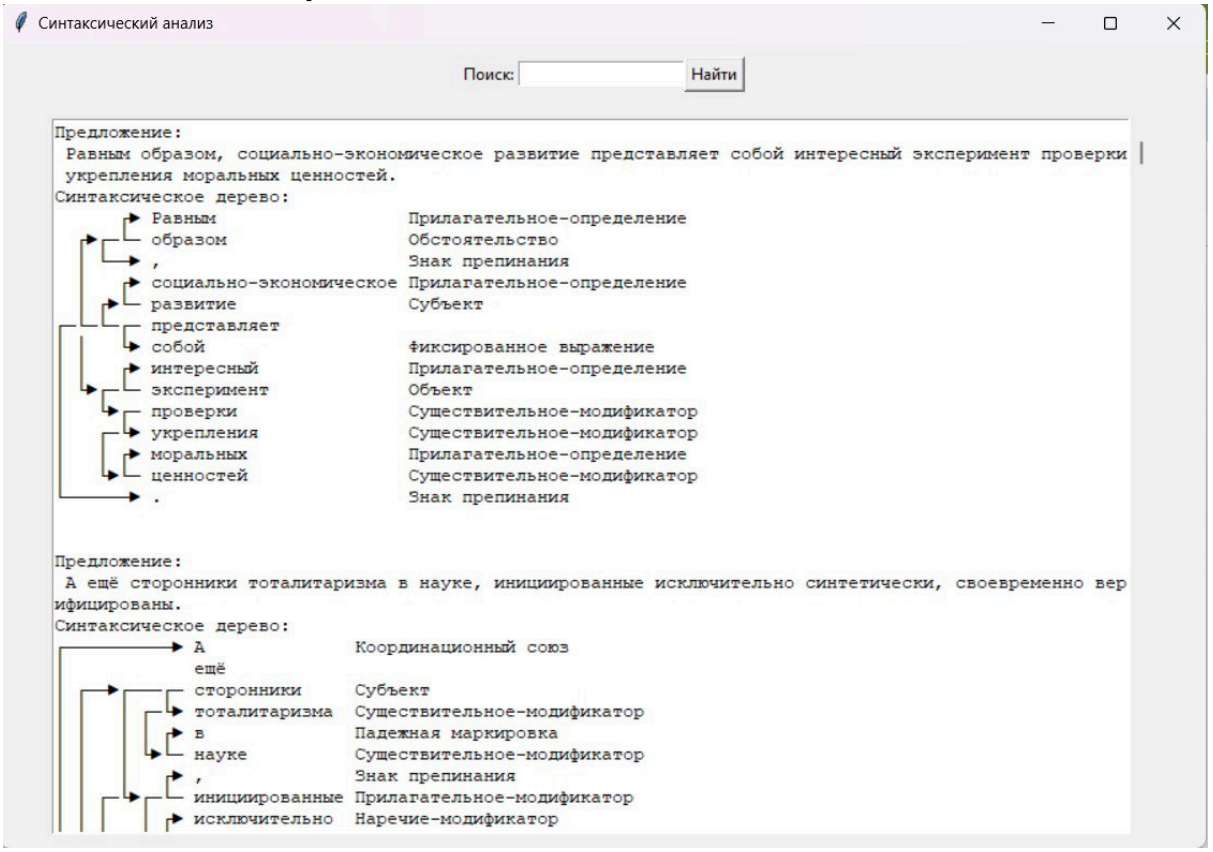
7) Открываются 3 окна-подсказки, объясняющие пользователю что, где и для чего расположено. Закрыть их можно повторным нажатием на “Помощь”. Программа возвращается к шагу 3.1



8) Результаты работы программы экспортируются в файл формата XML.

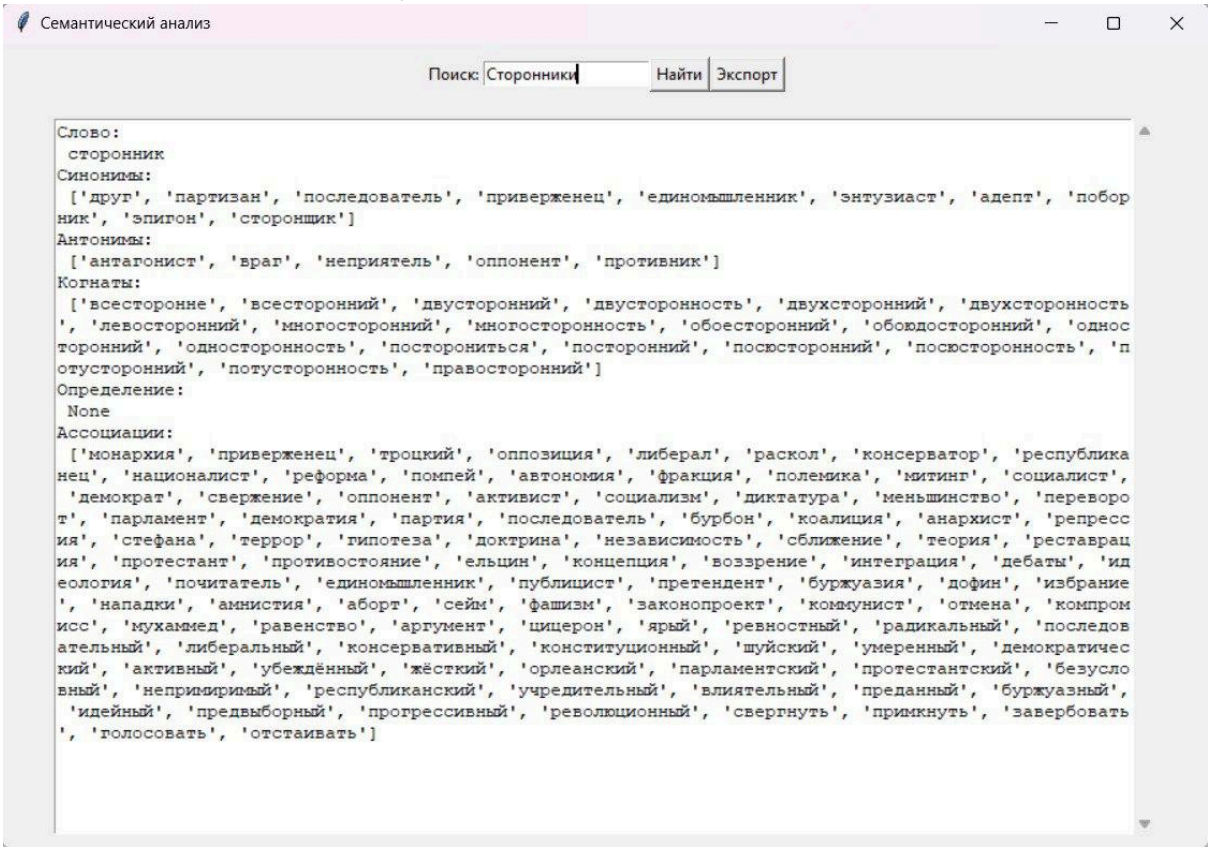
```xml
<?xml version='1.0' encoding='windows-1251'?>
<text>
<w>банальные<ana lemma="банальный" pos="Прилагательное (полное)" gram="ПРИЛ,кач мн,им" />
</w>
<w>но<ana lemma="но" pos="Союз" gram="СОЮЗ" />
</w>
<w>неопровержимые<ana lemma="неопровержимый" pos="Прилагательное (полное)" gram="ПРИЛ,кач мн,им" />
</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.

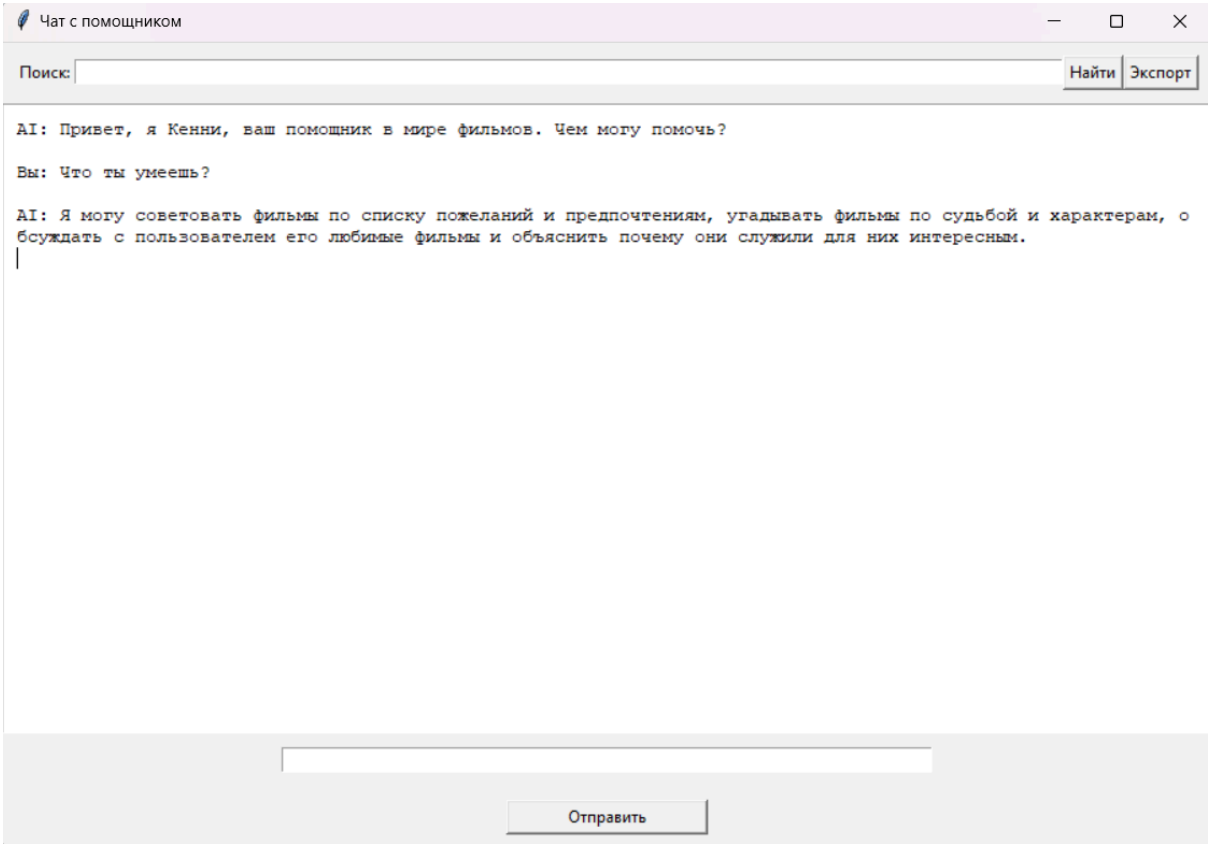


10) Программа формирует окно с семантическим анализом. Здесь же можно вводить запрос в строку поиска в соответствующее

поле,и, нажав на кнопку “Найти”, семантический анализ обновится.

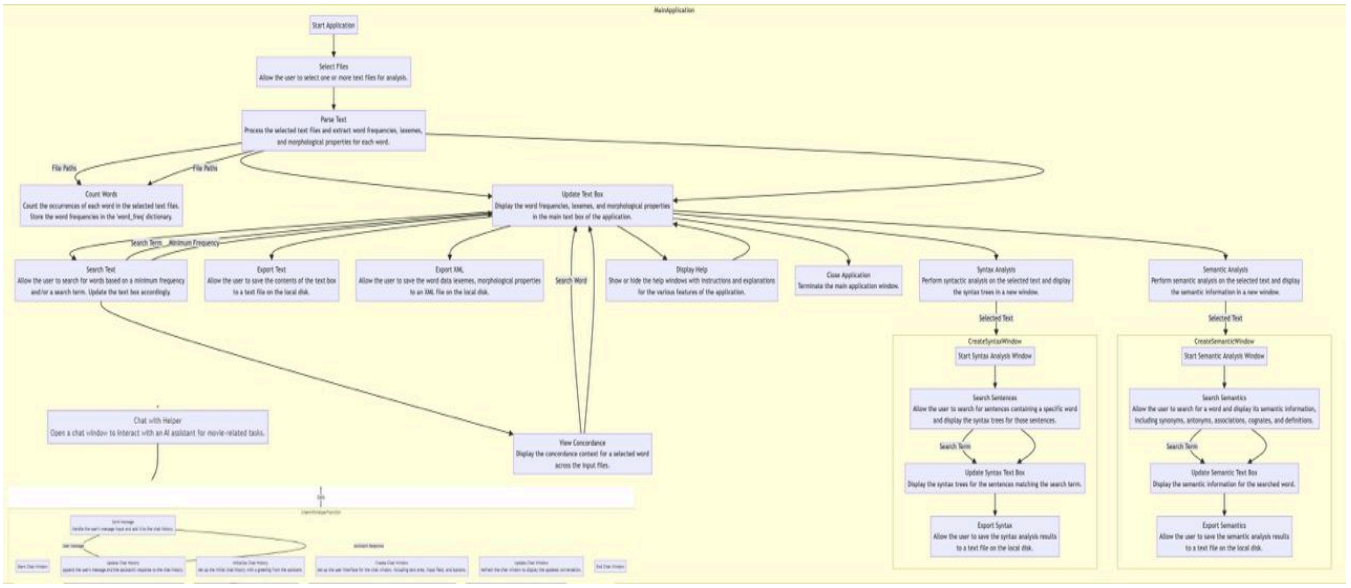


11) Программа формирует окно диалога с помощником. Пользователь может ввести сообщение в поле снизу и нажать отправить, чтобы получить ответ от помощника. Здесь также присутствует функция поиска по истории сообщений и экспорта файла.

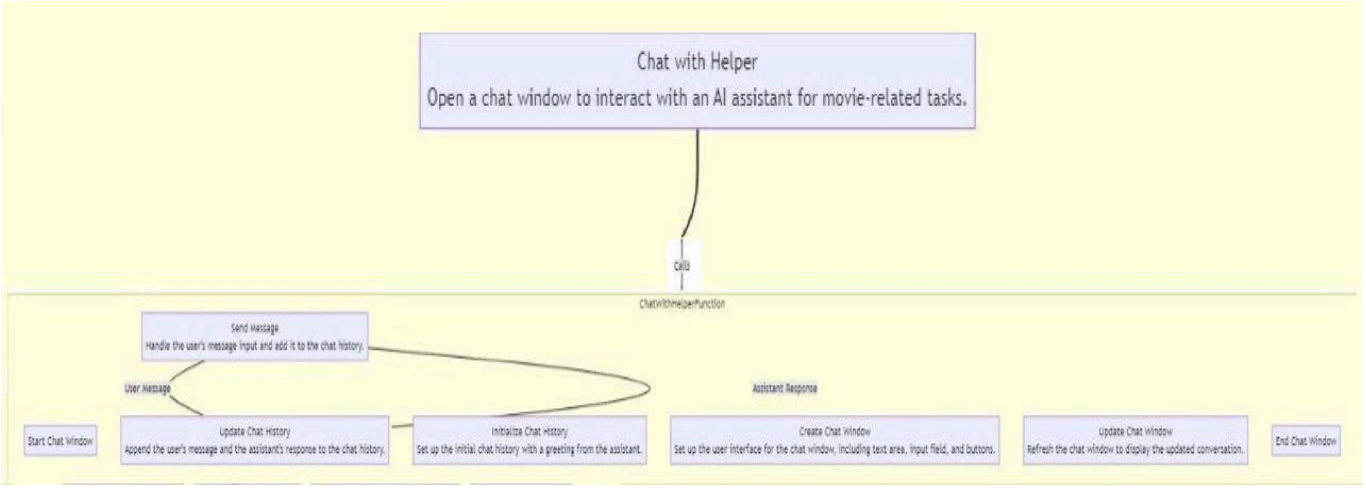


12) Программа завершает работу.

Структурно-функциональная схема приложения:



Структурно-функциональная схема приложения (часть с чат-помощником)



Использованные структуры хранения данных:

Алгоритм использует функционал сервиса AI21.
Он представлен в виде класса J2ChatAI:

```python
class J2ChatAI:
    def __init__(self):
        self.chat_url = "https://api.ai21.com/studio/v1/j2-ultra/chat"
        self.api_key = settings.chat_api_key

    def make_request_to_chat(self, messages: list[dict[str, str]], model_description: str):
        payload = {
            "numResults": 1,
            "temperature": 0.7,
            "messages": messages,
            "system": model_description
        }

        headers = {
            "accept": "application/json",
            "content-type": "application/json",
            "Authorization": f"Bearer {self.api_key}"
        }

        with httpx.Client() as client:
            result = client.post(self.chat_url, headers=headers, json=payload, timeout=20)
            return result.json()
```

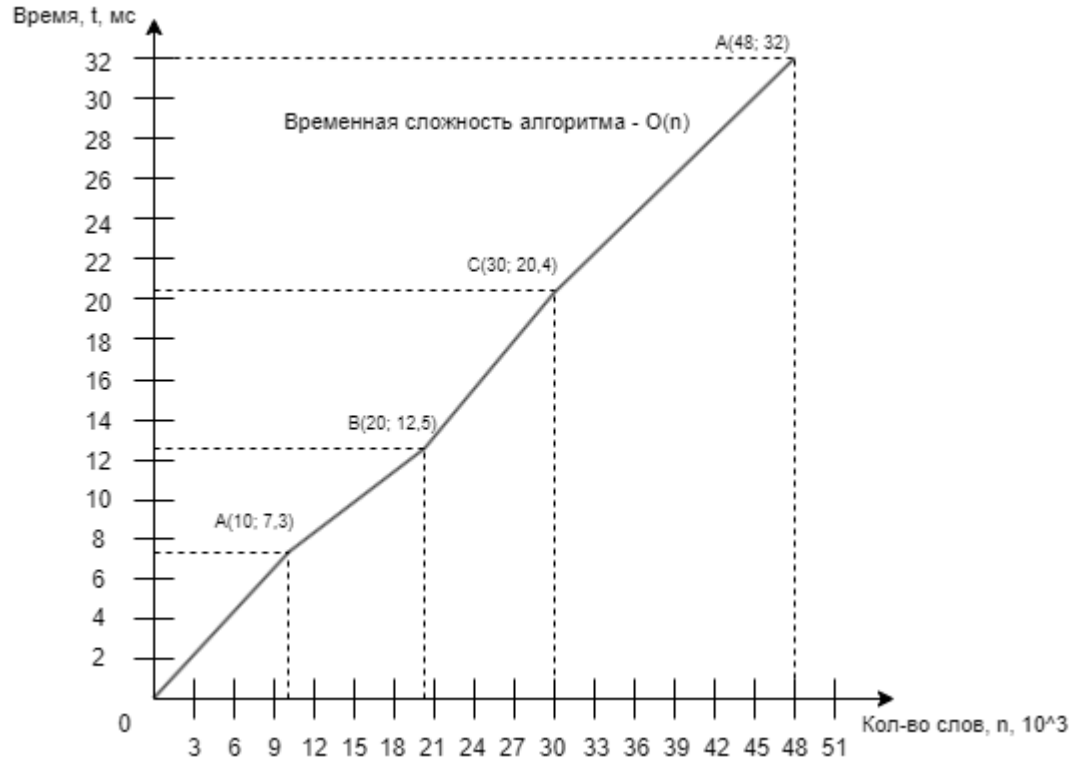
В классе определён основной метод: make_request_to_chat(self, messages: list[dict[str, str]], model_description)

Он принимает два аргумента:
messages представляет собой список сообщений.

Сообщение - словарь, где ключ - роль (пользователь или ассистент), а значение - текст сообщения.
model_description - описание модели, то есть текст, задающий поведение модели.

Полученный ответ добавляется в список сообщений и отображается в окне чата с помощником.

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: В результате выполнения данной работы были успешно изучены основы создания диалоговых систем с поддержкой естественного языка и закреплены навыки программирования, необходимые для решения задач организации диалогового взаимодействия. Мы ознакомились с ключевыми концепциями и методами в области обработки естественного языка, а также получили практический опыт в разработке диалоговых систем. Разработанное приложение может быть полезно, поскольку в настоящее время диалоговые системы с поддержкой естественного

языка становятся все более популярными и востребованными. Они находят применение в различных областях, таких как чат-боты, виртуальные помощники, системы автоматизации клиентского обслуживания и многое другое.