МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

ИНТЕЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

ОТЧЁТ

По лабораторной работе № 2

Выполнил:
Студент группы ИИ-22
Копанчук Евгений Романович
Проверил:
Булей Е. В.

Брест – 2024

Ход работы

Задание: Разработка корпусного менеджера

Разработать приложение, которое предоставляет пользователю возможность обрабатывать фрагменты текста на естественном языке, запрашивая различные частотные и морфологические характеристики словоформ.

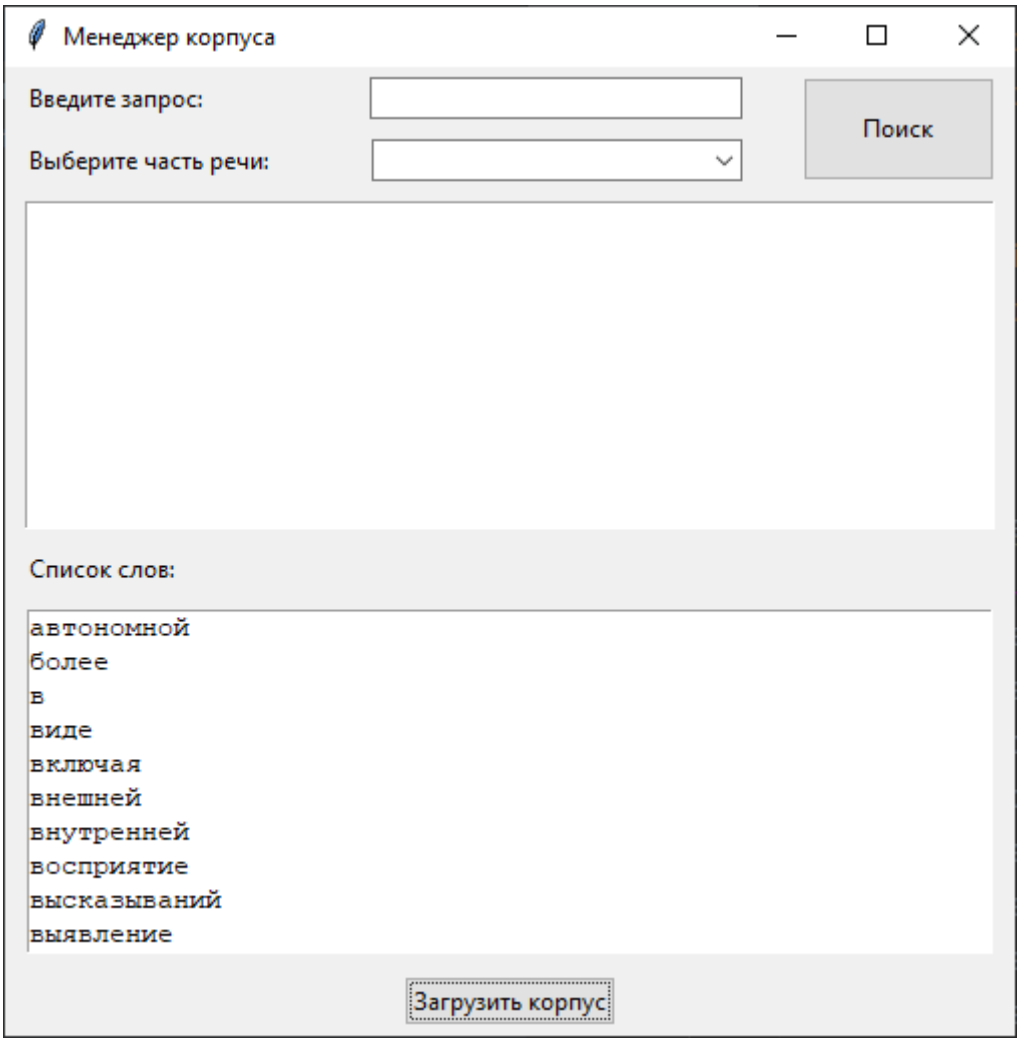
Требования:

Входные данные: Пользователь вводит фрагмент текста (фразу или слово) на естественном языке в качестве запроса к корпусному менеджеру.

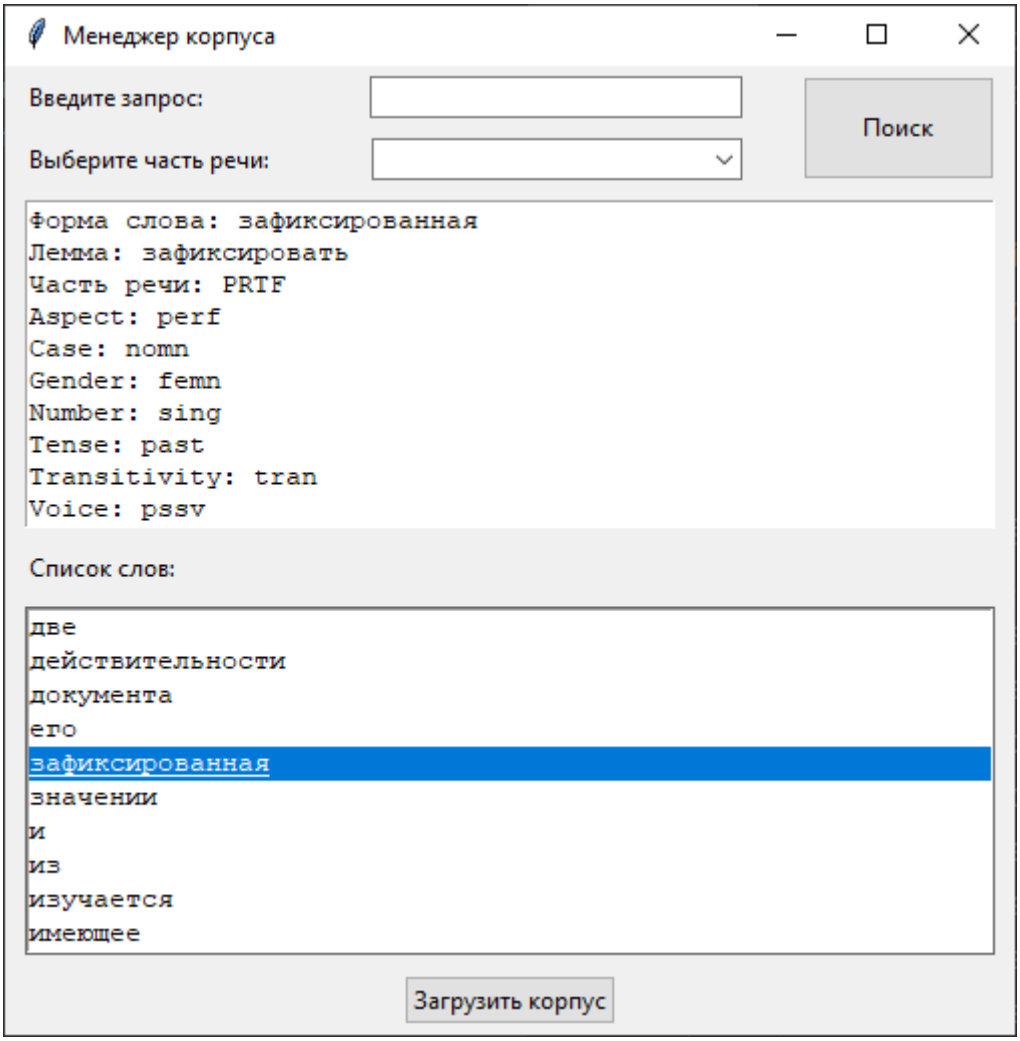
Выходные данные: Приложение предоставляет следующие выходные данные:

- Частотные характеристики словоформ и лексем.
- Грамматические категории.
- Леммы слов.
- Морфологические характеристики словоформ.

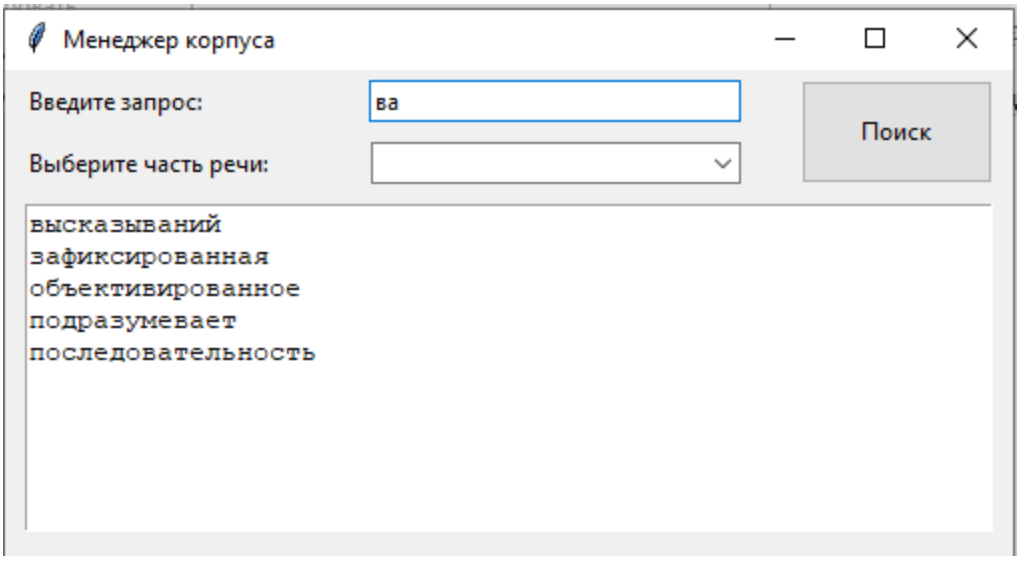
Взаимодействие с пользователем: Пользователь взаимодействует с приложением через графический интерфейс, который должен быть интуитивно-понятным и дружественным для пользователя.



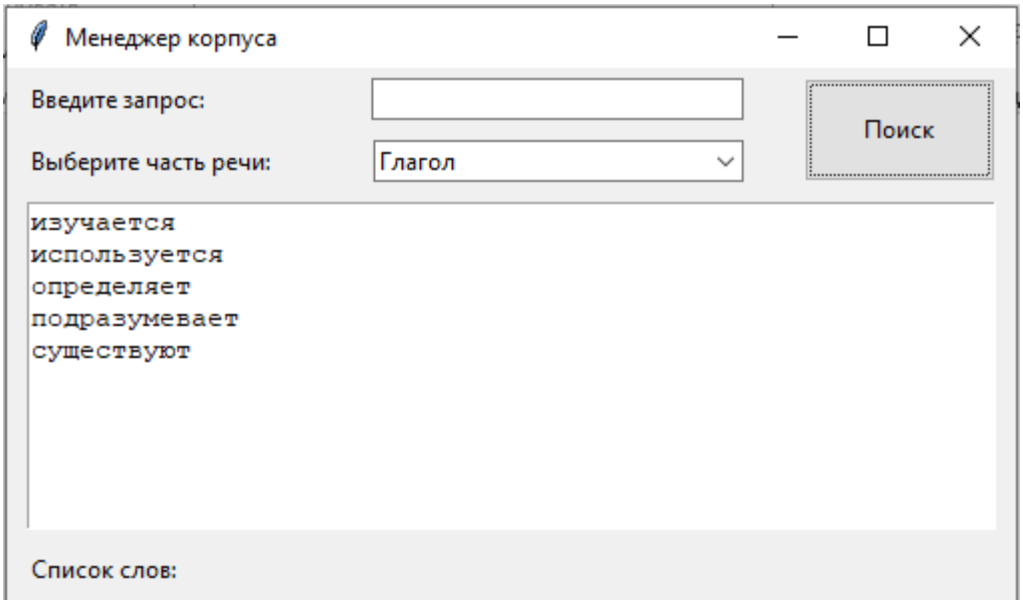
Приложение с загруженным корпусом текста



Просмотр характеристик слова



Поиск по вхождению подстроки в лексеме



Поиск по части речи

Код программы:

```
import tkinter as tk
from tkinter import filedialog
from tkinter import ttk
import os
import nltk
from pymorphy3 import MorphAnalyzer
from docx import Document
import re

class CorpusManagerGUI:
    def __init__(self, master):
        self.master = master
        self.master.title("Менеджер корпуса")
        self.label = ttk.Label(master, text="Введите запрос:")
        self.label.grid(row=0, column=0, padx=10, pady=5, sticky="w")
        self.query_entry = ttk.Entry(master, width=30)
        self.query_entry.grid(row=0, column=1, padx=10, pady=5)
        self.pos_label = ttk.Label(master, text="Выберите часть речи:")
        self.pos_label.grid(row=1, column=0, padx=10, pady=5, sticky="w")
        self.pos_combobox = ttk.Combobox(master, values=["", "Существительное", "Прилагательное", "Глагол", "Наречие"],
                                         width=27)
        self.pos_combobox.grid(row=1, column=1, padx=10, pady=5)
        self.search_button = ttk.Button(master, text="Поиск", command=self.search)
        self.search_button.grid(row=0, column=2, rowspan=2, padx=10, pady=5, sticky="nsew")
        self.results_text = tk.Text(master, height=10, width=60)
        self.results_text.grid(row=2, column=0, columnspan=3, padx=10, pady=5)
        self.word_list_label = ttk.Label(master, text="Список слов:")
        self.word_list_label.grid(row=3, column=0, padx=10, pady=5, sticky="w")
        self.word_list_box = tk.Listbox(master, height=10, width=60, font=("Courier", 10))
        self.word_list_box.grid(row=4, column=0, columnspan=3, padx=10, pady=5)
        self.word_list_box.bind("<<ListboxSelect>>", self.show_word_info)
        self.load_corpus_button = ttk.Button(master, text="Загрузить корпус", command=self.load_corpus)
        self.load_corpus_button.grid(row=5, column=0, columnspan=3, padx=10, pady=5)
        self.analyzer = MorphAnalyzer()
        self.corpus_words = []

    def search(self):
        query = self.query_entry.get()
        pos = self.pos_combobox.get()
        results = self.search_corpus(query, pos)
        self.display_results(results)

    def search_corpus(self, query, pos):
        filtered_words = self.corpus_words

        if pos:
            if pos == "Существительное":
                pos = "NOUN"
            elif pos == "Прилагательное":
                pos = "ADJF"
            elif pos == "Глагол":
                pos = "VERB"
```

```
            elif pos == "Наречие":
                pos = "ADVB"
            else:
                pos = None

            if pos:
                filtered_words = [word_data for word_data in filtered_words if word_data['pos'] == pos]

        results = [word_data['wordform'] for word_data in filtered_words if query.lower() in word_data['wordform'].lower()]
        return results

    def display_results(self, results):
        self.results_text.delete(1.0, tk.END)
        for result in results:
            self.results_text.insert(tk.END, result + "\n")

    def load_corpus(self):
        file_path = filedialog.askopenfilename(initialdir=os.getcwd(), title="Выберите файл корпуса",
                                               filetypes=[("Документы Word", "*.docx")])
        if file_path:
            self.load_docx_corpus(file_path)

    def load_docx_corpus(self, file_path):
        doc = Document(file_path)
        text = "\n".join([paragraph.text for paragraph in doc.paragraphs]).lower()
        pattern = re.compile(r'[а-яА-Я]+')
        words = list(set(pattern.findall(text)))
        self.corpus_words = [self.analyze_word(word) for word in words]
        self.display_word_list()

    def analyze_word(self, word):
        parsed_word = self.analyzer.parse(word)[0]
        return {
            "wordform": parsed_word.word,
            "lemma": parsed_word.normal_form,
            "pos": parsed_word.tag.POS,
            "animacy": parsed_word.tag.animacy,
            "aspect": parsed_word.tag.aspect,
            "case": parsed_word.tag.case,
            "gender": parsed_word.tag.gender,
            "involvement": parsed_word.tag.involvement,
            "mood": parsed_word.tag.mood,
            "number": parsed_word.tag.number,
            "person": parsed_word.tag.person,
            "tense": parsed_word.tag.tense,
            "transitivity": parsed_word.tag.transitivity,
            "voice": parsed_word.tag.voice
        }

    def display_word_list(self):
        self.word_list_box.delete(0, tk.END)
        self.corpus_words = sorted(self.corpus_words, key=lambda x: x['wordform'].lower())
        for word_data in self.corpus_words:
            self.word_list_box.insert(tk.END, word_data['wordform'])

    def show_word_info(self, event):
        selected_index = self.word_list_box.curselection()
        if selected_index:
            selected_index = int(selected_index[0])
            word_data = self.corpus_words[selected_index]
            info_text = f"Форма слова: {word_data['wordform']}\n"
            info_text += f"Лемма: {word_data['lemma']}\n"
            info_text += f"Часть речи: {word_data['pos']}\n"
            for key, value in word_data.items():
                if value is not None and key not in ('wordform', 'lemma', 'pos'):
                    info_text += f"{key.capitalize()}: {value}\n"
            self.results_text.delete(1.0, tk.END)
            self.results_text.insert(tk.END, info_text)

def main():
    nltk.download('punkt')
    root = tk.Tk()
    app = CorpusManagerGUI(root)
    root.mainloop()

if __name__ == "__main__":
    main()
```