Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине: «Естественно-языковой интерфейс ИС»

# Тема: «Построение и использование корпусов текстов естественного языка»

Выполнил:

Студент 3 курса

Группы ИИ-21

Литвинюк Т. В.

Проверила:

Якимук А. В.

Брест 2024

**Цель:** 1. изучить принципы построения корпусов текстов, виды разметки и способы аннотирования, инструменты работы с корпусами текстов, 2. построить корпус текстов и разработать корпусный менеджер.

**Ход работы:**

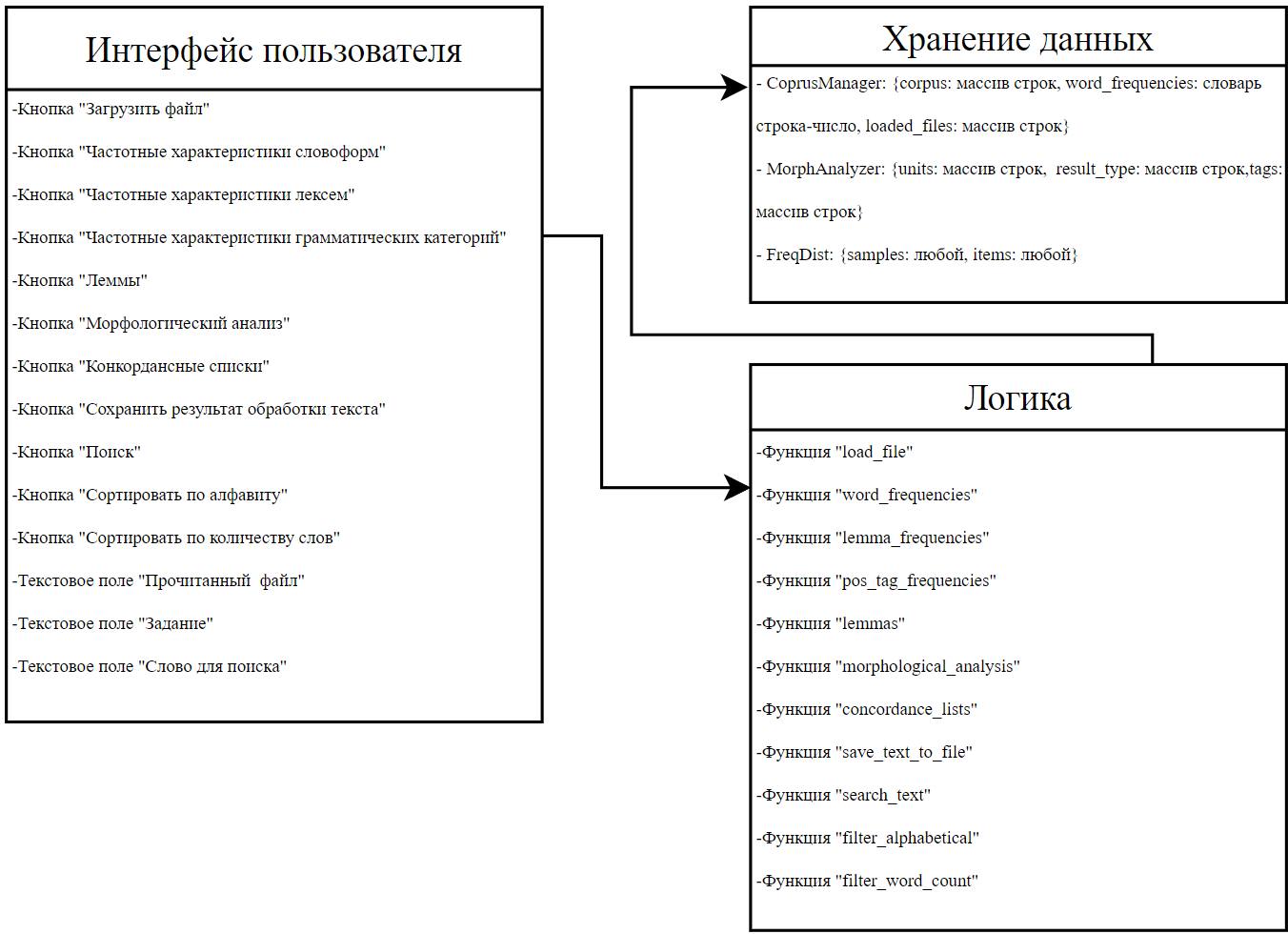
|  |  |  |
| --- | --- | --- |
| **Номер**  **варианта** | **Язык текста** | **Предметная область** |
| 7 | Русский | Транспорт |

**Методические указания:**

Требуется спроектировать и программно реализовать структуры хранения данных, алгоритмы их обработки, необходимые в рамках следующих базовых требований к разрабатываемому приложению:

* входные данные – фрагмент текста (фраза или слово) на естественном языке – запрос корпусному менеджеру;
* выходные данные – частотные характеристики словоформ, лексем, грамматических категорий, леммы, морфологические характеристики словоформ и их метаданные (библиографические, типологические), конкордансные списки, согласно согласованным с преподавателем требованиям к функциональности корпусного менеджера;
* взаимодействие с пользователем посредствам графического интерфейса (интерфейс должен быть интуитивно-понятным и дружественным пользователю);
* наличие системы средств помощи пользователю;
* обеспечение возможности построения, сохранения, просмотра, редактирования, пополнения, фильтрации и поиска по заданному условию, документирования текста и/или его фрагмента в соответствии с реализуемой функциональностью корпусного менеджера;
* поддержка различных форматов представления входных данных (TXT, RTF, PDF, DOC, DOCX).

**Структурно-функциональная схема:**



import os

from nltk.tag import pos\_tag

from tkinter import Tk, Button, PanedWindow, Text, filedialog, Entry, Label

import nltk

import docx2txt

from PyPDF2 import PdfFileReader

from nltk.text import ConcordanceIndex

import textract

from nltk.stem import WordNetLemmatizer

from nltk.tokenize import word\_tokenize

from nltk.probability import FreqDist

from nltk.corpus import stopwords

import pymorphy3

import re

from tkinter import Frame

nltk.download('wordnet')

nltk.download('stopwords')

# Создание объекта лемматизатора

lemmatizer = WordNetLemmatizer()

nltk.download('averaged\_perceptron\_tagger')

class CorpusManager:

    def \_\_init\_\_(*self*):

        self.corpus = []

        self.word\_frequencies = {}

        self.loaded\_files = []

    def load\_file(*self*, *file\_path*):

        \_, extension = os.path.splitext(file\_path)

        text = ""

        if extension == '.txt':

            with open(file\_path, 'r', *encoding*='utf-8') as file:

                text = file.read()

        elif extension == '.rtf':

            text = textract.process(file\_path, *encoding*='utf-8').decode('utf-8')

        elif extension == '.pdf':

            with open(file\_path, 'rb') as file:

                reader = PdfFileReader(file)

                for page\_num in range(reader.numPages):

                    text += reader.getPage(page\_num).extractText()

        elif extension == '.docx':

            doc = docx2txt.process(file\_path)

            for paragraph in doc.paragraphs:

                text += paragraph.text

        elif extension == '.doc':

            text = textract.process(file\_path).decode('utf-8')

        if text:

            self.corpus.append(text)

            self.loaded\_files.append(file\_path)

            print(f"Файл '{file\_path}' успешно загружен.")

    def tokenize\_corpus(*self*):

        for text in self.corpus:

            tokens = word\_tokenize(text.lower())

            filtered\_tokens = [word for word in tokens if word not in stopwords.words('russian')]

            self.word\_frequencies.update(FreqDist(filtered\_tokens))

    def analyze\_pos\_tags(*self*):

        pos\_tags = []

        for text in self.corpus:

            tokens = word\_tokenize(text)

            pos\_tags.extend(pos\_tag(tokens))

        return pos\_tags

# Создание экземпляра корпусного менеджера

manager = CorpusManager()

# Загрузка файла

def load\_file():

    file\_path = filedialog.askopenfilename(*filetypes*=(("Текстовые файлы", "\*.txt"), ("RTF файлы", "\*.rtf"),

                                                       ("PDF файлы", "\*.pdf"), ("Word файлы", "\*.docx;\*.doc"),

                                                       ("Все файлы", "\*.\*")))

    if file\_path:

        manager.load\_file(file\_path)

        text\_box\_left.insert('1.0', f'Загруженный текст из файла:\n{manager.corpus[-1]}\n\n')

# Функция для вывода частотных характеристик словоформ

def word\_frequencies():

    manager.tokenize\_corpus()

    word\_freqs = manager.word\_frequencies

    text\_box\_right.delete('1.0', 'end')  # Очистка правой области перед выводом нового результата

   # text\_box\_right.insert('end', 'Частотные характеристики словоформ:\n')

    for word, freq in word\_freqs.items():

        if word.isalnum():  # Проверка, является ли слово словоформой (без знаков препинания)

            text\_box\_right.insert('end', f'{word}: {freq}\n')

# Создаем экземпляр класса MorphAnalyzer

morph = pymorphy3.MorphAnalyzer()

# Функция для вывода частотных характеристик лексем

def lemma\_frequencies(*manager*):

    manager.tokenize\_corpus()

    lemmas = [morph.parse(word)[0].normal\_form for text in manager.corpus for word in word\_tokenize(text)]

    filtered\_lemmas = [lemma for lemma in lemmas if lemma not in stopwords.words('russian')]

    lemma\_freqs = FreqDist(filtered\_lemmas)

    text\_box\_right.delete('1.0', 'end')  # Очистка правой области перед выводом нового результата

   # text\_box\_right.insert('end', 'Частотные характеристики лексем:\n')

    for lemma, freq in lemma\_freqs.items():

        text\_box\_right.insert('end', f'{lemma}: {freq}\n')

# Функция для вывода частотных характеристик грамматических категорий

def pos\_tag\_frequencies():

    manager.tokenize\_corpus()

    pos\_tags = manager.analyze\_pos\_tags()

    pos\_tag\_freqs = FreqDist(tag for \_, tag in pos\_tags)

    text\_box\_right.delete('1.0', 'end')  # Очистка правой области перед выводом нового результата

   # text\_box\_right.insert('end', 'Частотные характеристики грамматических категорий:\n')

    for tag, freq in pos\_tag\_freqs.items():

        text\_box\_right.insert('end', f'{tag}: {freq}\n')

# Функция для вывода лемм

def lemmas():

    manager.tokenize\_corpus()

    lemmas = []

    for text in manager.corpus:

        tokens = word\_tokenize(text)

        for token in tokens:

            lemma = lemmatizer.lemmatize(token.lower())

            lemmas.append(lemma)

    filtered\_lemmas = [lemma for lemma in lemmas if lemma not in stopwords.words('russian')]

    unique\_lemmas = set(filtered\_lemmas)

    text\_box\_right.delete('1.0', 'end')  # Очистка правой области перед выводом нового результата

   # text\_box\_right.insert('end', 'Леммы:\n')

    for lemma in unique\_lemmas:

        text\_box\_right.insert('end', f'{lemma}\n')

# Функция для вывода морфологических характеристик словоформ и их метаданных

def morphological\_analysis():

    manager.tokenize\_corpus()

    morphological\_tags = []

    for text in manager.corpus:

        tokens = word\_tokenize(text)

        morphological\_tags.extend(pos\_tag(tokens))

    text\_box\_right.delete('1.0', 'end')  # Очистка правой области перед выводом нового результата

    #text\_box\_right.insert('end', 'Морфологические характеристики:\n')

    for tag in morphological\_tags:

        text\_box\_right.insert('end', f'{tag}\n')

# Функция для вывода конкордансных списков

def concordance\_lists():

    manager.tokenize\_corpus()

    concordance\_data = []

    for text in manager.corpus:

        tokens = word\_tokenize(text)

        concordance\_index = ConcordanceIndex(tokens)

        for word in manager.word\_frequencies:

            concordance = concordance\_index.offsets(word)

            if concordance:

                concordance\_data.append((word, concordance))

    text\_box\_right.delete('1.0', 'end')  # Очистка правой области перед выводом нового результата

    #text\_box\_right.insert('end', 'Конкордансные списки:\n')

    for word, concordance in concordance\_data:

        text\_box\_right.insert('end', f'Слово: {word}, Конкордансы: {concordance}\n')

def save\_text\_to\_file():

    file\_path = filedialog.asksaveasfilename(*defaultextension*=".txt", *filetypes*=[("Text files", "\*.txt")])

    if file\_path:

        text = text\_box\_right.get("1.0", "end-1c")

        with open(file\_path, 'w', *encoding*='utf-8') as file:

            file.write(text)

def search\_text():

    query = search\_entry.get()

    text\_to\_search = text\_box\_left.get("1.0", "end-1c")

    text\_box\_left.tag\_remove("highlight", "1.0", "end")

    if query:

        pattern = re.compile(rf"\b{re.escape(query)}\b", *flags*=re.IGNORECASE)

        matches = pattern.finditer(text\_to\_search)

        for match in matches:

            start = match.start()

            end = match.end()

            text\_box\_left.tag\_add("highlight", f"1.0+{start}c", f"1.0+{end}c")

        text\_box\_left.tag\_config("highlight", *background*="yellow")

        text\_box\_left.see("1.0")

    else:

        print("Введите слово для поиска.")

def filter\_alphabetical():

    sorted\_text = '\n'.join(sorted(text\_box\_right.get("1.0", "end-1c").split('\n')))

    text\_box\_right.delete("1.0", "end")

    text\_box\_right.insert("1.0", sorted\_text)

def filter\_word\_count():

    text = text\_box\_right.get("1.0", "end-1c")

    word\_counts = {word: text.count(word) for word in set(text.split())}

    sorted\_words = sorted(word\_counts.items(), *key*=lambda *x*: x[1], *reverse*=True)

    sorted\_text = '\n'.join([f"{word}: {count}" for word, count in sorted\_words])

    text\_box\_right.delete("1.0", "end")

    text\_box\_right.insert("1.0", sorted\_text)

# Создание графического интерфейса

root = Tk()

root.title("Корпусный менеджер")

# Создание PanedWindow для разделения окна на две области

paned\_window = PanedWindow(root, *orient*="horizontal")

paned\_window.pack(*expand*=True, *fill*="both")

# Левая область (загруженный текст)

text\_box\_left = Text(paned\_window, *height*=10, *width*=50)

paned\_window.add(text\_box\_left)

# Правая область (результат анализа)

text\_box\_right = Text(paned\_window, *height*=10, *width*=50)

paned\_window.add(text\_box\_right)

# Создание фрейма для первой колонки кнопок

left\_frame = Frame(root)

left\_frame.pack(*side*="left")

# Создание фрейма для второй колонки кнопок

right\_frame = Frame(root)

right\_frame.pack(*side*="right")

# Размещение кнопок в первой колонке

load\_button = Button(left\_frame, *text*="Загрузить файл", *command*=load\_file, *width*=50)

load\_button.pack()

word\_freq\_button = Button(left\_frame, *text*="Частотные характеристики словоформ", *command*=word\_frequencies, *width*=50)

word\_freq\_button.pack()

lemma\_freq\_button = Button(left\_frame, *text*="Частотные характеристики лексем", *command*=lambda: lemma\_frequencies(manager), *width*=50)

lemma\_freq\_button.pack()

pos\_tag\_freq\_button = Button(left\_frame, *text*="Частотные характеристики грамматических категорий", *command*=pos\_tag\_frequencies, *width*=50)

pos\_tag\_freq\_button.pack()

lemma\_button = Button(left\_frame, *text*="Леммы", *command*=lemmas, *width*=50)

lemma\_button.pack()

morph\_analysis\_button = Button(left\_frame, *text*="Морфологический анализ", *command*=morphological\_analysis, *width*=50)

morph\_analysis\_button.pack()

concordance\_button = Button(left\_frame, *text*="Конкордансные списки", *command*=concordance\_lists, *width*=50)

concordance\_button.pack()

# Размещение кнопок во второй колонке

save\_button = Button(right\_frame, *text*="Сохранить результат обработки текста", *command*=save\_text\_to\_file, *width*=50)

save\_button.pack()

search\_label = Label(right\_frame, *text*="Введите слово для поиска:")

search\_label.pack()

search\_entry = Entry(right\_frame, *width*=50)

search\_entry.pack()

search\_button = Button(right\_frame, *text*="Поиск", *command*=search\_text, *width*=50)

search\_button.pack()

alphabetical\_button = Button(right\_frame, *text*="Сортировать по алфавиту", *command*=filter\_alphabetical, *width*=50)

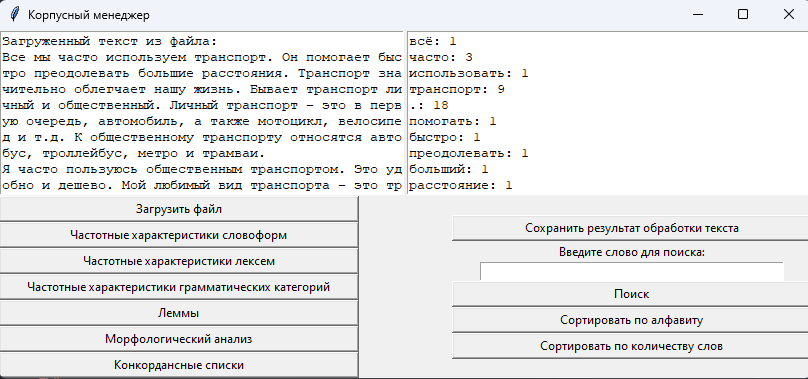
alphabetical\_button.pack()

word\_count\_button = Button(right\_frame, *text*="Сортировать по количеству слов", *command*=filter\_word\_count, *width*=50)

word\_count\_button.pack()

root.mainloop()

# Результат:



**Вывод:** в ходе выполнения лабораторной работы освоил принципы построения корпусов текстов, виды разметки и способы аннотирования, инструменты работы с корпусами текстов.