Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1

По дисциплине: «Естественно-языковой интерфейс ИС»

# Тема: «Разработка информационно-поисковой системы и методы оценки качества ее работы»

Выполнил:

Студент 4 курса

Группы ИИ-21

Ливтинюк Т. В.

Проверила:

Якимук А. В.

Брест 2024

**Цель:** освоить принципы разработки прикладных сервисных программ для решения задачи автоматического лексического и лексико-грамматического анализа текста естественного языка.



**Ход работы:**

на входе – множество естественно-языковых текстов по которым осуществляется

поиск;

• выделение ключевых слов документов осуществляется системой автоматически в

соответствие с формулой 1.6;

• система должна позволять пользователю формулировать ЕЯ-запрос;

• на выходе – список документов, релевантных запросу пользователя, в соответствие с

моделью поиска, согласно варианту;

• результаты поиска должны содержать: активную ссылку на документ, список слов

запроса присутствующих в документе.различных форматов представления входных

данных (TXT, RTF, PDF, DOC, DOCX).

import os

import PyPDF2

import docx

import tkinter as tk

from tkinter import filedialog, messagebox, scrolledtext, StringVar

from collections import defaultdict

import numpy as np

def extract\_text\_from\_pdf(pdf\_path):

    text = ""

    with open(pdf\_path, "rb") as file:

        reader = PyPDF2.PdfReader(file)

        for page in reader.pages:

            text += page.extract\_text() + "\n"

    return text

def extract\_text\_from\_docx(docx\_path):

    doc = docx.Document(docx\_path)

    text = "\n".join([para.text for para in doc.paragraphs])

    return text

def index\_files(directory):

    index = {}

    word\_freqs = defaultdict(lambda: defaultdict(int))  # Частота слов для каждого документа

    total\_words = defaultdict(int)  # Общее количество слов в каждом документе

    for root, \_, files in os.walk(directory):

        for file in files:

            if file.endswith('.pdf'):

                path = os.path.join(root, file)

                text = extract\_text\_from\_pdf(path)

                index[path] = text

            elif file.endswith('.docx'):

                path = os.path.join(root, file)

                text = extract\_text\_from\_docx(path)

                index[path] = text

            # Подсчет частоты слов для каждого документа

            words = text.lower().split()

            for word in words:

                word\_freqs[path][word] += 1

                total\_words[path] += 1

    return index, word\_freqs, total\_words

def calculate\_probabilities(word, word\_freqs, total\_words, smoothing=1):

    """

    Рассчитываем вероятность появления слова в каждом документе.

    """

    probabilities = {}

    for doc\_path in word\_freqs:

        word\_count = word\_freqs[doc\_path].get(word, 0)

        doc\_size = total\_words[doc\_path]

        # Применяем сглаживание Лапласа для обработки случаев отсутствия слова в документе

        probabilities[doc\_path] = (word\_count + smoothing) / (doc\_size + smoothing \* len(word\_freqs))

    return probabilities

def search\_index(index, word\_freqs, total\_words, query):

    query\_words = query.lower().split()

    results = defaultdict(float)  # Вероятностные результаты для каждого документа

    for word in query\_words:

        word\_probs = calculate\_probabilities(word, word\_freqs, total\_words)

        for doc\_path, prob in word\_probs.items():

            results[doc\_path] += np.log(prob)  # Суммируем логарифмы вероятностей

    # Сортировка результатов по вероятности

    sorted\_results = sorted(results.items(), key=lambda x: x[1], reverse=True)

    final\_results = {}

    for doc\_path, score in sorted\_results:

        final\_results[doc\_path] = {'score': score, 'text': index[doc\_path]}

    return final\_results

def browse\_directory():

    directory = filedialog.askdirectory()

    if directory:

        indexing\_label.set("Индексация...")

        root.update()

        index, word\_freqs, total\_words = index\_files(directory)

        indexing\_label.set("Индексация завершена.")

        messagebox.showinfo("Индексация завершена", f"Найдено {len(index)} файлов.")

        return index, word\_freqs, total\_words

    return {}, None, None

def update\_index():

    global index, word\_freqs, total\_words

    index, word\_freqs, total\_words = browse\_directory()

def search\_files():

    query = query\_entry.get()

    if not query:

        messagebox.showwarning("Предупреждение", "Введите запрос для поиска.")

        return

    results = search\_index(index, word\_freqs, total\_words, query)

    result\_text.delete(1.0, tk.END)

    if results:

        for path, data in results.items():

            result\_text.insert(tk.END, f"Найдено в: {path} (Релевантность: {data['score']:.4f})\nТекст:\n")

            result\_text.insert(tk.END, f"{data['text'][:200]}...\n")

            result\_text.insert(tk.END, "-----\n")

    else:

        result\_text.insert(tk.END, "Результаты не найдены.\n")

root = tk.Tk()

root.title("Лабораторная работа 2")

frame = tk.Frame(root)

frame.pack(pady=10)

index = {}

indexing\_label = StringVar()

indexing\_label.set("")

browse\_button = tk.Button(frame, text="Выбрать папку для индексации", command=lambda: update\_index())

browse\_button.pack(pady=5)

query\_label = tk.Label(frame, text="Введите запрос для поиска:")

query\_label.pack(pady=5)

query\_entry = tk.Entry(frame, width=50)

query\_entry.pack(pady=5)

search\_button = tk.Button(frame, text="Поиск", command=search\_files)

search\_button.pack(pady=5)

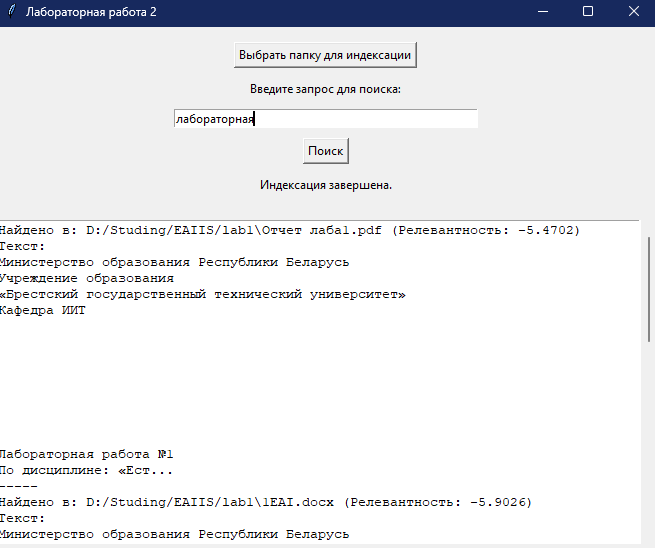
indexing\_status = tk.Label(frame, textvariable=indexing\_label)

indexing\_status.pack(pady=5)

result\_text = scrolledtext.ScrolledText(root, width=80, height=20)

result\_text.pack(pady=10)

root.mainloop()

****

**Вывод:** в ходе выполнения лабораторной работы освоил принципы разработки информационно-поисковых систем для поиска файлов с использованием естественного языка.