

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 2

з дисципліни «Програмне забезпечення інформаційно-пошукових систем. Частина 2. Методи організації пошуку інформації»

тема «Реалізація алгебраїчної моделі подання документів»

|  |  |  |
| --- | --- | --- |
| Виконав  Студент IV курсу  групи КП-01  Беліцький Олександр Сергійович |  |  |

Київ 2024

**Мета**

Реалізувати алгебраїчну (векторно-просторову) модель подання документів, таким чином ознайомившись з найбільш поширеними моделями та набувши практичні навички з їх реалізації.

**Індивідуальне завдання**

1. Для реалізації програмного забезпечення в ході лабораторної роботи може використовуватись будь-який стек (мова програмування, фреймворк і так далі) технологій.

2. Програмне забезпечення може мати будь-який з перелічених інтерфейсів користувача: консольний, веб, мобільний, настільний.

3. Розроблене програмне забезпечення має використовувати векторно-

просторову модель подання документів.

4. Міра TF-IDF, необхідна для реалізації векторно-просторової моделі, визначається відповідно до варіанту студента.

5. Для обчислення подібності між документами та пошуковими запитами необхідно використати косинусну міру. Граничне значення подібності, за якої документ повинен відображатися у результатах запиту, обирається студентом самостійно.

6. Робота з розробленим програмним забезпеченням має бути поділена на наступні етапи:

a. Введення колекції документів.

i. Користувач повинен мати можливість ввести будь-яку невід’ємну кількість документів (хоча б один документ у колекції є обов’язковим).

ii. Допускається (на вибір студента) реалізація введення колекції документів за допомогою її зчитування з окремих текстових файлів, розміщених у певній директорії (один текстовий файл = один документ). У такому випадку, на даному етапі користувач вводить шлях до директорії з файлами.

iii. Після закінчення введення, користувач автоматично переходить на наступний етап.

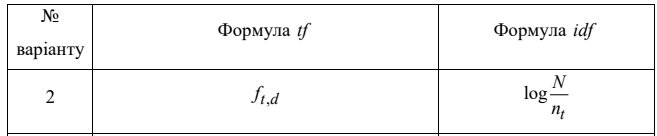
b. Виконання пошукових запитів.

i. На даному етапі користувач вводить пошуковий запит і переглядає його результати.

ii. При показі результатів запиту потрібно обов’язково вивести обчислене значення подібності для кожного документа.

iii. Після виконання пошукового запиту, користувач повинен залишатися на цьому ж етапі, з можливістю ввести новий пошуковий запит.

Номер варіанту – 2. Формули для обрахунку TF-IDF:



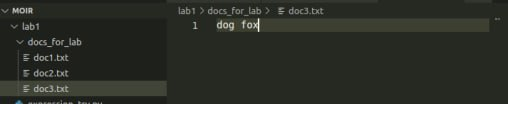
**Хід роботи**

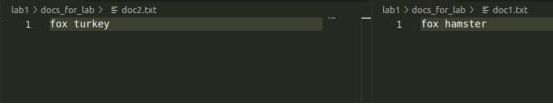
Спочатку визначимось із варіантами індивідуального завдання. Роботу будемо виконувати в IDE Visual Studio Code, використовуючи мову програмування Python. Розробимо консольний інтерфейс, через який будемо задавати шлях до директорії з документами. Далі користувач вводить запити.

Код програми:

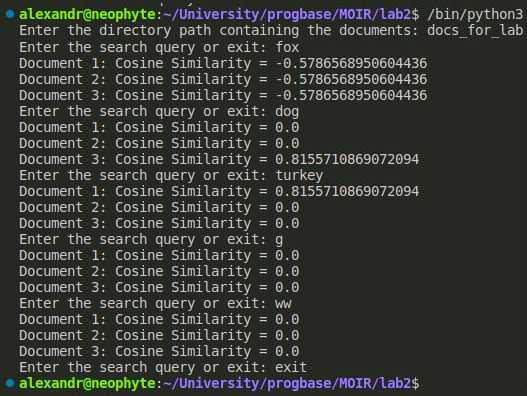
|  |
| --- |
| main.py |
| import os  import math  from collections import Counter  import string  def preprocess\_text(text):  translator = str.maketrans('', '', string.punctuation)  text = text.translate(translator) # Remove punctuation  text = text.lower() # Convert to lowercase  return text  def compute\_tf(documents):  tf\_dict = {}  for doc in documents:  words = preprocess\_text(doc).split()  word\_count = Counter(words)  tf = {word: word\_count[word] for word in word\_count}  tf\_dict[doc] = tf    return tf\_dict  def compute\_idf(documents):  total\_docs = len(documents)  word\_doc\_count = {}  for doc in documents:  words = preprocess\_text(doc).split()  for word in words:  word\_doc\_count[word] = word\_doc\_count.get(word, 0) + 1  idf = {word: math.log(total\_docs / (count + 1)) for word, count in word\_doc\_count.items()}  return idf  def compute\_tf\_idf\_vectors(documents, idf):  tf\_idf\_vectors = []    for doc in documents:  doc\_tf = compute\_tf([doc])[doc] # Compute TF for the current document  tf\_idf\_vector = {word: tf \* idf.get(word, 0) for word, tf in doc\_tf.items()}  tf\_idf\_vectors.append(tf\_idf\_vector)    return tf\_idf\_vectors  def compute\_cosine\_similarity(query, tf\_idf\_vector):  scalar\_product = sum(query.get(word, 0) \* tf\_idf\_vector.get(word, 0) for word in set(query) & set(tf\_idf\_vector))    query\_length = math.sqrt(sum(value \*\* 2 for value in query.values()))  tf\_idf\_length = math.sqrt(sum(value \*\* 2 for value in tf\_idf\_vector.values()))    cosine\_similarity = scalar\_product / (query\_length \* tf\_idf\_length) if (query\_length \* tf\_idf\_length) != 0 else 0    return cosine\_similarity  def read\_documents\_from\_directory(directory):  documents = []  for filename in os.listdir(directory):  with open(os.path.join(directory, filename), 'r', encoding='utf-8') as file:  documents.append(file.read())  return documents  def main():  directory = input("Enter the directory path containing the documents: ")  directory = "docs\_for\_lab"  documents = read\_documents\_from\_directory(directory)  idf = compute\_idf(documents)  while True:  # Read search query from user  query = input("Enter the search query or exit: ")  if query == 'exit':  break  query\_tf = compute\_tf([query])[query]  tf\_idf\_vectors = compute\_tf\_idf\_vectors(documents, idf)    similarities = [compute\_cosine\_similarity(query\_tf, tf\_idf\_vector) for tf\_idf\_vector in tf\_idf\_vectors]  for i, similarity in enumerate(similarities):  print(f"Document {i+1}: Cosine Similarity = {similarity}")  if \_\_name\_\_ == "\_\_main\_\_":  main() |

Створимо вхідні дані. Використаємо документи із першої лабораторної роботи. На скріншотах нижче видно перелік документів та текст всередині них.





Продемонструємо роботу застосунку:



Зі скріншоту бачимо, що для терму “fox” косинусна міра подібності є від’ємною. Це не є збоєм чи неправильним обрахунком. Справа в тому, що за варіантом обрахунок IDF є логарифмічною функцією із дробом всередині. Щоб уникнути ділення на 0 в знаменнику завжди додається 1. Якщо терм присутній у всіх документах, то ми отримуємо ситуацію, наприклад, , де 4 – кількість всіх документів, а 4 + 1 = 5 – кількість документів в яких зустрічається терм. Логарифмічна функція при таких значеннях значеннях буде давати від’ємні значення, отже, і подальший обрахунок буде з від’ємними значеннями.

Для вирішення даної проблеми можна взяти модуль логарифму, але для демонстрації можливого варіанту роботи програми, що описаний вище, вирішено залишити в такому вигляді.

**Висновки**

У результаті лабораторної роботи ми здійснили реалізацію алгебраїчної моделі подання документів, а саме векторно-просторової, моделі подання документів за допомогою мови програмування Python.

Ми використали IDE Visual Studio Code для написання програми та реалізації консольного інтерфейсу. За допомогою цієї програми ми змогли читати дані з документів, а також виконувати пошук за запитами користувача.

Основні кроки реалізації включали в себе читання вхідних даних з файлів, обрахунку необхідних мір TF, IDF, TF-IDF вектору, а також виконання пошуку за запитами користувача та виведення результатів.