The LNM Institute Of Information Technology
Information Retrieval
Project Report

Name: Khalid Mehtab Khan **Roll No:17UCS078**

I. Introduction

This is an implementation of a text retrieval system. The program retrieves text from web pages and creates a local dictionary. The dictionary contains a list of unique words and counts the number of instances of each word in the web pages, used as the database. This dictionary is used as a preprocessed data structure to help score documents for a particular user query.

When the user enters a query the program ranks the documents using a ranking function. This ranking function scores these documents using the dictionary and provides the user with a list of documents relevant to the query, ranked in descending order of relevance.

Relevance Judgement is the mapping of documents that are relevant to particular queries according to a human perspective. This file can be used to check the results of the program.

II. Database

The database used to implement this text retrieval system is formed using data based on 5 topics:
Economics
Android
Covid
Bitcoin
Football.

Online articles, reports and websites related to these topics which could be parsed using the system were added to the database. In total 50 documents (10 of each topic) were used to form the database and using these a dictionary of around 20,000 unique words was created.

A few of the links used in the system are given below:

1. https://hbswk.hbs.edu/item/has-the-new-economy-finally-arrived
2. https://learn.g2.com/android-app-development
3. https://www.gavi.org/vaccineswork/covid-19-vaccine-race
4. https://www.investopedia.com/terms/e/ethereum.asp
5. https://www.sportco.io/article/10-greatest-football-legends-957221

III. Working

Pre-Processing:

Database:

A CSV file was created with columns ‘DocID’ and ‘Doc’.The column Doc contains 50 different links to web pages. These web pages act as the database for our retrieval system.

Text Extraction:

Libraries used:

Url: returns HTML document for each web page link in the database.
BeautifulSoup: helps the program to extract plain text from these web pages by removing the script (Javascript), Style (CSS) and HTML tags from the file.
Nltk: used to tokenize i.e extract words from sentences of the file and create an array.

Removing Irrelevant items:

Using an array of stopwords and symbols we can remove irrelevant symbols which have no meaning and stop words that are words like ‘is’, ‘the’ etc present in huge numbers in all the text.

Creating Inverted Index:

The program loops over each document, scans each word of the document and adds it to the dictionary along with the count of its instances in particular documents.
In this iteration whenever the program finds a word that is not already present in the dictionary it adds it to the dictionary along with the current document the program is iterating over, finds the count of this word in the document and adds the corresponding count to the document id added.

Save the dictionary:

The dictionary is saved as a JSON file which can be used by the ranking function as a pre-processed data structure to score the documents for user Queries.

The Ranking Function:

Score:

A score is given to a document-query pair. When a query is entered, the program tokenizes the query. A score for each word is calculated for a particular document and the sum of all the terms of the query is the score for the document query pair.

Ranking Function Parameters:

Term Frequency in the query:

Counts the number of times a word is present in the query.

Term Frequency in the document:

The program contains a function ‘TFvalue’ which takes in a word, a document and a dictionary as input and returns the count of instances of the word in the document.

Inverse Document Frequency:

The function ‘IDFvalue’ takes in a word and a dictionary as arguments, finds the key of the word in the dictionary and counts the number of values the key has. This count is the number of documents the word is present in.

It further calculates the value :

log(M+1)/c

Where
c = count of documents the word if present in and returns the value.

M = total number of documents

Value of the ranking function:

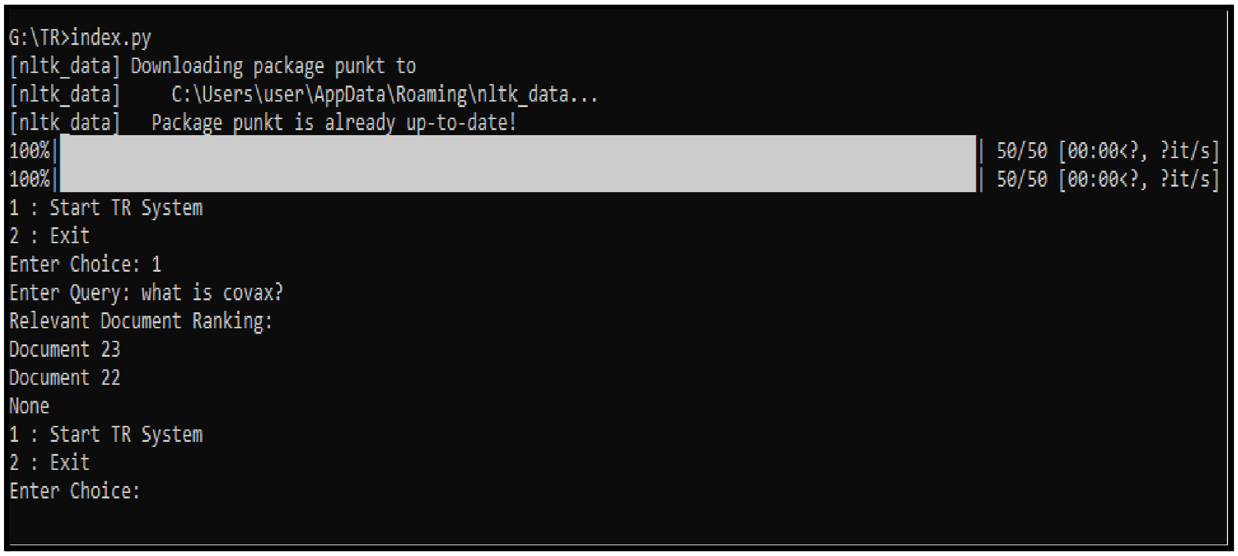
For a given query document pair, the value of the function is the summation of products of the three parameters mentioned above for each query term.

IV. Results

The Ranked List:

Using the values of the ranking function for each document query pair. The documents are then arranged in descending order of these values giving the user a ranked list of documents for the given query.

Search result of the query " What is covax?"



```
G:\TR>index.py
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
100%|                                                          | 50/50 [00:00<?, ?it/s]
100%|                                                          | 50/50 [00:00<?, ?it/s]
1 : Start TR System
2 : Exit
Enter Choice: 1
Enter Query: what is covax?
Relevant Document Ranking:
Document 23
Document 22
None
1 : Start TR System
2 : Exit
Enter Choice:
```