**Assignment 1**

**CSI4107: Information Retrieval and the Internet**

**Winter 2024  - Hiver 2024**

**School of Electrical Engineering and Computer Science**

**École de Génie Électrique et Science Informatique**

**University of Ottawa - Université d'Ottawa**

Professor :  Diana Inkpen

|  |
| --- |
|  |

Group #09

Student 1: Akotondji, Michel #300189827

Student 2: William Beaupré #300174392

Student 3: Charles Devine #300164246

Subsmission date : 11/02/2024

**TASK DISTRIBUTION**

We met up in sessions to work together on this assignment, so we were all able to help each other. Even though everyone had a part to do with everything, here are the main contributions led by each group member:

Michel : Test queries parsing

Charles : Collection parsing

William : TF-IDF and Cosine similarity calculation

**FUNCTIONALITY OF THE PROGRAM**

In step one of the programs, we start by preprocessing the documents provided. This process includes, the removal of the stop words provided, the stemming of the words and the tokenization of the words found in the documents. In the second step, we created an inverted index using the preprocessed tokens that were outputted from step one. We then stored this inverted index in a Json file. In the third step, we used the inverted index and the dictionary and from there we calculate the term frequency (TF) then the inverse document frequency (IDF) then the TF-IDF. From there, we calculate the cosine similarity of the query/documents.

**INSTRUCTIONS**

First ensure that you have all the necessary files in the right folders. Here’s what your file structure should look like this before the main program.

Files in the 4107-a1 folder:

* main.py
* stopwords.txt
* test\_queries.txt
* coll (folder containing all the collection documents)
* README
* Test Runs (folder containing the results from 2 test runs)

Here are the libraries needed to be downloaded:

* numpy : pip install numpy
* nltk : pip install nltk

The next step is to run the main.py program from within the 4107-a1 folder. The program will be taking the inputs out of the test\_queries.txt file and outputting a list of the program in the results.txt file.

Here is what your file structure should look like once the program has ran:

* main.py
* stopwords.txt
* test\_queries.txt
* results.txt
* tokens.json (Contains the tokens of each document found in the coll folder)
* indexed\_topics.json (Contains the tokens of each query found in test queries)
* coll (folder containing all the collection documents)
* README

**EXPLANATIONS**

Optimizations:

* Stemming: We’ve used the stemmers to reduce the length of the text and to reduce the number of different tokens. For example: if the term explanation and the term explaining were both in the same document they would both be stored as the token explain
* Use of custom stop words : We’ve used the custom stop words provided which reduced the amount of tokens we had.
* Only calculating the TD-IDF of documents that had at least one token that matches the tokens found in the query.

Size of the vocabulary: The vocabulary we got was of a total 153352 tokens found.

Sample of 100 tokens from vocabulary : 'col', 'friendship', 'dejoi', 'allow', 'wbztv', 'defiant', 'jump', 'chernenko', 'morn', 'modern', 'street', 'separ', 'novemb', 'uniqu', 'plate', 'mikhail', 'mark', 'particip', 'bangkok', 'wound', 'union', 'indic', 'polici', 'indi', 'error', 'republican', 'come', 'announc', 'bruce', 'revolt', 'school', 'command', 'crime', 'charg', 'backer', 'gain', 'theatric', 'khrushchev', 'georg', 'oblivion', 'repli', 'immigr', 'sharpen', 'fouropera', 'revers', 'vinh', 'ho', 'cb', 'counti', 'heal', 'septemb', 'overthrown', 'weapon', 'divis', 'close', 'top', 'solzhenitsyn', 'sort', 'foreign', 'want', 'hard', 'bottom', 'forbidden', 'dispatch', 'jailer', 'polic', 'welcom', 'januari', 'expect', 'version', 'wrote', 'destroy', 'winner', 'unruli', 'vike', 'speak', 'fraca', 'reeduc', 'hampshir', 'talk', 'soviet', 'australiavietnam', 'huge', 'call', 'missouri', 'assault', 'robertson', 'irrevers', 'secret', 'york', 'live', 'case', 'despit', 'south', 'direct', 'kick', 'arrest', 'risen', 'sought'

First 10 answers to queries 1 and 25 :

|  |  |  |
| --- | --- | --- |
| Query | Title | Title & Description |
| 1 | 1 Q0 AP880310-0051 1 0.9482 Run\_title  1 Q0 AP880519-0231 2 0.9482 Run\_title  1 Q0 AP880526-0013 3 0.9482 Run\_title  1 Q0 AP880906-0154 4 0.9482 Run\_title  1 Q0 AP880216-0195 5 0.785 Run\_title  1 Q0 AP880218-0255 6 0.785 Run\_title  1 Q0 AP880310-0198 7 0.785 Run\_title  1 Q0 AP880312-0057 8 0.785 Run\_title  1 Q0 AP880405-0072 9 0.785 Run\_title  1 Q0 AP880509-0008 10 0.785 Run\_title | 1 Q0 AP881219-0208 1 0.8785 Run\_title\_description  1 Q0 AP881226-0026 2 0.8785 Run\_title\_description  1 Q0 AP881229-0167 3 0.8785 Run\_title\_description  1 Q0 AP881223-0053 4 0.875 Run\_title\_description  1 Q0 AP880310-0198 5 0.8729 Run\_title\_description  1 Q0 AP880312-0057 6 0.8729 Run\_title\_description  1 Q0 AP880405-0072 7 0.8729 Run\_title\_description  1 Q0 AP880509-0008 8 0.8729 Run\_title\_description  1 Q0 AP880728-0104 9 0.8729 Run\_title\_description  1 Q0 AP880830-0013 10 0.8729 Run\_title\_description |
| 25 | 25 Q0 AP880424-0020 1 0.9299 Run\_title  25 Q0 AP880427-0240 2 0.9289 Run\_title  25 Q0 AP880426-0221 3 0.9276 Run\_title  25 Q0 AP880811-0163 4 0.9244 Run\_title  25 Q0 AP880812-0017 5 0.9212 Run\_title  25 Q0 AP880803-0033 6 0.9172 Run\_title  25 Q0 AP880929-0184 7 0.9172 Run\_title  25 Q0 AP880427-0150 8 0.9086 Run\_title  25 Q0 AP880525-0285 9 0.9081 Run\_title  25 Q0 AP880605-0026 10 0.9065 Run\_title | 25 Q0 AP880929-0184 1 0.9305 Run\_title\_description  25 Q0 AP880606-0019 2 0.9284 Run\_title\_description  25 Q0 AP880605-0026 3 0.9254 Run\_title\_description  25 Q0 AP880402-0121 4 0.9236 Run\_title\_description  25 Q0 AP881027-0024 5 0.9218 Run\_title\_description  25 Q0 AP880812-0017 6 0.9204 Run\_title\_description  25 Q0 AP880701-0027 7 0.9202 Run\_title\_description  25 Q0 AP880217-0158 8 0.9181 Run\_title\_description  25 Q0 AP880525-0285 9 0.9157 Run\_title\_description  25 Q0 AP880917-0094 10 0.9156 Run\_title\_description |

The results show the documents that should be the most relevant to the different queries. The closer the value is to 1 the more relevant the document should be. The cosine similarity was rounded to the 4th decimal place.

It proves to be harder to have a higher relevance score in the calculation that takes into account both the title and description. While looking at the whole sample size there is also a lot more variety in the score given by the title in description. For example, in the first query for the relevance scores we see that from the document ranked 193rd up until the document ranked 643rd all the documents share the same score of .6195. For the calculations on both the title and description, the longest streak of the same value we’ve seen is from the document ranked 863rd up until the 1000th document (they all had the score of 0.3925).

**MEAN AVERAGE PRECISION(MAP)**