CMSC 476/676 Information Retrieval
Homework 4 REPORT

JA52979
Sai Teja Challa

Steps to run the program:

In the IDE terminal, You can run the program described below:
python3 retrieval.py “query1” weight “query2” weight …. (multiple queries if needed).

Here, ‘retrieval.py’ is the main python program. You must have all libraries installed and your current working directory containing python files in order to run the method described above. Make sure that queries on the command prompt are in quotation marks.

Improvements made on implementation:

In this phase, we are not taking any input directory paths from the user. Simply, adding the postings and dictionary file paths in the program. Firstly, I preprocessed the query words by converting them to lowercase and checking if any of the query words are stopwords. Then, loading the dictionary from the file path mentioned, and reading every three lines, first the word, the number of documents containing the word, and the byte offset position of the word's postings list in the postings file. It stores this information in a dictionary data structure, with the words as keys and their associated data as values.Using the postings for a specific word, and based on the query terms and their weights, ranking is done.

The function *‘load_positngs’* retrieves the postings for a specific word from the postings file and stores the postings in a dictionary with document IDs as keys and their weights as values. Thus, using the two dictionaries mentioned above and the query terms, the function *‘rank_documents’* is executed. So, here the function iterates over each query term (which is preprocessed). For each term, it checks if it is in the dictionary. If it’s in the dictionary, then it loads the term's postings and updates the document scores by adding the term's weight multiplied by the term frequency in the document. This results in a dictionary of document scores, with document IDs as keys and their accumulated weights as values.

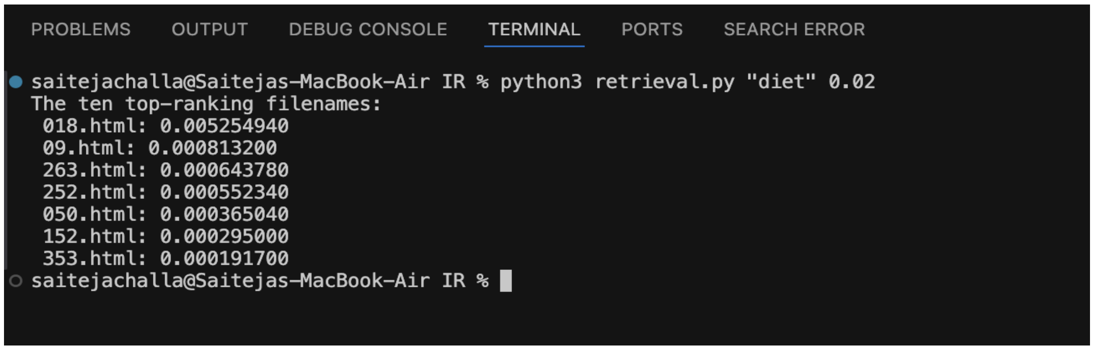
After executing the *‘rank_documents’* function, documents with zero scores are filtered out, and the documents are sorted by their scores in descending order, and then top 10 document file names and scores are printed.

<u>Data Structures used</u>:
I have mainly used dictionaries to maintain all the words with their document count and position in postings as a sub-dictionary, and to maintain the list for all the postings of the corpus. No temporary file creation is done along the process. Also, the entire dictionary, which maps terms to their respective postings information, is loaded into memory. This facilitates rapid access and lookup of term metadata during query processing. For the postings, a selective loading approach is used, only the necessary postings data for query terms are fetched from disk based on dictionary details. This strategy avoids the overhead of maintaining the entire TDM in memory, significantly reducing memory usage and enhancing efficiency.

<u>Complexity:</u>
The overall complexity of the program can be determined by looking into the process of handling query terms and their corresponding postings. To compute the cumulative scores for documents based on query terms, the primary computing step for a retrieval system based on an inverted index structure is to retrieve postings for every query term from a postings file. So, assuming we have ‘q’ query terms and each term has ‘p’ postings on average, the complexity of the program can be determined as O(q x p).

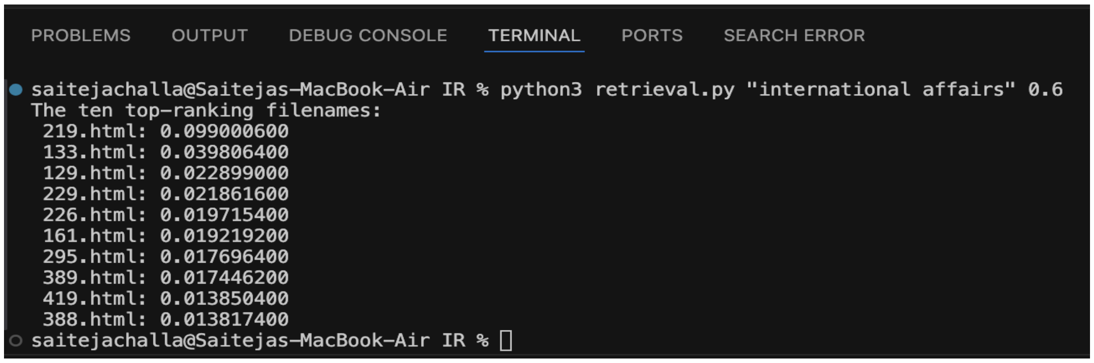
Also, the complexity of finding the relevant documents to retrieve is O (n * k), where n is the number of documents containing any query word and k is the average number of words in each document. Similarly, the complexity of finding documents containing any query term to retrieve is O (n * t), where n is the number of documents in the corpus and t is the number of query terms.

<u>Results</u>:



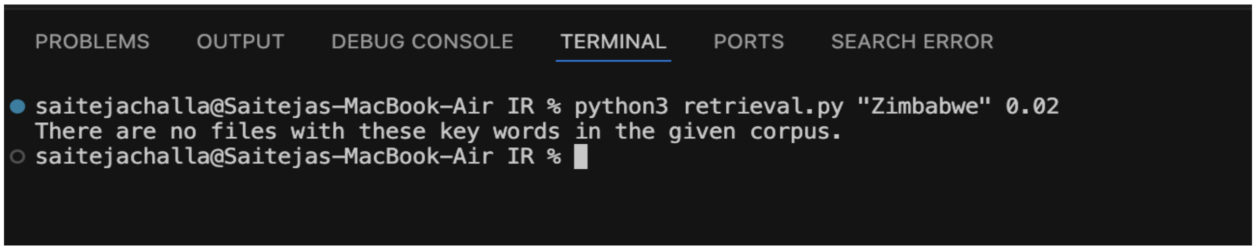
```
saitejachalla@Saitejas-MacBook-Air IR % python3 retrieval.py "diet" 0.02
The ten top-ranking filenames:
 018.html: 0.005254940
 09.html: 0.000813200
 263.html: 0.000643780
 252.html: 0.000552340
 050.html: 0.000365040
 152.html: 0.000295000
 353.html: 0.000191700
saitejachalla@Saitejas-MacBook-Air IR %
```

Output for the query term ‘diet’



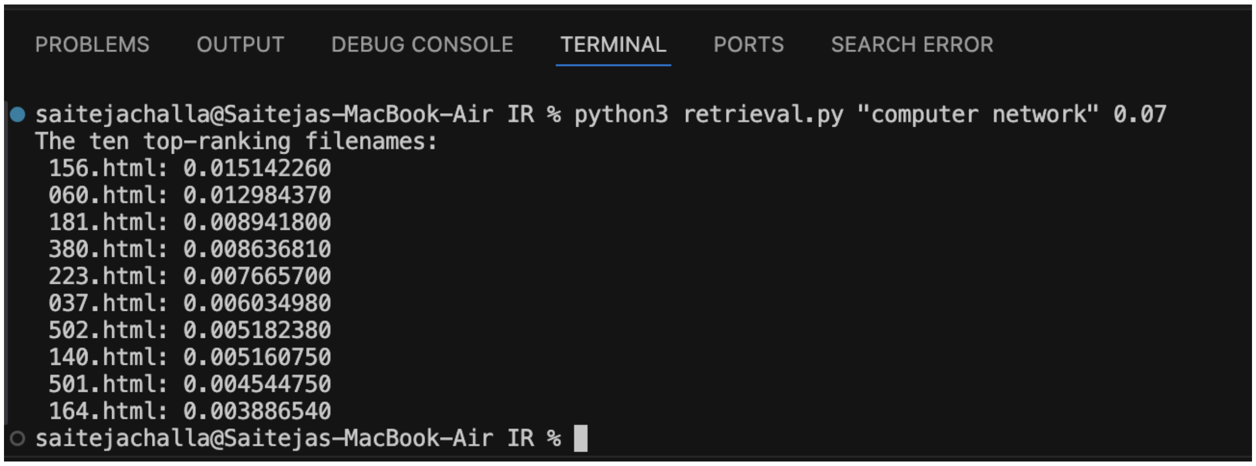
```
saitejachalla@Saitejas-MacBook-Air IR % python3 retrieval.py "international affairs" 0.6
The ten top-ranking filenames:
 219.html: 0.099000600
 133.html: 0.039806400
 129.html: 0.022899000
 229.html: 0.021861600
 226.html: 0.019715400
 161.html: 0.019219200
 295.html: 0.017696400
 389.html: 0.017446200
 419.html: 0.013850400
 388.html: 0.013817400
saitejachalla@Saitejas-MacBook-Air IR %
```

Output for the query term ‘international affairs’



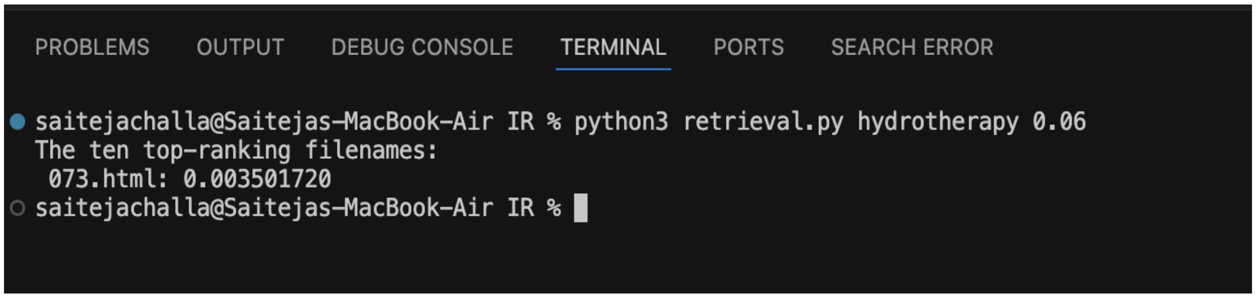
```
saitejachalla@Saitejas-MacBook-Air IR % python3 retrieval.py "Zimbabwe" 0.02
There are no files with these key words in the given corpus.
saitejachalla@Saitejas-MacBook-Air IR %
```

Output for the query term ‘Zimbabwe’



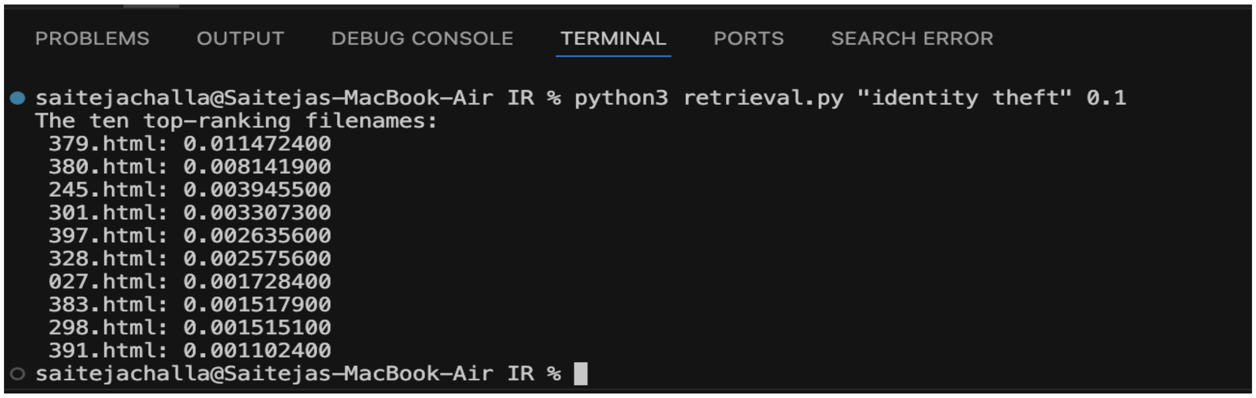
```
saitejachalla@Saitejas-MacBook-Air IR % python3 retrieval.py "computer network" 0.07
The ten top-ranking filenames:
 156.html: 0.015142260
 060.html: 0.012984370
 181.html: 0.008941800
 380.html: 0.008636810
 223.html: 0.007665700
 037.html: 0.006034980
 502.html: 0.005182380
 140.html: 0.005160750
 501.html: 0.004544750
 164.html: 0.003886540
saitejachalla@Saitejas-MacBook-Air IR %
```

Output for the query term ‘computer network’



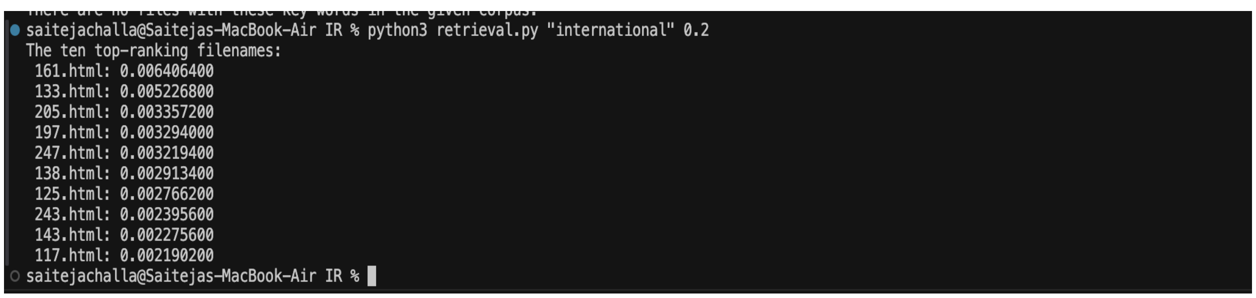
```
saitejachalla@Saitejas-MacBook-Air IR % python3 retrieval.py hydrotherapy 0.06
The ten top-ranking filenames:
 073.html: 0.003501720
saitejachalla@Saitejas-MacBook-Air IR %
```

Output for the query term ‘hydrotherapy’



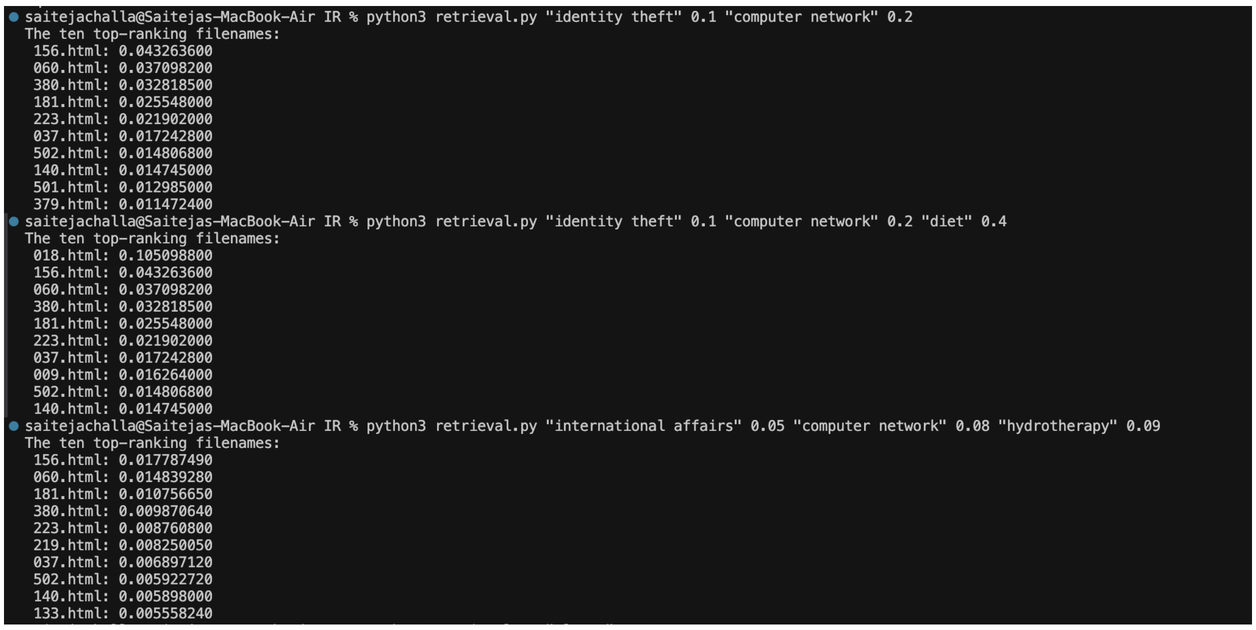
```
● saitejachalla@Saitejas-MacBook-Air IR % python3 retrieval.py "identity theft" 0.1
  The ten top-ranking filenames:
   379.html: 0.011472400
   380.html: 0.008141900
   245.html: 0.003945500
   301.html: 0.003307300
   397.html: 0.002635600
   328.html: 0.002575600
   027.html: 0.001728400
   383.html: 0.001517900
   298.html: 0.001515100
   391.html: 0.001102400
○ saitejachalla@Saitejas-MacBook-Air IR %
```

Output for the query term 'identity theft'



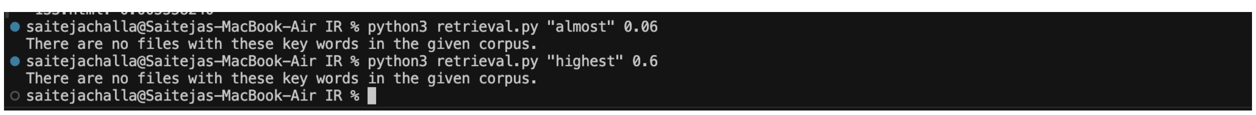
```
  There are no files with these key words in the given corpus.
● saitejachalla@Saitejas-MacBook-Air IR % python3 retrieval.py "international" 0.2
  The ten top-ranking filenames:
   161.html: 0.006406400
   133.html: 0.005226800
   205.html: 0.003357200
   197.html: 0.003294000
   247.html: 0.003219400
   138.html: 0.002913400
   125.html: 0.002766200
   243.html: 0.002395600
   143.html: 0.002275600
   117.html: 0.002190200
○ saitejachalla@Saitejas-MacBook-Air IR %
```

Output for the query term 'international'



```
● saitejachalla@Saitejas-MacBook-Air IR % python3 retrieval.py "identity theft" 0.1 "computer network" 0.2
  The ten top-ranking filenames:
   156.html: 0.043263600
   060.html: 0.037098200
   380.html: 0.032818500
   181.html: 0.025548000
   223.html: 0.021902000
   037.html: 0.017242800
   502.html: 0.014806800
   140.html: 0.014745000
   501.html: 0.012985000
   379.html: 0.011472400
● saitejachalla@Saitejas-MacBook-Air IR % python3 retrieval.py "identity theft" 0.1 "computer network" 0.2 "diet" 0.4
  The ten top-ranking filenames:
   018.html: 0.105098800
   156.html: 0.043263600
   060.html: 0.037098200
   380.html: 0.032818500
   181.html: 0.025548000
   223.html: 0.021902000
   037.html: 0.017242800
   009.html: 0.016264000
   502.html: 0.014806800
   140.html: 0.014745000
● saitejachalla@Saitejas-MacBook-Air IR % python3 retrieval.py "international affairs" 0.05 "computer network" 0.08 "hydrotherapy" 0.09
  The ten top-ranking filenames:
   156.html: 0.017787490
   060.html: 0.014839280
   181.html: 0.010756650
   380.html: 0.009870640
   223.html: 0.008760800
   219.html: 0.008250050
   037.html: 0.006897120
   502.html: 0.005922720
   140.html: 0.005898000
   133.html: 0.005558240
```

Output for multiple query terms.



```
   133.html: 0.005558240
● saitejachalla@Saitejas-MacBook-Air IR % python3 retrieval.py "almost" 0.06
  There are no files with these key words in the given corpus.
● saitejachalla@Saitejas-MacBook-Air IR % python3 retrieval.py "highest" 0.6
  There are no files with these key words in the given corpus.
○ saitejachalla@Saitejas-MacBook-Air IR %
```

Output for if the query term is a stopword.