PROGRAMMING ASSIGNMENT 4 – CMSC 676 INFORMATION RETRIEVAL – QUERY EXECUTION

Rishabh Sachdeva (UI73138, rishabs1@umbc.edu)

INSTALLATION AND PROGRAM EXECUTION:

Programming Language: Java 1.8

Prerequisite Installations:

Java 1.8,

Execution:

Program can be executed either via executable jar or manually creating class file and run.

Commands:

To compile: *javac informationRetrieval.assignment1.QueryExecution.java*

To run: *java informationRetrieval.assignment1.QueryExecution <input_query>*

Output: Top 10 documents with corresponding scores are displayed on console.

APPROACH:

Output Dictionary and Posting files from assignment 3 are used to evaluate scores of document corresponding to the query terms.

Strategy is to follow Term-at-a-Time approach. Term-at-a-time makes sense because the query doesn't have any restriction for number of words. Doc-at-a-time is not very useful when tokens in query increases.

Algorithm:

1. Query term weights are fetched (if provided in query). Maintain a map for this purpose (term VS wt).
2. Query is iterated over term by term.
3. Read the dictionary file (output from Assignment 3), stop when the query term is encountered.
4. Read the next two lines from dictionary to find term's frequency and start position in posting file. Start and end line numbers in the posting files corresponding to the query term can be easily evaluated using this information.
5. Now, read the specific lines (found in Step 4) from posting file to find the document weights.
6. Update all the relevant documents with the calculated weight taking query term factor into account.

7. Keep repeating Steps 3 to 6 for all the terms found in query.
8. Display the top 10 relevant documents.

Data Structures:

To maintain the partial document weights, using a simple array is not an efficient way because relevant documents are much less than the number of documents in corpus. Maintaining 0 weights in array for all the document does not make sense and is waste of memory.

So, I used a Hash Table is used to maintain these weights. An entry is created in this table whenever a new relevant document (relevance_score > 0) is encountered. The score is updated each time the same document is encountered while evaluating other terms in query.

Complexity:

O(q+n+N)

q = # terms in query

n = length of all lists, corresponding to each query term.

N =# docs in corpus.

Reading Dictionary and Postings:

My algorithms search for the query term in dictionary file line-by-line. Once term is found, corresponding frequency and start position in posting is read. The loop breaks once required information is fetched. Then, corresponding lines of posting file is read to find the scores. I chose this method because it does not make sense to me to fully read these files and take it to in-memory before finding information. The probability is strong that term will be found much before and there will be no need to read whole file line-by-line.

Query Term Weights (Extra Credit)

Assigning weight to Query Terms are important to give importance to terms which seems more important while finding relevant documents. To clarify, the very common ones which does not contribute any significant meaning to query should be assigned minimal weight because they should be given minimal importance while fetching the relevant documents.

In the scope of this assignment, the weights of the terms are provided in the input query itself. If the user is providing weights, then query should start with “Wt”. Sample query = “Wt 0.2 cat 0.8 mouse 0.6 dog” signifies weights of terms cat, mouse and dog as 0.2,0.8 and 0.6 respectively. To simplify, documents containing mouse and dog will be given more importance than those containing cat.

Sample Input and Output

1. Query = “International affairs”
   Output:

   ```
   TOP RESULTS
   Doc Id: 133.html Score:0.5873134881258011
   Doc Id: 226.html Score:0.5452690720558167
   Doc Id: 286.html Score:0.5278828293085098
   Doc Id: 235.html Score:0.5238257348537445
   Doc Id: 229.html Score:0.5217476189136505
   Doc Id: 232.html Score:0.5162864774465561
   Doc Id: 331.html Score:0.4992508888244629
   Doc Id: 242.html Score:0.4906625896692276
   Doc Id: 419.html Score:0.4892690181732178
   Doc Id: 426.html Score:0.4831032156944275
   ```

2. Query = “Zimbabwe”
   Output:
   This Search Engine has no relevant documents for query provided.

3. Query = “Computer network”

   ```
   TOP RESULTS
   Doc Id: 223.html Score:0.7011393904685974
   Doc Id: 164.html Score:0.6567123234272003
   Doc Id: 156.html Score:0.645739734172821
   Doc Id: 145.html Score:0.6296834051609039
   Doc Id: 64.html Score:0.6290841400623322
   Doc Id: 22.html Score:0.6266733109951019
   Doc Id: 47.html Score:0.5834581851959229
   Doc Id: 290.html Score:0.556940495967865
   Doc Id: 27.html Score:0.5488316714763641
   Doc Id: 388.html Score:0.5225338041782379
   ```

4. Query = ”hydrotherapy”

   ```
   TOP RESULTS

   Doc Id: 273.html Score:0.8039596676826477
   ```

5. Query = “identity theft”

```
TOP RESULTS
Doc Id: 379.html Score:0.7563017010688782
Doc Id: 380.html Score:0.7241280674934387
Doc Id: 292.html Score:0.6045958399772644
Doc Id: 301.html Score:0.45240241289138794
Doc Id: 328.html Score:0.44610345363616943
Doc Id: 245.html Score:0.44253939390182495
Doc Id: 332.html Score:0.39577779173851013
Doc Id: 298.html Score:0.3905530571937561
Doc Id: 397.html Score:0.38421863317489624
Doc Id: 27.html Score:0.37343406677246094
```

6. Query = “diet”

```
TOP RESULTS
Doc Id: 18.html Score:0.6788696646690369
Doc Id: 252.html Score:0.5742202401161194
Doc Id: 263.html Score:0.5536826848983765
Doc Id: 9.html Score:0.5402939319610596
Doc Id: 50.html Score:0.5234379172325134
Doc Id: 152.html Score:0.413370281457901
Doc Id: 353.html Score:0.2924499809741974
```

8. Query = “sick building syndrome”
Output:

```
TOP RESULTS
Doc Id: 99.html Score:1.5978300273418427
Doc Id: 260.html Score:0.9922026693820953
Doc Id: 383.html Score:0.986698716878891
Doc Id: 118.html Score:0.866781085729599
Doc Id: 230.html Score:0.8100775480270386
Doc Id: 382.html Score:0.6681161969900131
Doc Id: 271.html Score:0.5184950232505798
Doc Id: 267.html Score:0.47687268257141113
Doc Id: 266.html Score:0.46292853355407715
Doc Id: 394.html Score:0.45964744687080383
```

In this example, word “building” is relatively common. “Building” occurs in 46 documents of the corpus and “syndrome” in 6. So, the “syndrome” word is most important and should be highly weighted, and “building” should be least weighted.

After applying weights to the query terms, results change. A few new documents like (272.html) are included in the results because of presence of relatively important term (syndrome). Also, ranks of documents(394.html, 266.html) gets better because of same logic.

9. Query = “wt 0.7 sick 0.2 building 0.9 syndrome” [first string is wt to indicate that weights are present in the query]

```
TOP RESULTS
Doc Id: 99.html Score:1.0834395170211792
Doc Id: 260.html Score:0.8064916223287583
Doc Id: 118.html Score:0.6923261612653733
Doc Id: 383.html Score:0.6704363733530045
Doc Id: 271.html Score:0.46664552092552186
Doc Id: 230.html Score:0.43233315348625184
Doc Id: 266.html Score:0.4166356801986694
Doc Id: 394.html Score:0.41368270218372344
Doc Id: 272.html Score:0.40479013323783875
Doc Id: 323.html Score:0.3957401722669602
```

10. Query = “presidential candidate”

```
TOP RESULTS
Doc Id: 234.html Score:0.7476955950260162
Doc Id: 333.html Score:0.7370777726173401
Doc Id: 1.html Score:0.6759956479072571
Doc Id: 331.html Score:0.6171174645423889
Doc Id: 339.html Score:0.5510006099939346
Doc Id: 377.html Score:0.4931376427412033
Doc Id: 349.html Score:0.4660084843635559
Doc Id: 340.html Score:0.43652792274951935
Doc Id: 364.html Score:0.4318704903125763
Doc Id: 338.html Score:0.4235900193452835
```

11. Query = “wt 0.8 presidential 0.2 candidate”

```
TOP RESULTS
Doc: 234.html Score: 0.3604631364345551
Doc: 333.html Score: 0.35033704042434693
Doc: 1.html Score: 0.3007718026638031
Doc: 41.html Score: 0.2999309539794922
Doc: 331.html Score: 0.29525538682937624
Doc: 419.html Score: 0.2700530052185059
Doc: 426.html Score: 0.26782729625701907
Doc: 65.html Score: 0.2589663743972778
Doc: 81.html Score: 0.25853164196014405
Doc: 73.html Score: 0.2577741384506226
```