Assignment 2: Ranked Retrieval

Points: 75

Submission deadline: Friday, 02/17/17, 11.59 PM

Note: This is individual work.

In this assignment, you will improve on the indexes from assignment 1, and implement the vector space model with cosine similarity for retrieving the top K documents (ranked retrieval) from the collection of document provided as an attachment. You will then compare exact Top K search with methods of inexact top K search. The specifications are:

Part A: Exact Top K Retrieval [20]

- In part A, you will construct a retrieval system for exact top K retrieval (henceforth denoted as Method 1).
- Use the TF-IDF weighting for the terms to construct the vectors for the documents and queries.
- Your index will be of the form
    - ***Term ID: [idf_t,(ID1,w_{t_1,d_1}, [pos1,pos2,..]), (ID2, w_{t_2,d_2}, [pos1,pos2,…]),….]***
- As in assignment 1, your program will accept the collection, process the documents and construct the index. Additionally, you will use the stop list (provided as an attachment) to weed out words from your dictionary. Note you should not index the document for each query. Simply do this at the beginning.
- Your program will then accept a free text query, generate the vectors for the documents and the query and compute the cosine similarity score for the documents. You can assume that you will not get single term queries.
- The program will retrieve and display the names of the top K documents for each query in decreasing order of their score.
- For each query, you will consider the results for your top K results as the *baseline results*. You will use this in part B
- You will also note the time taken to retrieve the results for each query.

Part B: Comparison of Inexact retrieval methods with inexact retrieval [30]

- You will also compare the performance of the exact top K with three inexact methods given below.
- Method 2 Champion List: Implement a champion list method which will produce, for each term, a list of r documents that is based on the weighted term frequency $\boldsymbol{w_{t,d}}$. Come up with a formula for *r*. Discuss this in your report. Remember the champion list is created only once for a collection.
- Method 3 Index Elimination: Implement index elimination by using only half the queries terms sorted in decreasing order of their IDF values.
- Method 4 Simple Cluster pruning: You will randomly pick $\sqrt{N}$ leaders (where N is the number of documents in the collection) and then use them to implement the cluster pruning. Note that you need to select the leaders only once. For each query, you will select a leader closest to the

query and then retrieve the top K results. Remember, if you do not get the top K results with the first leader, you will also look at the next best leader and so on.

Part C: Comparison of Inexact retrieval methods with inexact retrieval [25]

Experimentally compare the performance of your implementations of the exact retrieval with that of inexact retrieval methods. What are your performance measures? How are they measured and compared? In your report, you will provide justifications for these measures. Use graphs, as appropriate, to compare performance and describe insights and conclusion drawn from your results. Also include details for your implementation in the report that may have an impact on performance.

Other instructions:

- Implement in Python 3.0
- Comment your code appropriately
- You may reuse the code from earlier assignment

Attachments:

1. Collection of documents

2. List of 25 stop words

3. Skeleton code – implement the functions in the code. Use additional functions as needed.

Submission:

Submit the following files on blackboard as a .zip file.

1. index.py

2. Report.doc: with performance analysis and other discussions.

3. Output.txt: containing 5 queries and the output generated by your code. This is for testing your code.