Assignment 3

This is an individual assignment. If you get help from others you must write their names down on your submission and explain how they helped you. If you use external resources you must mention them explicitly. You may use third party libraries but you need to cite them, too. The total credits for this assignment is 110'.

Date posted: March 24th, 2016

Date Due: April 6th, 2016

Goal: Retrieval Models; Lucene; Evaluation

Task1: Building your own retrieval models (30')

A. Data collection, text processing and building Index

A.1 Downloading the data collection

Download the **CACM data collection** from *Search Engines: Information Retrieval in Practice test collection site*. Please download the .tar.gz file version. The .corpus version is in a special format meant for use by the (book's) Galago search engine. The entire corpus contains 3,204 documents. You may notice that coming together with the document data, there are files for raw query (containing 64 queries), processed query (63 processed queries), and the relevance judgments for each query. In the relevance judgments file, 0 indicates non-relevant and 1 indicates relevant. Q0 can be simply ignored. Please roughly check them before your processing and retrieval.

CACM is a collection of abstracts of articles published in the Communications of ACM journal between 1958 and 1979. In many cases, documents just contain the titles of the abstracts. You can ignore the columns of numbers, which is an encoding of bibliographic references.

A.2 Stop-wording

Before creating the index, please remove the stop words from the attached list of words that is the default stop words set in StandardAnalyzer and EnglishAnalyzer (StopAnalyzer.ENGLISH_STOP_WORDS_SET) used in Lucene.

A.3 Building Index

Build an Inverted Index in the form of word → (docid1, tf1), (docid2, tf2) … (docidN, tfN). Please sort word and docid both in alphabetically ascending order. Docid for CACM document with name *CACM-0015.html* is CACM-0015.

There is no particular requirement in terms of the Index output format. You can use the separator as SPACE, TAB or any other special characters. Please specify your Index output format in the final submitted README.txt file.

***B.* TF-IDF Retrieval Model**

Apply the first TF-IDF Retrieval Model (in Assignment 1-B Task 2.A) for the CACM data set over the queries listed in Task 3.A. You are expected to output the Top 100 documents for one query, however, the ranking results (not just Top 100) for all queries will be used to do evaluation in Task 3, and therefore you should keep record of them in a certain way.

***C.* BM25 Retrieval Model**

Apply the BM25 Retrieval Model (in Assignment 1-B Task 2.C) for the CACM data set over the queries listed in Task 3.A. You are expected to output the Top 100 documents for one query, however, the ranking results (not just Top 100) for all queries will be used to do evaluation, and therefore you should keep record of them in a certain way.

Task 2: Using Lucene (40')

Download and install Lucene.4.7.2, and also download the very original HW3.java file. This java file is a simple yet workable demo that can help you build the inverted index for a given input data set, implement the search and return to you a ranking list based on your input query. Please read this HW3.java file carefully, compile it and make it runnable for a given query with only one term *"parallel"* as an initial trial. The index built by this Lucene demo does not conduct stemming, but it does remove the stopping words.

Please then modify the program so that it can do the following things:

A. Being able to return the Top 100 documents, and/or the entire 3,204 documents. 3,204 is the ideal maximum number of documents that can be retrieved, however, for most queries, only a subset (greater than 100 though) of all the documents can be retrieved and returned. (This is an easy modification).

B. Being able to output the *docFrequency* and overall *termFrequency* for each term in the query. (This is an easy modification)

C. Being able to output for each indexed term its term frequency across all documents. (i.e. the total number of term frequency across all documents). ***Hint:*** you may have to use the class of *TermsEnum*, *MultiFields*, and *BytesRef*.

D. Being able to output for each indexed term its document-based term frequency (i.e. the term frequency of a tem in each document). ***Hint:*** you may have to use the class of *TermsEnum*, *MultiFields*, *BytesRef* and *DocsEnum*.

E. By default, Lucene' ranking algorithm uses a combination of the Vector Space Model based on TF-IDF weighting and the Boolean Model. Please modify it so that the ranking algorithm will become BM25. ***Hint:*** you may have to use the BM25Similarity class.

You can seek help from Lucene 4.7.2 API and any online resources to make the above modification.

Task 3. Evaluation (40')

A. Testing Queries

Please work on the following 5 queries using your own retrieval models and the Lucene provided models.

QueryID	**QueryTerm**
1	time sharing system
7	distributed algorithms
9	network operating systems
10	parallel computation
14	database management applications

You may notice that the original queries as shown in cacm.query.txt or cacm.query.xml file are comparatively long. We manually created the short-version of some queries for testing. The QueryID is the original QueryID in cacm.query.txt/cacm.query.xml and cacm.rel.txt (the relevance judgment file).

B. Comparison

Since we are now provided with relevance judgments for each query, we can conduct effective evaluation and comparison in terms of the ranking performance among multiple models. Please compare the ranking performance among the following four retrieval models:

(1) Your own TF-IDF retrieval model

(2) Your own BM25 model

(3) Lucene-based TF-IDF model

(4) Lucene-based BM25 model

Over the following five measurements:

(1) Precision for each query and averaged Precision across all 5 queries
(2) Recall for each query and averaged Recall across all 5 queries
(3) F_1-score for each query and averaged F_1-score across all 5 queries
(4) Averaged Precision@N (N=5, 10, 20, 30, 50, 70, 100)
(5) MAP (Mean Average Precision)

What to hand in:

1) A README.txt file on how to run your codes and/or any specification on your implementation and output.
2) The Index you built for CACM data collection.
3) Your modified HW3.java for Lucene.
4) A file displaying each indexed term and its overall term frequency you can obtain using Lucene. Please output each term per line and separate the term and term frequency by a space.
5) A file displaying each indexed term and the term frequency in each document you can obtain using Lucene. The output is in the following format.

   Term docid1:tf1 docid2:tf2 … docidN:tfN

6) Your codes for implementing the 5 measurements: Precision, Recall, F_1-score, Precision@N (N can be a parameter), and MAP.
7) The Top 100 ranking results for QueryID=9 for four different retrieval models. Please display the output as follows:

   Rank DocID Score

   Please output the results for each model separately. DocID is the CACM docID.

8) Five Tables for evaluation on Precision, Recall, F_1-score and MAP. Please display your table in the following format.

The results for each individual model

Model1/2/3/4	Q1	Q7	Q9	Q10	Q14
Precision					
Recall					
F_1-score					
AP					

Where: AP indicates Average Precision.

Model1 -- your TF-IDF model; Model2 – your BM25 model;

Model3 – Lucence TF-IDF; Model4 – Lucene BM25.

And a comparison between them

	Model1	Model2	Model3	Model4
Averaged Precision				
Averaged Recall				
Averaged F_1-score				
MAP				

(9) A figure showing **four curves** for the averaged Precision@N for four models when the value of N changes from 5, 10, 20, 30, 50, 70 and 100. X-axis is the changeable N value, and Y-axis is the Precision@N value.

(10) A brief analysis on the evaluation results and comparisons among four models.