Assignment 4:

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.

Date posted: Sunday November 19, 2017

Date Due: Wednesday November 29, 2017 11:59pm

Goal: Introduction to Lucene. Retrieval and scoring using BM25.

Description:

Task 1: For this task, you need to download and setup Lucene https://lucene.apache.org/, an open source information retrieval library. Lucene is widely used in both academic and commercial search engine applications. Solr and ElasticSearch are both based on the Lucene libraries.

In order to make things easier for you, a starter program is provided (see attachment) to create your Lucene-based search engine. This code is based on Version 4.7.2 (see attached files, or go to https://archive.apache.org/dist/lucene/java/4.7.2/) and is written in Java. However, it is up to you to choose the implementation of your preference.

Once you download Lucene, the setup is pretty straightforward. You need to create a new Java project and add the following three jars into your project's list of referenced libraries:

1) lucene-core-VERSION.jar
2) lucene-queryparser-VERSION.jar
3) lucene-analyzers-common-VERSION.jar.

Where VERSION is to be substituted with the version of Lucene that you downloaded. For example, in the provided example, we have version 4.7.2, therefore, the first jar file would be lucene-core-4.7.2.jar. Make sure that the system requirements for that version are met.

You will need to go through Lucene's documentation (and the provided code) to perform the following:

1- Index the raw documents from HW3 Task 1. Make sure to use "SimpleAnalyzer" as your analyzer.
2- Perform search for all the queries provided in Task 2. You need to return the top 100 results for each query. Use the default document scoring/ranking function provided by Lucene.

Task 2: In HW3 Task 2, you have built your simple indexer and used it to generate different indexes. In this assignment, you will finish building your search engine by building the retrieval module. Implement the BM25 ranking algorithm, and write a program to provide a ranked list of documents for a file with one or more queries. Use the unigram index generated in HW3. Show the top 100 retrieved document IDs and their BM25 scores for each query according to the following format:
query_id Q0 doc_id rank BM25_score system_name

The string Q0 is a fixed literal used by the standard TREC evaluation script. Sort these results by score starting with the highest topical relevance score. Do not use spaces in the *system_name*.

How to perform retrieval:

1- Fetch all inverted lists corresponding to terms in a query.
2- Compute BM25 scores for documents in the lists. Make a score list for documents in the inverted lists.
3- Accumulate scores for each term in a query on the score list.
4- Assume that no relevance information is available.
5- For parameters, use k1=1.2, b=0.75, k2=100.
6- Sort the documents by the BM25 scores.

Use the following stopped, case-folded test queries:

Query ID	Query Text
1	hurricane isabel damage
2	forecast models
3	green energy canada
4	heavy rains
5	hurricane music lyrics
6	accumulated snow
7	snow accumulation
8	massive blizzards blizzard
9	new york city subway

What to hand in:

1- A readme.txt file with instructions to compile and run your programs for both tasks
2- Your source code for both tasks (both indexing and retrieval modules for Task 1)
3- A very short report describing your implementation.
4- 18 tables (one per query x two IR systems) each containing at MOST 100 docIDs ranked by score.
5- A brief discussion comparing the top 5 results between the two search engines for each query.