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: Saturday November 12, 2016

Date Due: Sunday November 20, 2016 11:59pm

Goal: Introduction to Lucene. Retrieval and scoring using Cosine Similarity.

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, we are providing you with a basic program (see attachment) that can serve as a starting point 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 Assignment 1 Part B. Make sure to use "SimpleAnalyzer" as your analyzer.
2- Perform search for the four 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, you have built your simple indexer. In this assignment, you will finish building your search engine by building the retrieval module. You will continue to use your "Sustainable Energy" Wikipedia corpus. Implement the Vectors Space Cosine Similarity ranking algorithm, and write a program to provide a ranked list of documents for a file with one or more queries. Submit the output from this run, with the top 100 document IDs and their similarity scores for each test query according to the following format:
query_id Q0 doc_id rank CosineSim_score system_name

The string Q0 is a literal used by the standard TREC evaluation script. You can use any space-free token for your system_name.

Perform the following:

1- Retrieve all inverted lists corresponding to terms in a query.
2- Compute Cosine Similarity 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- Sort the documents according to the scores.

Test Queries: Use the following stopped, case-folded test queries:

Query ID	Query Text
1	global warming potential
2	green power renewable energy
3	solar energy california
4	light bulb bulbs alternative alternatives

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 2)
3- A very short report describing your implementation.
4- Eight tables (one per query, for each search engine) each containing at MOST 100 docIDs ranked by score.
5- A brief discussion comparing the top 5 results between the two search engines across all four queries.