College of Computing and Information Science — CS6200
Northeastern University — Information Retrieval
Fall 2018 — November 10th, 2018

Homework Assignment #4
Due Date: November 19th, 2018 @ 11:59pm

Instructions:

- The assignment is due on the time and date specified.
- This is an individual assignment. You may discuss the problems with your friends, but the code, analysis, interpretation and write-up that you submit for evaluation should be entirely your own.
- You are encouraged to use the Piazza discussion board, and seek help from TAs and instructors to get clarifications on the problems posed.
- If you receive 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.

Goal: Introduction to Lucene. Retrieval and scoring using Language Models.

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.

You can download the latest version of Lucene (7.5.0) from:
http://lucene.apache.org/core/
The documentation is linked here:
http://lucene.apache.org/core/documentation.html).
We provide a starter program to help you get started with the essential components needed to create your first 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. Most of the code should work but some method signatures may have changed in more recent implementations (see documentation). It is up to you to choose the implementation and/or version 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 of your Wikipedia corpus. Make sure to use "StandardAnalyzer" 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, 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 *Language Model with Dirichlet Smoothing (LM Dirichlet)* 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. Use the word unigram index (no proximity or compression needed). Show the top 100 retrieved document IDs and their LM Dirichlet scores for each query according to the following format:
query_id Q0 doc_id rank LMDirichlet_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 LM Dirichlet 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- Use μ=1500.
5- Sort the documents based on scores (highest score at first rank).

Use the following stopped, case-folded test queries:

Query ID	Query Text
1	carbon emission
2	cutting carbon emission
3	flexible dieting
4	flexible dieting and vegetarianism
5	information on pest control in greenhouse
6	pest control greenhouse
7	greenhouse apples apple
8	green house apples apple

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- 16 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.
6- A brief discussion of results obtained for each pair of consecutive queries (i.e. queries 5 vs. 6) in terms of document overlap, result ordering, score magnitude, and other remarkable observations (if any).