Department of Computer Science and Software Engineering, Concordia University

COMP 479 (6791) Project 2

Due Date: November 8th at midnight
Objectives: Build a probabilistic search engine out of your indexer.

Description:

1. Consider the assumptions made in chapter 11 of your textbook about independence of terms and documents etc.
2. Implement rank retrieval based on term frequencies alone.
3. Implement rank retrieval using Okapi BM25 (probabilistic ranking).
4. Experiment with the value of k (see the textbook -- equation 86, or wikipedia). Keep b at 0.75, as we do not ask you to discuss it.
5. Build a system allowing to perform queries on the Reuters corpus and rank the results
6. You can re-use the code from project 1 but note that we do *not* care about memory usage at this point.
7. Your system should be able to display the retrieved documents

Note: if for any reason you did not complete project 1 or your implementation is unusable, you are allowed to use someone else's code or libraries available online, so long as you make this clear in your report. You should, of course, implement the new parts yourself.

Deliverables:

1. A report describing your system (inverted index data structure, high level description of the two implementations with file names and line numbers to where we can see the implementation)
2. A discussion of the differences observed between the two ranked retrieval implementations. Tell us how the output is influenced in terms of the top 5 documents retrieved and their ordering. To do so, use the following queries:
   "Democrats welfare and healthcare reform policies"
   "Drug company bankruptcies"
   "Dow jones great depression"
3. A discussion on the influence of the value of k
4. Any additional testing or aborted design ideas that show off particular aspects of your project.
5. Well documented code

The report should be around 4 pages long

Mark Breakdown:

- 50% Report
  - Data structure and architecture description (5%)
  - Discussion on the influence of k (15%)
  - Discussion about the differences in querying using term frequency scoring and BM25 (30%)
- 25% Implementation
  - Term Frequency Weighting (10%)
  - BM25 (15%)
- 25% Demo
  - Proper output (10%)
  - Can argue with the TA about the advantages of each method (15%)