CS47300 Homework 4

Due date: Tuesday November 22, 11:59pm (submit via Turnin)

Homework must be submitted using turnin. *Instructions below detail how to turn in your code and assignment on data.cs.purdue.edu.*

Overview

In this assignment you will implement different retrieval scoring functions in Galago and evaluate their impact on retrieval performance. You will use the inverted indexes and corpus files from HW3.

1 Retrieval functions (24 pts)

Write a class for each score function below that implements the interface `ScoreIterator` (which extends `BaseIterator`). You may also consider the interface `ScoreCombinationIterator`. Your classes should compute the specified retrieval function for each document (given a query) and return the relevance score.

1. TF (term frequency):
   $score(q, d) = \sum_{w \in q} freq(w, d)$
   Here $freq(w, d)$ refers to the frequency of word w in document d.

2. TFIDF (term frequency/inverse document frequency):
   $score(q, d) = \sum_{w \in q} freq(w, d) \cdot log \frac{N}{n_w}$
   Here n_w is the document frequency of w in the whole corpus. N is the total number of documents in the corpus.

3. logTFIDF (log TFIDF):
   $score(q, d) = \sum_{w \in q} log[freq(w, d) + 1] \cdot log \frac{N}{n_w}$
   Here n_w is the document frequency of w in the whole corpus. N is the total number of documents in the corpus.

4. COSINE (vector space model, cosine similarity):
   $score(q, d) = \frac{\sum_{w \in q} freq(w,q) \cdot freq(w,d)}{\sqrt{\sum_{w \in q} freq(w,q)^2 \cdot freq(w,d)^2}}$
   Here $freq(w, q)$ refers to the frequency of word w in document q.

Implement each of the above models as a separate class respectively named: `TFScoringIterator`, `TFIDFScoringIterator`, `logTFIDFScoringIterator`, `CosineScoringIterator`.
Add your classes to: `org.lemurproject.galago.core.retrieval.iterator.scoring`.

Add the class names to the list of operators in `FeatureFactory` (use the names `tf`, `tfidf`, `logtfidf`, and `cosine` respectively). Then compile so the operators are available during querying with Galago.

2 Evaluation (16 pts)

Use the `cacm` corpus and index from HW3.

For each of your four score functions from Q1, as well as BM25 and Jelinek-Mercer (which are implemented in Galago), do the following:

(a) Construct a json query file consisting of the first 25 `cacm` queries.

(b) Use the specified score function with the Galago `batch-search` command to query and output the top 10 documents (per query) to a file to use below.

(c) Use the Galago `eval` command to evaluate the results against the relevance judgements. Report mean average precision, NDCG, and precision@3 averaged over the 25 queries.

Discuss the differences of the scoring functions (ie. strengths/weaknesses) in terms of their search effectiveness based on your experimental results.

Submission Instructions:

After logging into data.cs.purdue.edu, please follow these steps to submit your assignment:

1. Make a directory named `yourName_yourSurname` and copy all of your files there.
2. While in the upper level directory (if the files are in /homes/jihwan/jihwan_lee, go to /homes/jihwan), execute the following command:

   ```
   turnin -c cs473 -p HW4 yourName_yourSurname
   ```

   For example, your TA would use: turnin -c cs473 -p HW4 jihwan_lee to submit his work. Keep in mind that old submissions are overwritten with new ones whenever you execute this command.

   You can verify the contents of your submission by executing the following command:

   ```
   turnin -v -c cs473 -p HW4
   ```

   Do not forget the -v flag here, as otherwise your submission would be replaced with an empty one.

Your submission should include the following files:

1. Java class files for each of the score functions (use class names specified in Q1).
2. Jar file named "`core-3.10`" compiled with your new classes, and updated `FeatureFactory`, from Q1.
3. Report in **.pdf** format containing your answers to Q2.
4. (Optional) A README file containing anything you would like us to know about your code (like errors, special conditions etc.).