Information Retrieval

SS2014

Exercise 2 - Group 19

Our task in this exercise builds upon our experiments and experiences of the previous exercise. We were asked to integrate in our project Lucene functionality and to compare it with our results from the previous exercise.

Lucene (https://lucene.apache.org/) is an open source information retrieval software library developed (and supported and released) by the Apache Software Foundation. It is originally developed in Java, so the integration in our project was not a problem.

For our experiments we used the same 20 Newsgroups Subset document collection provided to us for the first exercise, and also the same 20 search topics for testing. We used the default Lucene implementation to index the documents and search them. For comparison reasons, we used our indexing from the first exercise, the Lucene default Similarity, and the Lucene BM25. Moreover, we implemented a BM25L Similarity score calculator, as described in the paper “When Documents Are Very Long, BM25 Fails!” published by Yuanhua Lv and ChengXiang Zhai (<http://sifaka.cs.uiuc.edu/~ylv2/pub/sigir11-bm25l.pdf>) and used this one also for our experiments.

The BM25 retrieval function has been the state of the art for nearly decades. BM25 scores a Document *D* with respect to Query *Q* as follows:

where *c(q,D)* is the count of *q* in *Q*, *N* is the total number of documents, *df(q)* is the document frequency of *q*, and is a parameter. The term frequency is calculated by the following formula (TF normalization formula):

where represents document length, stands for average document length, *c(q,D)* is the raw Term Frequency of *q* in *D*, and *b* and are two parameters. is the normalized Term Frequency by document length using pivoted length normalization:

As is shown in the above article, these formulas perform very well in the general case, but tend to penalize very long documents. So it introduces an improved TF normalization formula:

This equation still satisfies the constraints for BM25, but performs better with very long documents by adding a shift parameter. By substituting into the first equation we obtain a new retrieval function BM25L.

For our implementation we choose the following values for the parameters: we set , as it is proposed in the paper, (paper proposed a value between 0.2 and 3.0), (paper proposed a value between 0.1 and 0.9) and finally . These values are hard coded in our implementation, with the exception of , which is set in the Class constructor, as it was asked for this exercise.

Our implementation is once again in Java, and it includes the same source code with exercise 1, adjusted for the purpose of the second assignment, with additional functionality for Lucene, including the external Lucene jars. It is still a command line executable Java program. In addition to our initial commands (!info, !setPath, !bigram, !stemmer, !stopwords, !buildVoc, !search) from exercise 1, we added the following CLI commands:

**!luceneBuildVoc** - builds our vocabulary with Lucene

**!enableLucene** - enables Lucene functionality

**!enableBM25L** - enables our implementation of BM25 Similarity

**!lucSearch** - searches the documents with Lucene

For each of the 20 given topics, the result appears in a ranked list, with each line having the following format:

*topic Q0 document-id rank score run-name*

where

**topic** is ‘’topic# with # the number of the given topic

**document-id** is an identifier for the document given by us and containing the

parent folder in the documents subset and the file name.

**rank** is an integer holding the rank of the object in the list

**score** is the similarity score result calculated by Lucene

**run-name** a name given from us to our experiment

For our evaluation we used the provided program *trec\_eval*. This is a small C application, which calculates the Mean Average Precision of our result lists over the 20 topics. The program offers many command parameters (about evaluation, different measures used in the result calculation). We choose to use the default parameters. The results are printed in a table, which represent the results from our indexer (from exercise 1), the Lucene default Similarity, the Lucene BM25 Similarity and our implementation of the BM25L Similarity over each topic and the average results.

You can see all results from our experiments in the following table. There are no results for the topics 8, 12 and 20, as these topics have no query relevance. Our results from BM25L are overly better than the ones from the first exercise, and on some topics (4, 6, 10, 17) comparable with the ones from Lucene BM25. We see also that the Lucene BM25 performs overly better than the default Lucene score, proving that this method is clearly better and therefore considered the state of the art for the last two decades.

In this exercise we had the opportunity to use a free library for Information Retrieval, developed over the years from dedicated people in this field, extend its functionality and finally compare the overall results, including the ones from our own implementation. Although far from a real world example, it gave us an inside view in a case of text Retrieval.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Topic | Default Lucene | BM25 | BM25L | TFIDF |
| 1 | 0.1187 | 0.1535 | 0.0475 | 0.0235 |
| 2 | 0.2539 | 0.3383 | 0.0961 | 0.0008 |
| 3 | 0.1966 | 0.3245 | 0.0324 | 0.0167 |
| 4 | 0.5000 | 0.5000 | 0.5000 | 0.0000 |
| 5 | 0.2993 | 0.5000 | 0.0111 | 0.0000 |
| 6 | 0.5000 | 0.5000 | 0.5000 | 0.0000 |
| 7 | 0.2408 | 0.3395 | 0.1332 | 0.0022 |
| 8 | X | X | X | X |
| 9 | 0.5556 | 0.6429 |  | 0.0000 |
| 10 | 0.5000 | 0.5152 | 0.5000 | 0.0000 |
| 11 | 0.1597 | 0.2064 | 0.0053 | 0.0602 |
| 12 | X | X | X | X |
| 13 | 0.2518 | 0.4643 | 0.0746 | 0.0008 |
| 14 | 0.2715 | 0.4376 | 0.0932 | 0.0025 |
| 15 | 0.0523 | 0.0640 | 0.0325 | 0.0000 |
| 16 | 0.4750 | 0.4750 | 0.3080 | 0.0000 |
| 17 | 0.2080 | 0.2667 | 0.2000 | 0.0000 |
| 18 | 0.4247 | 0.4141 | 0.2776 | 0.0556 |
| 19 | 0.4286 | 0.6667 | 0.3475 | 0.0000 |
| 20 | X | X | X | x |
| All | 0.3198 |  |  | 0.0095 |