Evaluation of the Default Similarity Function in Lucene

Hui Fang and ChengXiang Zhai
Updated: August 20, 2009

1 Introduction

Lucene [4, 3] is a popular open-source IR toolkit, which has been widely used in many search-related applications [5]. However, there was no study on evaluating the retrieval performance of the default retrieval function that is implemented in Lucene. Clearly, an improved retrieval function would enable all the applications based on Lucene such as Nutch to achieve higher search accuracy. Thus it would be interesting to perform a quantitative evaluation of the retrieval function implemented in Lucene to see how well it perform relative compared with one of the state of the art retrieval functions.

In this report, we evaluate the default retrieval function of Lucene over three representative evaluation collections [6], and compare it with a state-of-the-art retrieval function, i.e., F2-EXP axiomatic retrieval function, which was proposed in [2]. Experiments show that the retrieval performance of the default function is worse than axiomatic retrieval function, suggesting that the axiomatic retrieval function is a good alternative retrieval function that could be implemented in Lucene.

2 Evaluation Methodology for Retrieval Functions

Intuitively, a better retrieval function should be able to return more relevant documents to users and rank them at top. To quantitatively evaluate the ranking list of a retrieval function, we can use Cranfield evaluation, which is the major evaluation methodology in information retrieval. The evaluation needs to be conducted based on test collections. Every test collection contains a set of documents, a set of topics (i.e., queries) and the relevance judgements (i.e., labels that indicate whether a document is relevant to a query). Many evaluation collections have been constructed in TREC conferences [6].

Two basic evaluation measures are *precision* and *recall*. The precision is the percentage of the returned documents that are relevant. The recall is the percentage of the relevant documents that are returned. But both measures are based on the set of documents returned by a retrieval function.

$$Precision = \frac{num\ of\ returned\ relevant\ documents}{num\ of\ returned\ documents}$$

$$Recall = \frac{num\ of\ returned\ relevant\ documents}{num\ of\ relevant\ documents}$$

To evaluate the overall ranking of a document list, a commonly used measure is Mean Average Precision, i.e., MAP. For every query, Average Precision (AP) is an average of the precision value obtained after every relevant document is returned. If a relevant document is not returned, the corresponding precision is 0. MAP is an average of the AP for all the queries in the test collection. A higher MAP value means that the retrieval function can retrieve relevant documents more quickly and completely.

$$AP = \frac{\sum(precision\ value\ obtained\ after\ every\ relevant\ document\ is\ retrieved)}{num\ of\ relevant\ documents}$$
$$MAP = \frac{\sum(AP\ of\ every\ query)}{num\ of\ queries}$$

3 Retrieval Functions

3.1 the Default Similarity Function in Lucene

The default similary function in Lucene is derived from vector space model [1], where both document D and query Q are assumed to be vectors of terms. Higher similarity betwteen these two vectors means that the document is more relevantn to the query. The scoring function is shown as follows.

$$S(Q,D) = \frac{|Q \cap D|}{|Q|} \times \frac{1}{\sum_{t \in Q}(1 + log\frac{N}{df(t)+1})^2} \times \sum_{t \in Q \cap D} \frac{\sqrt{c(t,D)} \times (1 + log\frac{N}{df(t)+1})^2}{\sqrt{|D|}},$$

where $c(t,D)$ is the number of occurrences of term t in document D, $df(t)$ is the number of documents that contain term t, N is the number of documents in the collection, $|D|$ is the length of document D, $|Q|$ is the length of query Q, $|Q \cap D|$ is the number of terms that occur in both query Q and document D.

Note that the default similarity function can also boost term weighting based on user specified requirements, e.g., the importance of the fields. Since the data used in the evaluation only contain one field, we do not include the boosting factors in the retrieval functions for the sake of clarity.

3.2 Axiomatic Retrieval Function

Axiomatic approaches to Information Retrievla have been proposed to develop retrieval functions. Previous study [2] has shown that the derived retrieval function is more robust than other state-of-the-art retrieval functions with comparable optimal performance. The following function (F2-EXP) is one of the best performing functions that have been derived using axiomatic approach.

$$S(Q,D) = \sum_{t \in Q \cap D} c(t,Q) \cdot (\frac{N}{df(t)})^k \cdot \frac{c(t,D)}{c(t,D) + s + \frac{s \cdot |D|}{avdl}},$$

where $S(Q,D)$ represents the relevance score of document D for query Q, $c(t,Q)$ is the number of occurrences of term t in query Q, $df(t)$ is the number of documents that contain term t, $c(t,D)$ is

Table 1. Performance Comparison

Collection	Function	MAP	P@5	P@10	P@20	P@100	NumRR
web	Lucene Default	0.18	0.34	0.32	0.26	0.13	1474
	Axiomatic	0.26	0.44	0.41	0.33	0.18	1644
trec7	Lucene Default	0.15	0.4	0.35	0.30	0.15	2018
	Axiomatic	0.18	0.41	0.38	0.33	0.17	2137
trec8	Lucene Default	0.17	0.4	0.38	0.31	0.18	2155
	Axiomatic	0.19	0.42	0.38	0.30	0.17	2168

the number of occurrences of term t in document D, $|D|$ is length of document D, and $avdl$ is the average document length in the collection. s and k are two parameters. Setting them to default values, i.e., $k = 0.35$ and $s = 0.5$, can often yield to near-optimal performance for different data colletions.

4 Experiments

In this section, we experimentally compare the default function in Lucene with the axiomatic retrieval function. Experiment results show that the axiomatic retrieval function achieves much higher search accuracy than the default function in Lucene.

4.1 Experiment Setup

We conduct the experiments over three standard TREC collections: the Web data used in TREC 8 (**web**) and the ad hoc data used in TREC 7 (**trec7**) and TREC 8 (**trec8**). Both collections are relatively large and represent two different genres. **web** collection includes 227,725 documents and 50 queries, and either **trec7** or **trec8** collection contains 528,155 documents and 50 queries. We use StandardAnalyzer to pre-process both query sets and document collections. In particular, we represent all queries as BOOLEAN queries provided by Lucene.

4.2 Experiment Results

Table 1 shows the evaluation results for both the default function in Lucene and the axiomatic retrieval function. P@N means the precision at top N ranked documents. NumRR is the number of returned relevant documents. MAP is the mean average precision as we described in the previous section. For all the measures, higher value means the better results.

On **web** collection, the axiomatic function has higher MAP value compared with Lucene's default function, i.e., 0.262 vs. 0.188. In particular, we can see the axiomatic function is able to rank relevant documents higher based on P@N value. For example, it returns approximately 0.5 more relevant document in top 5 ranked documents, 1 more relevant documents in top 10, 1.4 more relevant documents in top 20, and 5 more relevant documents in top 100. Furthermore, for all the queries in a collection, the axiomatic function can return more relevant documents in the top ranked 1000 documents than the Lucene's default function, i.e., 1644 vs. 1474. We can make the similar observation on the other two collections.

5 Conclusions

In this report, we empirically evaluate the retrieval performance of the default retrieval function in Lucene, and compared it with a state-of-art retrieval function, i.e., axiomatic retrieval function. Experiments show that the performance of the default function is much worse than the state-of-the-art retrieval function. Therefore, it would be desirable to implement a state-of-the-art retrieval function such as the axiomatic retrieval function in Lucene. We have implemented the axiomatic retrieval functions (AXTermQuery.java and AXTermScorer.java) based on Lucene and created a web page that provides the download the implementation and example codes of using the implementation (BuildIndex.java and RetAXEval.java). The codes can be downloaded at http://www.ece.udel.edu/~hfang/LuceneAX.html. We hope that the implementation could be combined with the future releases of Lucene.

References

[1] Default Similarity Function of Lucene. http://lucene.zones.apache.org:8080/hudson/job/lucene-nightly/javadoc/org/apache/lucene/search/similarity.html.

[2] H. Fang and C. Zhai. *An Exploration of Axiomatic Approaches to Information Retrieval*. 2005.

[3] E. Hatcher and O. Gospodnetic. *Lucene in Action*. Manning Publications Co., 2004.

[4] Lucene. http://lucene.apache.org/java/docs/.

[5] Nutch. http://lucene.apache.org/nutch/.

[6] E. Voorhees and D. Harman, editors. *Proceedings of Text REtrieval Conference (TREC1-9)*. NIST Special Publications, 2001. http://trec.nist.gov/pubs.html.