Practical Relevance Ranking for 10 Million Books.

Tom Burton-West

Digital Library Production Service, University of Michigan Library, Ann Arbor, Michigan, US
tburtonw@umich.edu

Abstract. In this paper we briefly describe our production environment and some of the open questions about relevance ranking for 10 million books. Then we describe our participation in the Prove It task of the INEX Social Book Search Track. We found that the queries supplied with the Prove It topics were not specific enough to provide good retrieval results. In contrast, the *fact* fields of the topics, when used as queries, provided good retrieval results. However, our query logs show that users are unlikely to enter queries as long as the *fact* fields. We tried to create queries that provided good retrieval results but better represented the queries in our logs. We also experimented with simulating the two-stage search process used in our production system when searching the entire corpus of 10 million books to find relevant books and then searching within the book to find relevant pages. While we succeeded in creating queries that were more specific than those supplied in the Prove It topics, and those queries produced better results, questions remain about how representative these created queries are of real user queries.

1 Introduction

The HathiTrust Digital Library is a digital preservation repository and access platform supported by a partnership of over sixty research institutions. The Digital Library Production Service of the University of Michigan Library supports search services over the full text of more than 10 million books in the repository, through HathiTrust Full-text search. Indexing on this scale presents a number of issues for relevance ranking.

The corpus used in previous INEX Book Tracks and in the 2011 and 2012 “Prove It” task contains the OCR and MARC metadata for about 50,000 public domain books. Since it is likely that most of these books are included among the over 3 million public domain books within the HathiTrust repository, this corpus should provide a good test bed for relevance ranking experiments for HathiTrust Full-text search.[1]

1.1 The HathiTrust Full-Text Search System

We use custom-built middleware on top of the open-source Solr/Lucene search platform to index the 10 million books in the repository. In the original design of HathiTrust search, for performance and scalability reasons, we decided on a two tier indexing and searching architecture. Each tier is implemented as a separate Solr instance with its own separate index.[2] The first tier indexes all 10 million books with the unit of indexing the complete book. The second tier uses the page as the unit of indexing, but rather than indexing all 3 billion pages in the repository, documents are indexed on the page level on-demand. Searches are first executed against the index of 10 million books. Once a user clicks on a result, they are taken to a book viewer application. If they search within the book, the book is indexed on-the-fly and added to the separate page-level index.

1.2 Practical Relevance Concerns

Because we have both OCR and high quality MARC metadata, our current relevance ranking in production combines scores from the OCR field and various MARC fields using Solr/Lucene's boosting capability to weight fields. Our current boost values were determined by trial and error. We would like to have a systematic way to determine the optimum relative weights of the OCR and the MARC fields.

We suspect that the default term frequency normalization and length normalization provided by Lucene do not work well with our book length documents. With the exception of a few studies, the work on length normalization for information retrieval has been done on relatively small newswire size documents. Our average document length is around 100,000 words as compared to around 300 for the TREC ad hoc collections or 1,500-1,700 for the Gov2 or ClueWeb collections.

Lucene currently provides an alternative length normalization implementation (SweetSpotSimilarity), and Lucene 4.0 provides alternatives to the current Lucene

[1] Due to the absence of sufficient standard identifiers in the MARC metadata, we were not able to calculate how much of the INEX Prove It corpus is included in the HathTrust repository.

[2] For performance reasons the book-level index is split into 12 shards using Solr's distributed indexing/searching features

ranking model, including BM25 and DFR, which both allow tuning of parameters related to document length.[3] We would like to experiment with these approaches.

In 2007 in the TREC "Million Query Track," researchers at IBM modified Lucene's default length normalization and term frequency normalization with significantly better results on the web length documents in the Gov2 collection used in the Million Query Track [2]. We are interested in determining whether a similar approach would improve relevance ranking for book page size documents in our page-level index.

1.3 The INEX Prove It task

The goal of the Prove It task is to return individual book pages that confirm or refute a factual statement. Determining algorithmically whether a page confirms or refutes a factual statement is a hard problem. As first time participants in the Book Track, we decided to concentrate on retrieving relevant pages in the top of the ranked list and forgo the optional task of identifying whether a page confirms or refutes the fact. Our original goal for the Prove It task was to set up a baseline and then experiment with some of the length normalization approaches discussed above.

2 Preliminary Testing and Experimentation

2.1 Technical Issues

Our indexing, search, and document viewing infrastructure (both for testing and for production) assumes that digitized books are in our repository. For a number of reasons, we couldn't simply insert the INEX Prove It corpus into our repository. We had to do some re-engineering in order to set up a new environment which would allow indexing and running queries against the Prove It corpus. To simulate our production system, we created two indexes of the Prove It corpus, one which indexes the entire book as a single document, and another which indexes individual pages. We mapped MARC metadata into appropriate title, subject, author, etc., fields according to the mapping we use in production. For the Prove It page-level index, we copied the MARC fields (which apply to the whole book) to the indexing unit for each page. Each index has two fields containing the OCR text. The first field uses no stemming or stop words.[4] The second field contains the OCR content stopped with the Lucene default English stop word list and stemmed with the Porter stemmer.

[3] http://searchhub.org/dev/2011/09/12/flexible-ranking-in-lucene-4/

[4] This field simulates our production OCR indexing. In our production index we do not use stop words because we index works in over 400 languages and a stop word in

We also encountered some minor technical issues with corrupted MARC metadata and cleaned up what we could of the MARC metadata before adding it to the indexes.

2.2 Tests and Experiments

We began by running tests against the set of 21 training topics from the previous INEX Prove It task. [5]

To establish a baseline we decided to start by using a default Boolean "OR" operator between the query words and the default Lucene relevance ranking algorithm. Lucene's ranking is a vector-space, tf-idf variation which also includes a "coordination" factor where documents containing a higher percentage of the words in the query get a boost.[6]

We experimented with using different fields of the Prove It task topics. We used the *query* field alone, the *fact* field alone, the *query* and *fact* fields combined, and the *query*, *fact*, and *narrative* fields combined. The *query* fields tend not to include stop words, but the other fields do, so we did a version of each run with and without stopping and stemming. We also experimented with various weighting schemes including down-weighting stemmed versions of the fields, and schemes incorporating the MARC metadata.

It soon became apparent that the relative changes in relevance scores between different treatments using the same field from the topic, such as using stemming and stopping and not using them, or using different weighting schemes, were very small compared to the very large difference between using different fields from the topics. For example, using the *query* field achieved an NDCG@10 of 0.26, while the using the *fact* field achieved NDCG@10 of 0.68 (for the runs using stemming and stopping.)

---

one language is a content word in another. For example the word "die" is a stop word in German but a content word in English. (See http://www.hathitrust.org/blogs/large-scale-search/slow-queries-and-common-words-part-2.) We don't use stemming in the OCR field because stemming is a recall-enhancing process and we want to enhance precision, not recall.

[5] All results reported here use the 2011 qrels (inex11proveit.0_1_2.qrels) which include the additional 535 relevance judgments added by the University of Massachusetts.

[6] http://lucene.apache.org/core/3_6_0/api/core/org/apache/lucene/search/Similarity.html

The queries, (from the *query* field,) are insufficiently specific to retrieve relevant documents, compared to using the *fact* field verbatim as a query. For example, topics 2, 9, 10, 15, 42, 60, and 70 received NDCG@10 scores of 0 for the runs using the *query* field. The *fact* field for topic 70 is: *"Charles Darwin was born in Shrewsbury in 1809."* The *query* field for topic 70 is *"Charles Darwin."* This query does retrieve relevant results, but not in the top 10. Without providing the search engine any clue that the user is looking for a birth date or location, no amount of tuning the relevance algorithm will get these into the top results. Similarly, the *query* field for topic 15, *"Rome capital"* provides neither a clue that the topic is about the re-unification of Italy nor any other clue to the details in the *fact* or *narrative* fields. (The *fact* and *narrative* fields are used in making the relevance judgments). The problems with topic 15, as well as the general problems with the queries being underspecified, were noted by [4] in the INEX 2011 pre-proceedings.

These preliminary results presented us with a dilemma. In order to apply our findings to tune the production system, we wanted to use queries that simulate how real users interact with our production system. However, the *fact* fields supplied with the Prove It topics do not resemble the queries in our logs. The *fact* fields average 21 words. In contrast, in our query logs, the average query length is 2.74 words, and about 77% of our queries are three words or less. Only 10% of our queries are over five words in length and only 1% of queries are 12 words or more. Cummins and O'Riordan [3], found that length normalization needs different tuning parameters for short or long queries. Since we want to experiment with length normalization approaches appropriate for our production system, we do not want to use the *fact* fields which are significantly longer than most of the queries in our production system, as queries for our Prove It experiments.

We did a spot check of a random sample of queries from our logs that were longer than 10 words and found that most of them fell into three categories:

1. Long book titles or titles of government hearings or reports
2. Long quotes, such as "It was the best of times, it was the worst of times, it was the age of wisdom…"
3. Complex Boolean queries

The long quotes tend to be of two types: either they appear to be attempting to find the origin or uses of a well-know phrase such as "a stitch in time saves nine," or they appear to be an attempt to find the book from which a student copied some text (often the quotes are obviously from a textbook). Very few of these long quotes could be interpreted as attempts to verify a fact by copying it verbatim into the search box.

We plan to do user studies to characterize the various search tasks our users engage in when using HathiTrust search. In the absence of such studies, based on the brief log analysis above, and on the literature on interactive information retrieval, we believe that most of our users, who want to verify a fact, don't enter the fact verbatim into the search engine. Instead, we suspect that they start with a relatively short query, (such as those provided by the *query* field of the Prove It topics). When they see that the results are too broad, they add more terms to narrow the scope and bring relevant results to the top of the list.

We decided to simulate this process by recruiting a group of librarians and library programmers to come up with better queries. The intention was to come up with a query a user would eventually use after several iterations of interactive search. Two librarians and two library programmers each contributed queries for 20-21 topics, to provide new queries for all 83 Prove It task topics.

3 Production of New Queries

We asked each query creator to come up with two sets of queries for each topic. The first set was a "reasonable" query designed to bring the most relevant results to the top of the result list. The second set of queries was inspired by recent attempts to simulate interaction [1], and was modeled on the way users search using our production search process. We asked the query creators to give us a two-part query for each topic. The first part is the query they would use when searching our book-level index. The second part is the query that they would use when they searched within the book.

For the first set of queries, a relatively simple change often improved results significantly. For example, the *query* field for topic 9 is [Enchanted Windmill]. This query had an NDCG@10 score of 0. The query creator enclosed the terms in quotes to force the search engine to search for the terms as a phrase: ["Enchanted Windmill"]. This change alone increases the NDCG@10 from 0 to 0.48.

In order to facilitate the process of designing better queries, we modified a copy of our book search application to search the Prove It task page-level index, and to display ranked lists of pages with search terms highlighted. This proved useful in designing the first set of queries.

For the second set of queries, we wanted to provide an interface that executed the first part of the query against the PROVE IT index which indexed the entire book as one unit; and then allowed searching within the book at a page level using the second part of the query, to simulate our production system. Unfortunately, we did not have enough time to implement an interface to allow the query creators to query the PROVE IT indexes in this way. They had to use our production indexes (limited to Public Domain books) to test their queries. One of the librarians noted that, for a number of queries, the production system had many more relevant documents, and suggested that the queries that worked well in the production system might not work as well in the PROVE IT collection.

4 Submitted Runs

Table 1 shows the results for the submitted runs. Unless otherwise specified, the runs all use the default Lucene English stop word list and the Porter stemmer on the OCR. (The "umich" prefix which is attached to all our run names has been removed for ease of reading.) The runs are described below.

Table 1. Submitted Runs

Run	NDCG@10	MAP	P@10
F	0.68	0.32	0.65
FQ	0.61	0.32	0.60
L	0.53	0.21	0.52
Ldismax_marc	0.53	0.21	0.52
HT25	0.34	0.11	0.33
Ldismax_marc (corrected)	0.30	0.11	0.32
Q	0.26	0.14	0.32

Standard runs:

F: The *fact* field

FQ: The *fact* and *query* fields concatenated

Q: The *query* field

Runs using queries created by librarians and library programmers:

L: The "Librarian" queries. These were created by the librarians and library programmers.

Ldismax_marc: The “Librarian” queries processed using the Solr dismax handler with weighting of the OCR, the stemmed and stopped OCR, and the MARC fields.[7] (See Appendix A for the details of the weighting)

HT25: The HathiTrust simulation. This run attempts to simulate how a user would search our production system. In our production system the default is to show the first 25 books on the first result page. We assume that the user would then click on one or more of the results on the first result page and then search within each book. In these runs, the first part of the query was run against the book-level index and the book ids of the books in the top 25 results were used to limit results to only pages within those books when combined with the second part of the query (which was run against the page-level index.)

Some of the queries created by the librarians and library programmers used the advanced search operators available in the production system, such as combining a search within the title, author, or subject fields with a search within the OCR. Due to time constraints we were not able to modify our software to run these automatically for the submitted runs. They were replaced with searches that did not require the advanced search capabilities.

5 Analysis

5.1 Towards “Realistic” Queries with More Specificity

The “Librarian” queries, run L, significantly improved upon the default queries (*query* field) supplied with the INEX topics. The default (Q) queries had an NDCG@10 score of 0.26, while the “Librarian” (L) queries had an NDCG@10 score of 0.53. However, the improved queries still did worse than the “fact” (F) run which had a score of 0.68. We spent some time trying to determine why the queries based on the *fact* field did so much better than the “Librarian” queries. We compared the NDCG@10 scores for individual topics and looked at the topics that had the greatest differences in the scores between the “Librarian” queries and the *fact* queries. In one case, the “Librarian” query misspelled an important proper name. Correction of that error brought the score for that topic from 0.05 to 0.89.

We also noted that some topics were quite complex and seemed to require a relatively large number of terms in order to get good results, for example topics 2 and 3. When relevance judgments are available for all 83 topics, it may be found that a larg-

[7] Due to a parsing error in the argument processing of our test harness, the submitted run was actually run with the same settings as the L run. Ldismax_marc_corrected is the unsubmitted corrected run that actually used the MARC fields and weighting.

er pool of topics might reduce the effect of particularly difficult or complex topics on the overall scores.

5.2 Simulation of a Two-Stage Search Process.

The HT25 run did very poorly. We investigated several topics where the score for the HT25 run was much worse than for the L run. We found a number of issues (discussed below.) We still believe the idea of simulating a book-level query followed by a page-level query is sound, but we need to further investigate the best way to implement this so that it achieves a reasonable simulation of the production system.

As we did not have time to set up a user interface to simulate our production system, our query creators had to use our production system when working on the HT25 two-part queries. We believe that differences between the production environment and our PROVE IT test environment are the major cause of the poor performance of the HT25 queries. As an example, for topic 0 the book-level query was [“battle of new Orleans” killed wounded] and the page-level query was [killed wounded]. In this case, we suspect that the query creators were misled because they were testing queries in our production system. In our production system, because users are searching the full text of 10 million records, we have set the default operator to a Boolean “AND.” For the PROVE IT book-level search we had the default operator set to a Boolean “OR.” This appears to have been a mistake.

In this particular case, in the production index, because of the default “AND” operator, the book-level query [“battle of new orleans” killed wounded] produced only books containing both the phrase “battle of new Orleans” and the words “killed” and “wounded.” Thus the subsequent page-level query [killed wounded] was only searching within books that at a minimum contained the phrase “battle of new Orleans,” and that page-level query appeared to be sufficiently specific to get good results.

On the other hand, in the PROVE IT book-level index, because of the default “OR” operator, some of the top results contained books that did not contain the phrase “battle of new Orleans”, but contained a huge number of occurrences of the words “killed” and “wounded.” When the page-level query was run against the page-level index, limited to the top 25 books from the book-level query, the pages that had the most occurrences of the words “killed” and “wounded” but did not mention “battle of new orleans” came to the top. A similar problem was found for several other low scoring HT queries, where the page-level query was specific enough in the context of the production environment, but not specific enough in the test environment.

We also discovered that, due to a parsing error in converting the submitted queries to a format suitable for running against Solr, the HT25 query for topic 60 returned 0 results.

6 Conclusions and Future Work

Our original goal was to set up a baseline and then experiment with some of the weighting and length normalization approaches available in Solr 4.0. However, we found we could not use the existing Prove It queries (from the *query* field of the topics) to generate a baseline. As noted in the previous INEX Prove It track, retrieval using the *fact* field produced much better results than using the *query* field. The differences in relevance scores between using these two different fields were very large. In contrast, when using a single field, changes in relevance scores resulting from using different query processing approaches or field weighting approaches were comparatively small. Since our query logs suggest that the *fact* field is not a realistic approximation of typical user behavior, and since our goal was to establish a baseline in order to improve our production system, we focused on trying to create more realistic queries and on simulating user interaction in our production system.

While we succeeded in creating queries that were more specific than those supplied in the Prove It topics, and those queries produced better results, questions remain about how representative these created queries are of real user queries. We plan to do user studies and further analysis of our query logs to determine the extent of fact-checking queries and better understand their characteristics. After that analysis is completed, we will need to determine whether to use the INEX Prove It corpus and our “Librarian” queries as a baseline to experiment with weighting and length normalization or instead to use real queries from our logs. If we use real queries from our logs, we will need to find a way to get relevance judgments.

We would also like to investigate further why using the HT25 queries did so poorly and try to devise a better way to simulate the two-level interactive query process used in our production system.

7 Acknowledgements

The author would like to thank Shevon Desai, Philip Farber, Christina Powell and Peter Ulintz, from the University of Michigan Library who provided the queries used in this work.

Bibliography

1. Azzopardi, Leif, Jarvelin, Kalervo, Kamps, Jaap and Mark D. Smucker. 2011. Report on the SIGIR 2010 workshop on the simulation of interaction. *SIGIR Forum* 44, 2 (January 2011), 35-47
2. Cohen, D., Amitay, E., Carmel, D.: Lucene and juru at trec 2007: 1-million queries track. In: Proceedings of the 16th Text REtrieval Conference (TREC 2007) (November 2007)
3. Cummins, Ronan and O'Riordan, Colm. 2009. The effect of query length on normalisation in information retrieval. In *Proceedings of the 20th Irish conference on Artificial intelligence and cognitive science* (AICS'09), Lorcan Coyle and Jill Freyne (Eds.). Springer-Verlag, Berlin, Heidelberg, 26-32.
4. Feild, H., Cartright, M. and Allan, J. 2011l "The University of Massachusetts Amherst's participation in the INEX 2011 Prove It Track," In the INEX 2011 Workshop Pre-proceedings. Shlomo Geva, Jaap Kamps, Ralf Shenkel eds. IR Publications Amsterdam. INEX Working Notes Series, Volume 2011 Saarbrucken, Germany, December 12-14, 2011.

Appendix: Weighting for Ldismax_marc run

The weighting is given below in Solr edismax query syntax. See http://wiki.apache.org/solr/ExtendedDisMax for an explanation of the syntax. The fields are as follows:

ocr: The OCR content with no stopping or stemming	
ocrPorterStop:	The OCR content with stopping using Lucene default English stop words and the Porter stemmer
allfiedsProper:	A concatenation of all the MARC fields no stopping or stemming

```
{!edismax'
pf='ocr^25000+ocrPorterStop^400+allfieldsProper^50+'
pf3='ocr^2500+ocrPorterStop^40+allfieldsProper^10+'
pf2='ocr^250+ocrPorterStop^25+allfieldsProper^10+'qf='ocr
^25+ocrPorterStop^10+allfieldsProper^10+' mm='2<-1
5<67%25' tie='0.1' }'
```