Combining Page Scores for XML Book Retrieval

Ray R. Larson

School of Information
University of California, Berkeley
Berkeley, California, USA, 94720-4600
ray@ischool.berkeley.edu

Abstract. In the 2010 INEX Evaluation UC Berkeley participated only in the Book track, and specifically the "Best Books to Reference" task that seeks to produce a list of the "best books" for a topic. This year we wanted to compare our best performing method from last year with approaches that combine the scores obtained by ranking at the page level to arrive at the ranking for a book. We tested a number of combinations for this approach, and were able to obtain the top score for the "Best Books" task.

1 Introduction

In our 2009 work on the INEX Book Track, we tried a variety of different approaches for our Book Track runs, including the TREC2 logistic regression probabilistic model as well as various fusion approaches including combining the Okapi BM-25 algorithm with other approaches. But in most cases our previous approaches used only full-book level indexes. We observed that most of the fusion approaches that we had tried were not as effective as the TREC2 Logistic Regression with Blind Feedback, so we took that as our baseline for this year, and none of our approaches were as effective as those reported by some other groups. After testing a number of approaches using book-level indexes and different fusion weights, we found that this was still the case, and that these attempts often led to poor matches being ranked highly. We decided instead to try some radical simplification of the ranking process for books. This was driven by observations from earlier INEX evaluations and from some of our digital library work that often the books with highly ranked *pages* turned out to be more better choices for the user than books with high overall ranking scores. Since we had generated page-level indexes for all of the books (see below), we decided to try two simple approaches. A probabilistic approach based on our logistic regression algorithm (but without blind feedback), and a simple coordination-level match for pages.

In this paper we will first discuss the algorithms and operators used in our official INEX 2010 Book Track runs. Then we will look at how these algorithms and operators were used in combination with page-level indexes for our submissions, and finally we will discuss possible directions for future research.

S. Geva et al. (Eds.): INEX 2010, LNCS 6932, pp. 154–163, 2011.

2 The Retrieval Algorithms and Fusion Operators

This section largely duplicates parts of earlier INEX papers in describing the probabilistic retrieval algorithms used for the Book track in INEX this year. Although the algorithms are the same as those used in previous years for INEX and in other evaluations (such as CLEF and NTCIR), including a blind relevance feedback method used in combination with the TREC2 algorithm, we are repeating the formal description here instead of refering to those earlier papers alone. In addition we will discuss the simple methods used to combine the results of searches of book page elements in the collections. All runs used our Cheshire II XML/SGML search engine [9,8,7] which also supports a number of other algorithms for distributed search and operators for merging result lists from ranked or Boolean sub-queries.

2.1 TREC2 Logistic Regression Algorithm

Once again the primary algorithm used for our INEX baseline runs is based on the *Logistic Regression* (LR) algorithm originally developed at Berkeley by Cooper, et al. [5]. The version that we used for Adhoc tasks was the Cheshire II implementation of the "TREC2" [4,3] that has provided good retrieval performance earlier evaluations[9,10]. As originally formulated, the LR model of probabilistic IR attempts to estimate the probability of relevance for each document based on a set of statistics about a document collection and a set of queries in combination with a set of weighting coefficients for those statistics. The statistics to be used and the values of the coefficients are obtained from regression analysis of a sample of a collection (or similar test collection) for some set of queries where relevance and non-relevance has been determined. More formally, given a particular query and a particular document in a collection $P(R \mid Q, D)$ is calculated and the documents or components are presented to the user ranked in order of decreasing values of that probability. To avoid invalid probability values, the usual calculation of $P(R \mid Q, D)$ uses the "log odds" of relevance given a set of S statistics derived from the query and database, such that:

$$\begin{aligned}
\log O(R|C,Q) &= log\frac{p(R|C,Q)}{1-p(R|C,Q)} = log\frac{p(R|C,Q)}{p(\overline{R}|C,Q)} \\
&= c_0 + c_1 * \frac{1}{\sqrt{|Q_c|}+1}\sum_{i=1}^{|Q_c|}\frac{qtf_i}{ql+35} \\
&+ c_2 * \frac{1}{\sqrt{|Q_c|}+1}\sum_{i=1}^{|Q_c|}\log\frac{tf_i}{cl+80} \\
&- c_3 * \frac{1}{\sqrt{|Q_c|}+1}\sum_{i=1}^{|Q_c|}\log\frac{ctf_i}{N_t} \\
&+ c_4 * |Q_c|
\end{aligned}$$

where C denotes a document component and Q a query, R is a relevance variable, and

$p(R|C,Q)$ is the probability that document component C is relevant to query Q,
$p(\overline{R}|C,Q)$ the probability that document component C is not relevant to query Q, (which is 1.0 - $p(R|C,Q)$)
$|Q_c|$ is the number of matching terms between a document component and a query,
qtf_i is the within-query frequency of the ith matching term,
tf_i is the within-document frequency of the ith matching term,
ctf_i is the occurrence frequency in a collection of the ith matching term,
ql is query length (i.e., number of terms in a query like $|Q|$ for non-feedback situations),
cl is component length (i.e., number of terms in a component), and
N_t is collection length (i.e., number of terms in a test collection).
c_k are the k coefficients obtained though the regression analysis.

Assuming that stopwords are removed during index creation, then ql, cl, and N_t are the query length, document length, and collection length, respectively. If the query terms are re-weighted (in feedback, for example), then qtf_i is no longer the original term frequency, but the new weight, and ql is the sum of the new weight values for the query terms. Note that, unlike the document and collection lengths, query length is the relative frequency without first taking the log over the matching terms.

The coefficients were determined by fitting the logistic regression model specified in $\log O(R|C,Q)$ to TREC training data using a statistical software package. The coefficients, c_k, used for our official runs are the same as those described by Chen[1]. These were: $c_0 = -3.51$, $c_1 = 37.4$, $c_2 = 0.330$, $c_3 = 0.1937$ and $c_4 = 0.0929$. Further details on the TREC2 version of the Logistic Regression algorithm may be found in Cooper et al. [4].

2.2 Blind Relevance Feedback

It is well known that blind (also called pseudo) relevance feedback can substantially improve retrieval effectiveness in tasks such as TREC and CLEF. (See for example the papers of the groups who participated in the Ad Hoc tasks in TREC-7 (Voorhees and Harman 1998)[11] and TREC-8 (Voorhees and Harman 1999)[12].)

Blind relevance feedback is typically performed in two stages. First, an initial search using the original queries is performed, after which a number of terms are selected from the top-ranked documents (which are presumed to be relevant). The selected terms are weighted and then merged with the initial query to formulate a new query. Finally the reweighted and expanded query is run against the same collection to produce a final ranked list of documents. It was a simple extension to adapt these document-level algorithms to document components for INEX.

The TREC2 algorithm has been been combined with a blind feedback method developed by Aitao Chen for cross-language retrieval in CLEF. Chen[2] presents

a technique for incorporating blind relevance feedback into the logistic regression-based document ranking framework. Several factors are important in using blind relevance feedback. These are: determining the number of top ranked documents that will be presumed relevant and from which new terms will be extracted, how to rank the selected terms and determining the number of terms that should be selected, how to assign weights to the selected terms. Many techniques have been used for deciding the number of terms to be selected, the number of top-ranked documents from which to extract terms, and ranking the terms. Harman [6] provides a survey of relevance feedback techniques that have been used.

Obviously there are important choices to be made regarding the number of top-ranked documents to consider, and the number of terms to extract from those documents. We used the same default as last year, i.e., top 10 terms from 10 top-ranked documents. The terms were chosen by extracting the document vectors for each of the 10 and computing the Robertson and Sparck Jones term relevance weight for each document. This weight is based on a contingency table where the counts of 4 different conditions for combinations of (assumed) relevance and whether or not the term is, or is not in a document. Table 1 shows this contingency table.

Table 1. Contingency table for term relevance weighting

	Relevant	Not Relevant	
In doc	R_t	$N_t - R_t$	N_t
Not in doc	$R - R_t$	$N - N_t - R + R_t$	$N - N_t$
	R	$N - R$	N

The relevance weight is calculated using the assumption that the first 10 documents are relevant and all others are not. For each term in these documents the following weight is calculated:

$$w_t = log \frac{\frac{R_t}{R - R_t}}{\frac{N_t - R_t}{N - N_t - R + R_t}} \quad (1)$$

The 10 terms (including those that appeared in the original query) with the highest w_t are selected and added to the original query terms. For the terms not in the original query, the new "term frequency" (qtf_i in main LR equation above) is set to 0.5. Terms that were in the original query, but are not in the top 10 terms are left with their original qtf_i. For terms in the top 10 and in the original query the new qtf_i is set to 1.5 times the original qtf_i for the query. The new query is then processed using the same TREC2 LR algorithm as shown above and the ranked results returned as the response for that topic.

2.3 Coordination Level Matching

Coordination level matching is the first simple step towards ranking results beyond simple Boolean matches. Basic coordination level matching (CML) is

simply the number of terms in common between the query and the document component or $|Q_c|$ as defined above. In the implementation that we use in the Cheshire II system, the coordination level match (CLM) also takes into account term frequency, thus it is simply:

$$CLM_c = \sum_{i=1}^{|Q_c|} tf_i \tag{2}$$

Where the variables are the defined the same as defined above. Obviously, with this simple form, it is possible for terms that have very high frequency to dominate. To combat this an additional filter removes all results that match on fewer than 1/4 of the search terms.

2.4 Result Combination Operators

As we have also reported previously, the Cheshire II system used in this evaluation provides a number of operators to combine the intermediate results of a search from different components or indexes. With these operators we have available an entire spectrum of combination methods ranging from strict Boolean operations to fuzzy Boolean and normalized score combinations for probabilistic and Boolean results. These operators are the means available for performing fusion operations between the results for different retrieval algorithms and the search results from different components of a document.

However, our approach for the page-level searches done for this evaluation was to simply sum the page-level results for each book. Thus, for the CLM runs, if a particular query retrieved 10 pages from a given book, the final ranking score would be the sum of the CLM values for each page. Although the runs using the TREC2 logistic regression algorithm return estimates of the probability of relevance for each page, we decided to treat these also as simple scores and sum each matching page estimate for each book.

3 Database and Indexing Issues

The Book Track data used this year was the same as last year. In indexing we attempted to use multiple elements or components that were identified in the Books markup including the Tables of Contents and Indexes as well as the full text of the book, since the goal of the "Best Books" task was to retrieve entire books and not elements, the entire book was retrieved regardless of the matching elements.

Table 2 lists the Book-level (/article) indexes created for the INEX Books database and the document elements from which the contents of those indexes were extracted.

Cheshire system permits parts of the document subtree to be treated as separate documents with their own separate indexes. Tables 3 & 4 describe the XML components created for the INEX Book track and the component-level indexes that were created for them.

Table 2. Book-Level Indexes for the INEX Book Track 2009-10

Name	Description	Contents	Vector?
topic	Full content	//document	Yes
toc	Tables of Contents	//section@label="SEC_TOC"	No
index	Back of Book Indexes	//section@label="SEC_INDEX"	No

Table 3. Components for INEX Book Track 2009-10

Name	Description	Contents
COMPONENT_PAGE	Pages	//page
COMPONENT_SECTION	Sections	//section

Table 3 shows the components and the paths used to define them. The first, refered to as COMPONENT_PAGE, is a component that consists of each identified page of the book, while COMPONENT_SECTION identifies each section of the books, permitting each individual section or page of a book to be retrieved separately. Because most of the areas defined in the markup as "section"s are actually paragraphs, we treat these as if they were paragraphs for the most part.

Table 4. Component Indexes for INEX Book Track 2009-10

Component or Index Name	Description	Contents	Vector?
COMPONENT_SECTION			
para_words	Section Words	* (all)	Yes
COMPONENT_PAGES			
page_words	Page Words	* (all)	Yes

Table 4 describes the XML component indexes created for the components described in Table 3. These indexes make the individual sections (such as COMPONENT_SECTION) of the INEX documents retrievable by their titles, or by any terms occurring in the section. These are also proximity indexes, so phrase searching is supported within the indexes.

We also have indexes created using the MARC data (book-level metadata) made available, but these were not used this year.

3.1 Indexing the Books XML Database

Because the structure of the Books database was derived from the OCR of the original paper books, it is primarily focused on the page organization and layout and not on the more common structuring elements such as "chapters" or "sections". Because this emphasis on page layout goes all the way down to the individual word and its position on the page, there is a very large amount

of markup for page with content. For this year's original version of the Books database, there are actually NO text nodes in the entire XML tree, the words actually present on a page are represented as attributes of an empty word tag in the XML. The entire document in XML form is typically multiple megabytes in size. A separate version of the Books database was made available that converted these empty tags back into text nodes for each line in the scanned text. This provided a significant reduction in the size of database, and made indexing much simpler. The primary index created for the full books was the "topic" index containing the entire book content.

We also created page-level "documents" as we did last year. As noted above the Cheshire system permits parts of the document subtree to be treated as separate documents with their own separate indexes. Thus, paragraph-level components were extracted from the page-sized documents. Because unique object (page) level identifiers are included in each object, and these identifiers are simple extensions of the document (book) level identifier, we were able to use the page-level identifier to determine where in a given book-level document a particular page or paragraph occurs, and generate an appropriate XPath for it.

Indexes were created to allow searching of full page contents, and component indexes for the full content of each of individual paragraphs on a page. Because of the physical layout based structure used by the Books collection, paragraphs split across pages are marked up (and therefore indexed) as two paragraphs. Indexes were also created to permit searching by object id, allowing search for specific individual pages, or ranges of pages.

The system problems encountered last year have been (temporarily) corrected for this years submissions. Those problems were caused by the numbers of unique terms exceeding the capacity of the integers used to store them in the indexes. For this year, at least, moving to unsigned integers has provided a temporary fix for the problem but we will need to rethink how statistical summary information is handled in the future – perhaps moving to long integers, or even floating point numbers and evaluating the tradeoffs between precision in the statistics and index size (since moving to Longs could double index size).

4 INEX 2010 Book Track Runs

We submitted nine runs for the Book Search task of the Books track,

As Table 5 shows, a small number of variations of algorithms and search elements were tried this year. The small number was largely due to some issues in indexing (due to a bug in page indexes that took a lot of time to locate and fix). With more that 16 million pages, response time was very good for the basic search operations, but slowed dramatically whenever data from records was needed.

In Table 5 the first column is the run name (all of our official submissions had names beginning with "BOOKS10" which has been removed from the name), the second column is a short description of the run. We used only the main "fact" element of the topics in all of our runs. The third column shows which algorithms

Table 5. Berkeley Submissions for the INEX Book Track 2009

Name	Description	Algorithm	Combined?
T2FB_BASE_BST	Uses book-level topic index and blind feedback	TREC2 +BF	NA
CLM_PAGE_SUM	Uses page components and page_words index	CLM	Sum
CLM_PAGE_SUM_300	Uses page components and page_words index	CLM	Sum
T2_PAGE_SUM_300	Uses page components and page_words index	TREC2	Sum

Table 6. Evaluation results for the INEX 2010 Best Books task

Run ID	MAP	P@10	**NDCG@10**
p14-BOOKS2010_CLM_PAGE_SUM.trec	0.1507	0.2714	0.2017
p14-BOOKS2010_CLM_PAGE_SUM_300.trec	0.1640	0.2810	0.2156
p14-BOOKS2010_T2FB_BASE_BST.trec	0.3981	0.5048	0.5456
p14-BOOKS2010_T2_PAGE_SUM_300.trec	0.5050	0.6667	0.6579
p6-inex10.book.fb.10.50.trec	0.3087	0.4286	0.3869
p6-inex10.book.trec	0.3286	0.4429	0.4151
p98-baseline_1.trec	0.4374	0.5810	0.5764
p98-baseline_1_wikifact.trec	0.4565	0.5905	0.5960
p98-baseline_2.trec	0.4806	0.6143	0.6302
p98-baseline_2_wikifact.trec	0.5044	0.6381	0.6500
p98-fact_query_10wikibests.trec	0.4328	0.5714	0.5638
p98-fact_query_entropy.trec	0.4250	0.5476	0.5442
p98-fact_query_tfidfwiki.trec	0.3442	0.4667	0.4677
p98-fact_query_tfwiki.trec	0.4706	0.5571	0.5919
p98-fact_stanford_deps.trec	0.4573	0.5857	0.5976

where used for the run, TREC2 is the TREC2 Logistic regression algorithm described above, "BF" means that blind relevance feedback was used in the run, and CLM means that the CLM algorithm described above (2) was used.

Table 6 shows the official results for the Best Books task in the book retrieval tasks. The first four lines of Table 6 show the results for our official submitted runs described in Table 5 (as Mean Average Precision, Precision at 10, and Normalized Discounted Cumulative Gain at 10, the official measures used for the task). As these results show, the simplest methods are definitely not the best methods in the best book retrieval task. Our runs using simple coordination level matching both showed significantly worse results (note that the full result data for each participant was not available, and so we could not calculate true statistical significance. They certainly appear to be considerably worse than any of the other submitted runs). On the other hand, our baseline run (T2FB_BASE_BST), appears in the middle range of the score distribution, and our test run using the TREC 2 algorithm at the page level and simple summation of scores obtained the best results of any official run.

5 Conclusions and Future Directions

The results of the Books track were only recently made available to participants, but a few observations can be made about the runs. Our belief that the simple CLM on pages might outperform the TREC2 approach on pages (as suggested in our notebook paper) was wildly wrong, given how the CLM approaches were outperformed by every other approach tried this year. Perhaps this can be seen as an object lesson of the unreliability of "eyeballing" evaluation results in the absence of actual result data. We have known for some time that the base TREC 2 algorithm usually provides fairly high precision (and usually good recall too when combined with the blind feedback approach described above). This appears to be the main factor for the success in the Best Books task, high precision search at the page level, with book ranking scores based on a summation of page scores for the books with highly ranked pages.

Of course, it is quite reasonable that the TREC2 algorithm outperforms the CLM approach. The CLM ranking value, as noted above is purely concerned with term occurrences in the text, while the TREC2 logistic regression algorithm attempts to estimate the probability that a document component is relevant for a given query. While a CLM value can be inflated by multiple occurrences of any given term (even very common terms), the TREC2 approach averages the contributions of the various terms, and therefore tends to favor pages with more query terms with higher IDF values.

As noted by one reviewer, there is a question of why page-level search seems to capture more appropriate information than document (or book-level) search. I would suggest that this may be more an artifact of the types of questions/topics than necessarily a characteristic of book searching. Most of the topics are actually framed as "factoid" types of questions, most of which can be answered in one or a few pages, and which are unlikely to have full book-length treatments (although some of the particular people, places, or events making up parts of the questions might well have full books, which contain the pages that could answer the questions).

Now that the relevance data for this task is available, we plan to test other approaches to merging page-level results, as well as other basic ranking algorithms for the page search component, including, naturally, the use of blind feedback at the page level. The reason that this latter was not included in the official runs was that we underestimated the time it would take to build the vector files needed for each page, and ran out of time to make the official deadline.

References

1. Chen, A.: Multilingual information retrieval using english and chinese queries. In: Peters, C., Braschler, M., Gonzalo, J., Kluck, M. (eds.) CLEF 2001. LNCS, vol. 2406, pp. 44–58. Springer, Heidelberg (2002)
2. Chen, A.: Cross-Language Retrieval Experiments at CLEF 2002. In: Peters, C., Braschler, M., Gonzalo, J. (eds.) CLEF 2002. LNCS, vol. 2785, pp. 28–48. Springer, Heidelberg (2003)

3. Chen, A., Gey, F.C.: Multilingual information retrieval using machine translation, relevance feedback and decompounding. Information Retrieval 7, 149–182 (2004)
4. Cooper, W.S., Chen, A., Gey, F.C.: Full Text Retrieval based on Probabilistic Equations with Coefficients fitted by Logistic Regression. In: Text Retrieval Conference (TREC-2), pp. 57–66 (1994)
5. Cooper, W.S., Gey, F.C., Dabney, D.P.: Probabilistic retrieval based on staged logistic regression. In: 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Copenhagen, Denmark, June 21-24, pp. 198–210. ACM, New York (1992)
6. Harman, D.: Relevance feedback and other query modification techniques. In: Frakes, W., Baeza-Yates, R. (eds.) Information Retrieval: Data Structures & Algorithms, pp. 241–263. Prentice-Hall, Englewood Cliffs (1992)
7. Larson, R.R.: A logistic regression approach to distributed IR. In: SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland, August 11-15, pp. 399–400. ACM, New York (2002)
8. Larson, R.R.: A fusion approach to XML structured document retrieval. Information Retrieval 8, 601–629 (2005)
9. Larson, R.R.: Probabilistic retrieval, component fusion and blind feedback for XML retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 225–239. Springer, Heidelberg (2006)
10. Larson, R.R.: Ranking and fusion approaches for XML book retrieval. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 179–189. Springer, Heidelberg (2010)
11. Voorhees, E., Harman, D. (eds.): The Seventh Text Retrieval Conference (TREC-7). NIST (1998)
12. Voorhees, E., Harman, D. (eds.): The Eighth Text Retrieval Conference (TREC-8). NIST (1999)