Distributed EDLSI, BM25, and Power Norm at TREC 2008

April Kontostathis, Department of Mathematics and Computer Science, Ursinus College, Collegeville PA 19426, akontostathis@ursinus.edu

Andrew Lilly, Department of Computer Science, University of Saskatchewan, Saskatoon, SK, S7N 5C9, abl657@mail.usask.ca

Raymond J. Spiteri, Department of Computer Science, University of Saskatchewan, Saskatoon, SK, S7N 5C9, spiteri@cs.usask.ca

Abstract—This paper describes our participation in the TREC Legal competition in 2008. Our first set of experiments involved the use of Latent Semantic Indexing (LSI) with a small number of dimensions, a technique we refer to as Essential Dimensions of Latent Semantic Indexing (EDLSI). Because the experimental dataset is large, we designed a distributed version of EDLSI to use for our submitted runs. We submitted two runs using distributed EDLSI, one with $k = 10$ and another with $k = 41$, where k is the dimensionality reduction parameter for LSI. We also submitted a traditional vector space baseline for comparison with the EDLSI results. This article describes our experimental design and the results of these experiments. We find that EDLSI clearly outperforms traditional vector space retrieval using a variety of TREC reporting metrics.

We also describe experiments that were designed as a followup to our TREC Legal 2007 submission. These experiments test weighting and normalization schemes as well as techniques for relevance feedback. Our primary intent was to compare the BM25 weighting scheme to our power normalization technique. BM25 outperformed all of our other submissions on the competition metric (F1 at K) for both the ad hoc and relevance feedback tasks, but Power normalization outperformed BM25 in our ad hoc experiments when the 2007 metric (estimated recall at B) was used for comparison.

1 Introduction

In the 2007 TREC Legal competition, our submissions were designed to test the effectiveness of a variety of simple normalization schemes on a large dataset. We had planned a comparison with some of the state-of-the-art systems for weighting and normalization, but these activities were not completed in time for the TREC 2007 submissions. Thus, in 2008 one of our primary goals was to compare power normalization [9] to the BM25 weighting scheme [13]. We submitted five ad hoc runs and six relevance feedback runs that compare these algorithms.

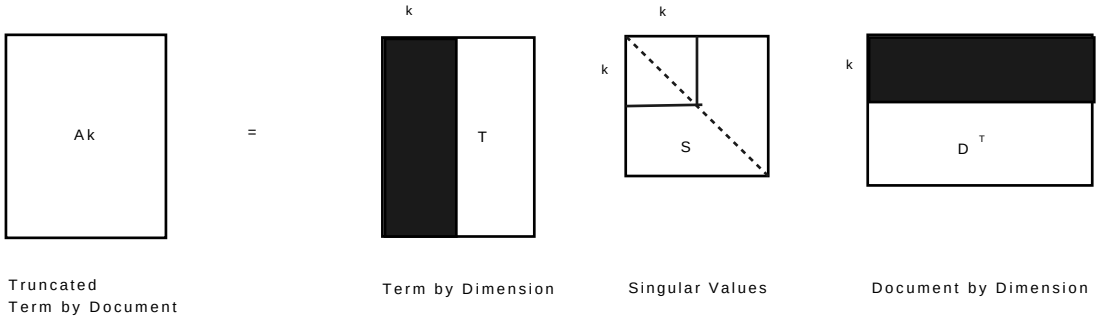
We have extensive experience with Latent Semantic Indexing (LSI) [5], [7], [8], [15], [16], [11]. Thus we were also eager to see how LSI would work on the IIT Complex Document Information Processing (IIT CDIP) test collection, which contains approximately 7 million documents (57 GB of uncompressed text). Specifically, we wanted to see if the Essential Dimensions of Latent Semantic Indexing (EDLSI) [5] approach would scale to this large collection and what the optimal k value would be. We have used SVD updating, folding-in, and folding-up in previous work [15], [16], [11], and it appeared that these techniques would be useful for handling a collection of this size.

This year, teams participating in the TREC Legal task were required to indicate the K and K_h value for each query. K is the threshold at which the system believes the competing demands of recall and precision are best balanced (using the F1@K measure shown in Equation 1), and K_h is the corresponding threshold for highly relevant documents. Assessors assigned a value of highly relevant when judging documents for the first time this year. Much of our preparatory work for the competition centered on determining appropriate ways to set K and K_h for each query.

$$F1@K = (2 * P@K * R@K)/(P@K + R@K) \quad (1)$$

This paper is organized as follows: Sections 2, 3, and 4 describe the methodologies used. Section 5 discusses our approach for finding the optimal K and K_h values for each query. Section 6 describes our experiments and results when EDLSI was used in the ad hoc task. Section 7 details our power normalization and BM25 ad hoc experiments and results. Section 8 discusses our relevance feedback submissions with power normalization

Fig. 1. Truncation of SVD for LSI



and BM25. Section 9 presents our conclusions.

2 Essential Dimensions of LSI

In this section we first describe LSI. We then discuss EDLSI and how it improves upon LSI. We also detail the distributed EDLSI approach we used for our TREC Legal 2008 submissions.

2.1 Latent Semantic Indexing

LSI is a well-known approach to information retrieval that is believed to reduce the problems associated with polysemy and synonymy [3]. At the heart of LSI is a matrix factorization, the *singular value decomposition* (SVD), which is used to express the $t \times d$ term-by-document matrix as a product of three matrices, a term component (T), a "diagonal" matrix of nonnegative singular values in non-increasing order (S), and a document component (D). This is shown pictorially in Figure 1, taken from [2]. The original term-by-document matrix A is then given by $A = TSD^T$. In LSI, these matrices are truncated to $k \ll \min(t, d)$ dimensions by zeroing elements in the singular value matrix S. In practice, fast algorithms exist to compute only the required k dominant singular values and the associated vectors in T and D [1].

The primary problem with LSI is that the optimal number of dimensions k to use for truncation is dependent upon the collection [4], [10]. Furthermore, as shown in [8], LSI does not always outperform traditional vector space retrieval, especially on large collections.

2.2 EDLSI

To address this problem, Kontostathis developed an approach to using LSI in combination with traditional vector space retrieval called EDLSI. EDLSI uses a convex combination of the resultant vector from LSI and the resultant vector from traditional vector space retrieval. Early results showed that this fusion approach provided retrieval performance that was better than either LSI or vector space retrieval alone [5]. Moreover, this approach is not as sensitive to the k value, and in fact, it performs well when k is small ($10 - 20$). In addition to providing better retrieval performance, keeping k small provides significant runtime benefits. Fewer SVD dimensions need to be computed, and memory requirements are reduced because fewer columns of the dense D and T matrices must be stored.

The computation of the resultant vector w using EDLSI is shown in Equation 2, where x is a weighting factor ($0 \leq x \leq 1$) and k is kept small. In this equation, A is the original term-by-document matrix, A_k is the term-by-document matrix after truncation to k dimensions, and q is the query vector.

$$w = (x)(q^T A_k) + (1 - x)(q^T A) \tag{2}$$

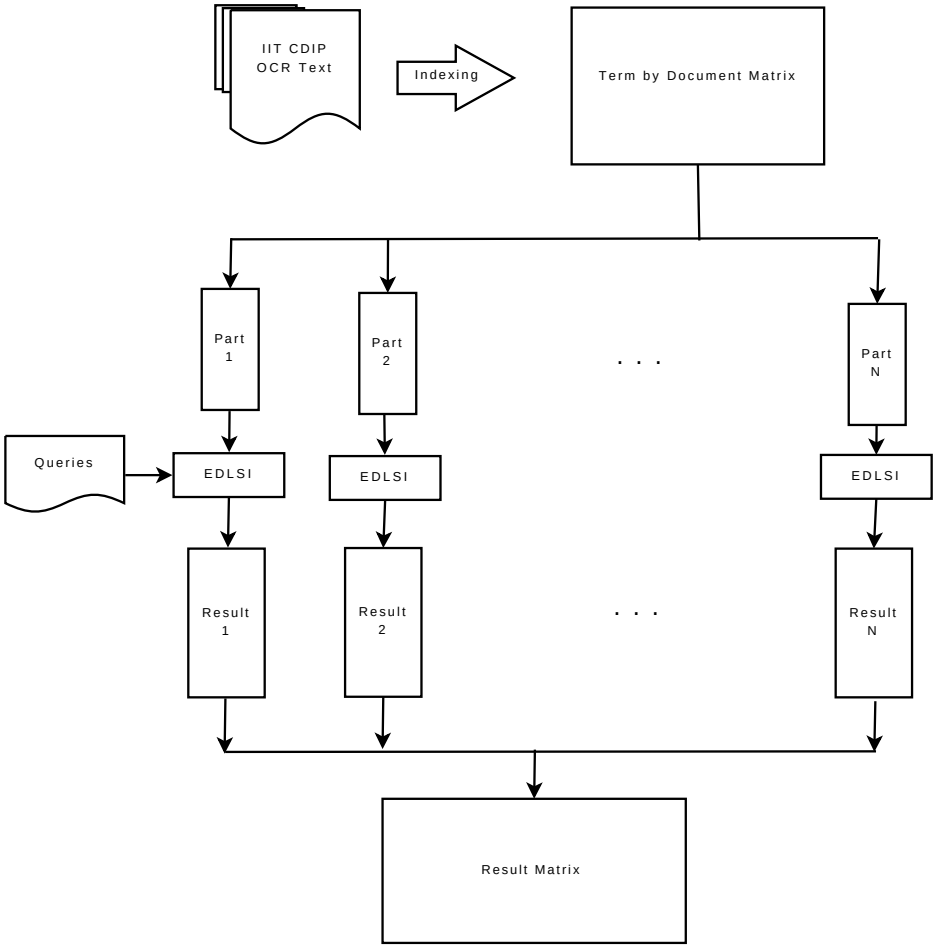
2.3 Distributed EDLSI

Although the runtime requirements, specifically the memory, for EDLSI are reduced over LSI, they are still significant, especially for a corpus the size of IIT CDIP. After indexing, we produced a term-by-document matrix that contained 486,654 unique terms and 6,827,940 documents. Our indexing system used log-entropy weighting and OCR error detection. Details can be found in [9], [6].

In order to process this large collection, we decided to sub-divide it into 81 pieces, each of which was about 300 MB in size. Each piece contained approximately 85,000 documents and approximately 100,000 unique terms (although the term-by-document matrix for each piece was approximately 85K by 487K). Each piece was treated as a separate collection (similar to [14]), and the 2008 queries were run against each piece using EDLSI. The result vectors were then combined so that the top 100,000 scoring documents overall could be submitted to TREC. The architecture model appears in Figure 2.

In Section 6 we discuss our 2008 TREC submissions using distributed EDLSI.

Fig. 2. Distributed EDLSI



3 POWER NORM

A variety of well-known and effective term weighting schemes can be used when forming the document and query vectors [17]. Term weighting can be split into three types: A $local$ weight based on the frequency within the document, a $global$ weight based on a term's frequency throughout the dataset, and a *normalizing* weight that negates the discrepancies of varying document lengths. The entries in the document vector are computed by multiplying the global weight for each term by the local weight for the document-term pair. Normalization may then be applied to the document and query vectors.

Our power normalization studies employed log-entropy weighting. The purpose of the log-entropy weighting scheme is to reduce the relative importance of high-frequency terms while giving words that distinguish the documents in a collection a higher weight. Once the term weight is computed for each document, we then normalize the document vectors using Equation 3. In this equation dtw is the document term weight, qtw is the query term weight, dc is the number of terms in the document, qc is the number of terms in the query, p is the normalization parameter, and the sum is over all terms in the query.

$$w_d = \sum_{t \in Q} \left(\frac{dtw}{dc^p} \frac{qtw}{qc^p} \right) \tag{3}$$

Power normalization is designed to reduce, but not completely eliminate, the advantage that longer documents have within a collection. Without normalization, long documents would dominate the top ranks because they contain so many terms. In contrast, cosine normalization ensures in all documents have exactly the same weight for retrieval purposes. Experiments from the TREC Legal competition in 2007 showed that power normalization using $p = 1/3$ and $p = 1/4$ results in retrieval performance improvements over cosine normalization for the 2006 and 2007 query sets [9].

4 BM25

The BM25 algorithm was introduced at TREC 3 [13]. It combines an inverse collection frequency weight with collection specific scaling for documents and queries. The weight function for a given document (d) and query (Q) appears in Equation 4. In this equation, idf_t refers to the inverse document frequency weight for a given term (Equation 5), K is given by Equation 6, tf is the number of times term t appears in document d, *qtf* is the number of times term t appears in the query Q, N is the number of documents in the collection, n is the number of documents containing t, dl is the document length (we measured this in words), adl is the average document length for the corpus (also measured in words), and b, $k1$, and $k3$ are tuning parameters.

$$w_d = \sum_{t \in Q} idf_t \frac{(k1+1)tf}{(K+tf)} \frac{(k3+1)qtf}{(k3+qtf)} \quad (4)$$

$$idf_t = \frac{N - n + .5}{n + .5} \quad (5)$$

$$K = k1 \left((1-b) + b \frac{dl}{adl} \right) \quad (6)$$

The full BM25 weighting scheme is slightly more complicated than this and requires two additional tuning parameters, but BM25 reduces to the above formula when relevance feedback is not used and when those two parameters take their standard values. Interested readers will find details in [13].

We recently compared BM25 to power normalization and determined that BM25 outperforms power normalization at top ranks, but power normalization outperforms BM25 in terms of recall after rank 300 [6]. Our submissions using BM25 and power normalization for TREC Legal 2008 are described in Sections 7 and 8.

5 Defining K and K_h

We ran a variety of experiments to determine how to identify the optimal K and K_h for each query. Our approach centered on finding a scoring threshold to use for determining optimal K and K_h values. After several false starts using the judged documents from 2006 and 2007, we determined that the collection size for judged only was insufficient for training. Thus, we used the entire collection, in conjunction with the 2006 and 2007 queries and judgment data, for training purposes.

Initially we ran the 2006 and 2007 queries against the IIT CDIP corpus and developed a pseudo submission file containing the top 100,000 scoring documents and their scores for each query. We then used these files as input and measured F1 at a variety of cutoff threshold levels. If a document score was greater than the threshold $Thres_K$ the document was considered 'relevant'; otherwise it was considered 'not relevant'. We then measured precision, recall, and F1 for each query and computed the average across all queries. We used the TREC2006 and TREC2007 relevance judgment files for computing precision, recall, and F1. All unjudged documents were considered not relevant. Again, using only judged documents did not provide sufficient training data, in our opinion. This approach provided data that were more consistent across runs.

Two methods were used for determining the optimal $Thres_K$. The first method determined the optimal F1 for each query individually by incrementing K from 1 to 100,000 and measuring F1. The document score at the optimal K was determined to be $Thres_K$ for a given query. The average $Thres_K$ for each query was used to identify the final $Thres_K$ for a query set.

The second approach involved iterating through thresholds to identify which threshold gave us the optimal F1 for the entire query set. For power normalization, thresholds from 3 to .01 were tried (in increments of .02); for BM25, thresholds of 400 to 25 were tried (in increments of 5). The maximum F1 determined the optimal $Thres_K$ value. We measured F1 to 3 significant digits. When ties for the best F1 were found, maximum recall was used as a tie breaker to determine the optimal $Thres_K$.

Both techniques for determining $Thres_K$ were used on the queries for 2006 and 2007 separately and then for 2006 and 2007 combined, for both power normalization and BM25. The optimal threshold values appear in Table 1. Because two different approaches were used, they always produced different optimal values (although sometimes only slightly different). The lower of these was used as the K threshold, and the higher was used as the K_h threshold.

6 Ad Hoc Experiments with Distributed EDLSI

In this section we discuss our distributed EDLSI experiments for the TREC 2008 competition.

6.1 Experimental Design

Our main focus during these experiments was the development and implementation of the system. Initially we hoped to treat the entire collection as a single LSI space. To accomplish this, we planned to implement EDLSI with the folding-up algorithm [11]. Folding-up combines folding-in [3], [2] with SVD updating [12], [18], [2] to maintain the integrity of the LSI space (with simple folding-in the columns of T and D generally become less orthogonal with every added term and document, respectively). We initially sub-divided the collection into

TABLE 1
Determining Optimal K and K_h

Weighting Scheme	Query Set	Method 1			Method 2		
		$Thres_K$	F1	recall	$Thres_K$	F1	recall
BM25	TREC 2006	162.4	.104	.177	190	.044	.123
BM25	TREC 2007	307.9	.130	.164	280	.054	.234
BM25	Combined 2006/2007	238.7	.117	.170	215	.040	.236
Power normalization	TREC 2006	.655	.050	.187	.800	.026	.075
Power normalization	TREC 2007	1.997	.135	.171	1.720	.061	.259
Power normalization	Combined 2006/2007	1.358	.095	.179	1.700	.032	.139

81 pieces and used traditional SVD on the first piece. We then attempted to integrate the remaining 80 pieces via the folding-up process. Unfortunately, the folding-up process is memory intensive, and we estimated that the process would run approximately 35 days; we did not have sufficient time for it to complete.

Our next approach combined distributed EDLSI with folding-up. Our design included distributing EDLSI across 8 processors, each processing about 10 chunks of data. The first chunk would perform the SVD, the remaining 9 would be folded-up. We estimated that this process would take about 10 days, and once again we did not have sufficient time.

Finally we decided to run a fully distributed EDLSI system with no folding-up. In this system, we performed the SVD on each of the 81 pieces separately, used LSI to run the queries against the SVD space, and combined the LSI results with the traditional vector space retrieval results (EDLSI). Interestingly, we were able to run this algorithm on an IBM x3755 8-way with 4 AMD Opteron 8222 SE 3.0GHz dual-core processors and 128GB RAM running 64-bit RedHat Linux in approximately 4 hours.

We used only the request text portion of each query for all runs.

6.2 Choosing the k Parameter

For any LSI system, the question of which k to use must be addressed. To consider this question, we tested our distributed EDLSI with folding-up system on a subset of the IIT CDIP corpus using the TREC 2007 queries. In this training run, we extracted all documents that were judged for any of the 2007 queries from the full IIT CDIP collection. This collection contained approximately 22,000 documents and 85,000 terms. After preliminary testing to ensure that the results for TREC Legal 2007 were similar to other collections, we tested values for k from 15 to 49 (in increments of 2) and values for x from .16 to .22 (in increments of .02). The optimal values were $k = 41$ and $x = .2$. These values are consistent with the values identified in the early EDLSI experiments described in [5].

The distributed EDLSI system (without folding-up) with $k = 41$ and $x = .2$ was run against the 2008 queries and was submitted for judging as UCEDLSIa.

One of the most interesting things about EDLSI is that it works with very few dimenions of the SVD space; thus we decided to submit a second run with an even lower k value. We expected a lower k value to provide improved results because keeping k constant while increasing the number of divisions (over the space we used for the training run) resulted in a greater number of values being used in the T_k, S_k, and D_k matrices. For instance, at $k = 41$, there are a total of 4.4×10^8 numbers being stored in T_k, S_k, and D_k for 8 partitions, but there are 18.8×10^8 numbers being stored in T_k, S_k and D_k for 80 partitions.

However, with 80 partitions and $k = 10$, there are 4.6×10^8 numbers in T_k, S_k and D_k. We speculate that this may indicate that roughly the same amount of noise has been eliminated from the system as with 8 partitions and $k = 41$. Therefore we decided to use $k = 10$ and $x = .2$ for our second submission run for 2008 (UCEDLSIb). All experiments to date with EDLSI seem to indicate that .2 is a suitable weighting factor, and [5] shows that performance does not vary much with x, even on fairly large collections.

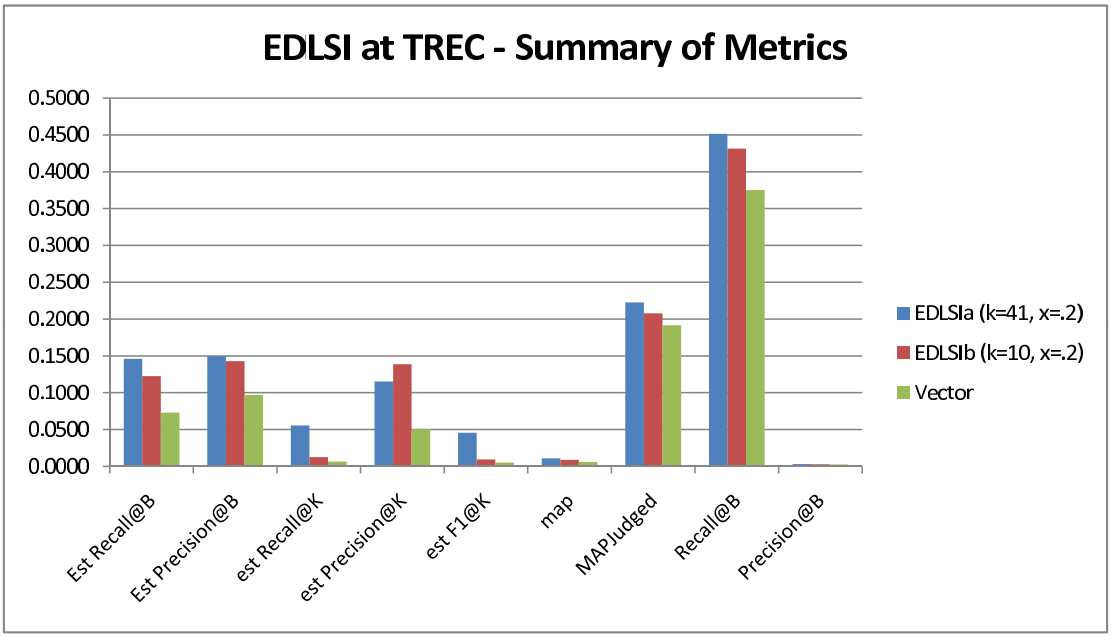
6.3 Baseline Run

Because comparison to a complete LSI run was not feasible for a collection this size, we chose instead to compare our distributed EDLSI system to the traditional vector space system that forms a part of EDLSI. Our hypothesis is that EDLSI will elicit enough term relationship information from LSI to boost the performance of the vector space model. Our vector space baseline was submitted as UrsinusVa.

6.4 Defining K and K_h for EDLSI

For UCEDLSIa, the K and K_h thresholds were set using the TREC 2007 training run via the process described in Section 5. This was not ideal, as explained above,

Fig. 3. Distributed EDLSI Result Summary



because only the judged documents were included in the EDLSI 2007 training run, not the full collection. The run was optimized using threshold values from .01 to 1.00 in steps of .01. This yielded the values $Thres_K = .207$ and $Thres_{K_h} = .227$. These same values were used for the vector baseline run UrsinusVa.

Initially we planned to use these values for UCEDLSIb run also, but a glance at the output indicated that the thresholds were too low to yield meaningful results. Almost all queries for UCEDLSIb had K and K_h values close to 100,000. As a result, we raised these thresholds, to $Thres_K = .300$ and $Thres_{K_h} = .350$ for UCEDLSIb because they seemed to produce more reasonable K and K_h values. We should have trusted the original data however (the top scoring runs at TREC Legal all had average K values at or near 100,000). A follow-up run which set $K = 100,000$ for all queries, resulted in an Estimate $F1$ at K that was 11 times larger than the submitted run (.1168 vs .0094); this would have made UCEDLSIb our top scoring run.

6.5 Results

Figure 3 shows a comparison of the two EDLSI submissions and the vector baseline using a variety of metrics. EDLSI clearly outperforms vector space retrieval across the board. Furthermore, the performance of EDLSIa, the optimized EDLSI run, is not dramatically better than the performance of EDLSIb, the baseline EDLSI run ($k = 10$), although it is slightly better on most metrics. The exception is estimated precision at F1, where EDLSIb outperforms EDLSIa. This is a direct result of the average K value submitted for the two runs. Average K for EDLSIa was 15,225; for EDLSIb it was only 1535. Interestingly, the average K for the vector baseline was even lower (802), but it still did not come close to matching the performance of either EDLSI run.

7 Ad Hoc Experiments using BM25 and Power Normalization

In this section we describe our submitted runs for BM25 and power normalization.

7.1 Experimental Design

Our experiments in 2007 focused on comparing power normalization to cosine normalization. We discovered that power normalization significantly outperformed cosine normalization, and we also identified the optimal power parameter for the 2006 and 2007 queries [9]. Subsequently, we ran some experiments that compared power normalization to BM25 [6] and learned that BM25 has better retrieval metrics (recall and precision) at top ranks, but power normalization has better recall as rank increases. Furthermore, power normalization overtakes BM25 in terms of recall at about rank 300. Thus, for 2008 we wanted to compare BM25 to power normalization. UrsinusBM25a and UrsinusPwrA are our baseline runs for BM25 and power normalization.

As noted in Section 4, BM25 requires a large number of tuning parameters. For our UrsinusBM25a baseline run we set $b = .6$, $k1 = 2.25$ and $k3 = 3$. These are the average of the optimal values for the TREC Legal 2006 and TREC Legal 2007 query sets. We used number of terms to represent document size and the average was 435; the total number of indexed documents was 6,827,940.

In section 3 we noted that power normalization also requires a parameter, the normalization factor, p. The TREC 2006 normalization factor was used for our UrsinusPwrA submission. The reason for this is described in Section 7.2.

In addition to comparing BM25 and power normalization, we wanted to see if automatic query expansion could be used to improve retrieval performance. Thus we ran training experiments using the TREC Legal 2007 queries. In these experiments we extracted the most highly weighted 5 and 10 terms from the top-ranked document, as well as the top three documents, and added these terms to the original query before rerunning the search. For both BM25 and power normalization, we required the document to have some minimal weight before chosing terms to add. Multiple document thresholds were tested, and between 0 and 30 terms were added to each query.

Results from the automatic relevance feedback tests were submitted as runs UrsinusBM25b (5 terms from the top document, with a minimum score of 200), UrsinusPwrB (5 terms from the top document with a minimum score of 1.0), and UrsinusPwrC (5 terms from the top 3 documents with a minimum score of 1.4).

We used only the request text portion of each query for all runs.

7.2 Defining K and K_h for Power Normalization and BM25

Optimal K and K_h values were identified as described in Section 5. For BM25, this resulted in a $Thres_K = 215$ and $Thres_{Kh} = 239$. These values were identified by combining the TREC Legal 2006 and 2007 values into a single run of the optimization loop. This threshold was used for both BM25 runs.

Training using both the 2006 and 2007 query sets for power normalization to find the optimal the $Thres_k$ values for the 2008 queries led to K values that were very small (we never let $K = 0$, but there were many queries with $K = 1$). Therefore, we decided to use the $Thres_K$ and $Thres_{Kh}$ values that were identified by training on the 2006 queries only, instead of using the combined values, because this produced more reasonable results for K. This leads us to conclude that the 2008 queries are more like the 2006 queries than the 2007 queries (or some combination of the two), so we also used the optimal p for the 2006 query set ($p = .36$) rather than an average of the optimal p for 2006 and 2007 ($p = .32$). The corresponding $Thres_K = .655$ and $Thres_{Kh} = .800$ values were used for the power normalization runs.

7.3 Results

Figure 4 shows a comparison of the BM25 run with relevance feedback and the BM25 baseline run. The relevance feedback run outperformed the baseline run on every statistic except estimated precision at K. This was not surprising as the average K for the relevance feedback run was almost three times the average K for the baseline run (10,168 vs. 3157) and precision decreases as additional documents are retrieved.

Figure 5 shows the results for the three power normalization runs. The results for power norm are more mixed. Although all three runs resulted in similar retrieval performance across the statistics shown, there is no run that consistently outperforms the other two. The baseline run outperforms the relevance feedback runs for estimated recall at B, estimated precision at F1, recall at B, and precision at B. PowerC adds in the top 5 terms from the top 3 documents, if the term weight is greater than 1.4 and performs nearly identically to the baseline run (PowerA) across the board. PowerC is usually less than PowerA, only slightly outperforming PowerA when the estimated precision at B is used. This suggests that PowerC is adding terms that are slightly unhelpful. PowerB does significantly better than PowerA on the competition metric (estimated F1 at K). It outperforms both PowerA and PowerC for estimated precision at B, estimated recall at K, estimated F1 at K, mean average precision (MAP), and MAP using judged documents only. The average K for PowerB is 3292, for PowerA it is 905, and for PowerC it is 850. Interestingly, although PowerB retrieves approximately three times as many documents at K on average, the precision at K is only slightly less than precision at K for PowerA and PowerC. Recall is again naturally much better due to the larger K value.

Finally, Figure 6 compares the performance of the optimal BM25 to the two best Power normalization schemes. BM25 is the clear winner. It soundly outperforms power normalization on the track metric of estimated F1 at K. There is again a three-to-one advantage in terms of documents retrieved (BM25b retrieves 10,168 on average, PowerB retrieves 3292), but BM25b manages to retrieve this large number of documents without sacrificing precision. Indeed, BM25b outperforms the power normalization schemes on all three F1 metrics. In fact, the only algorithm that happens to beat BM25b on any of the at-K based metrics is EDLSIa, which outperforms all other metrics in estimated recall at K. EDLSIa

Fig. 4. BM25 Baseline vs. Automated Relevance Feedback

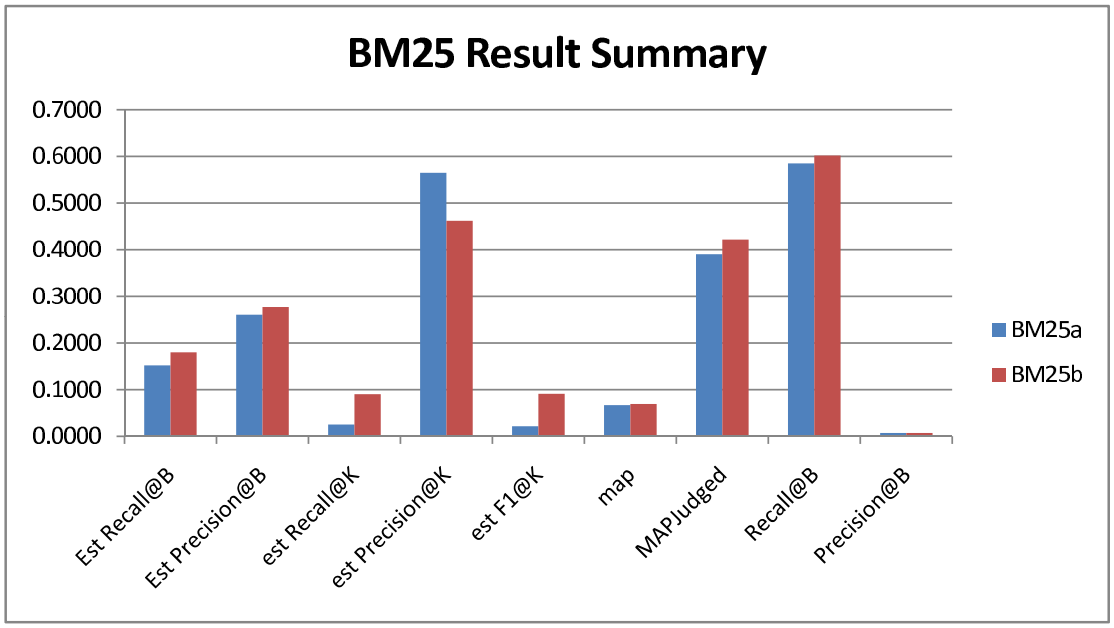


Fig. 5. Power Baseline vs. Automated Relevance Feedback

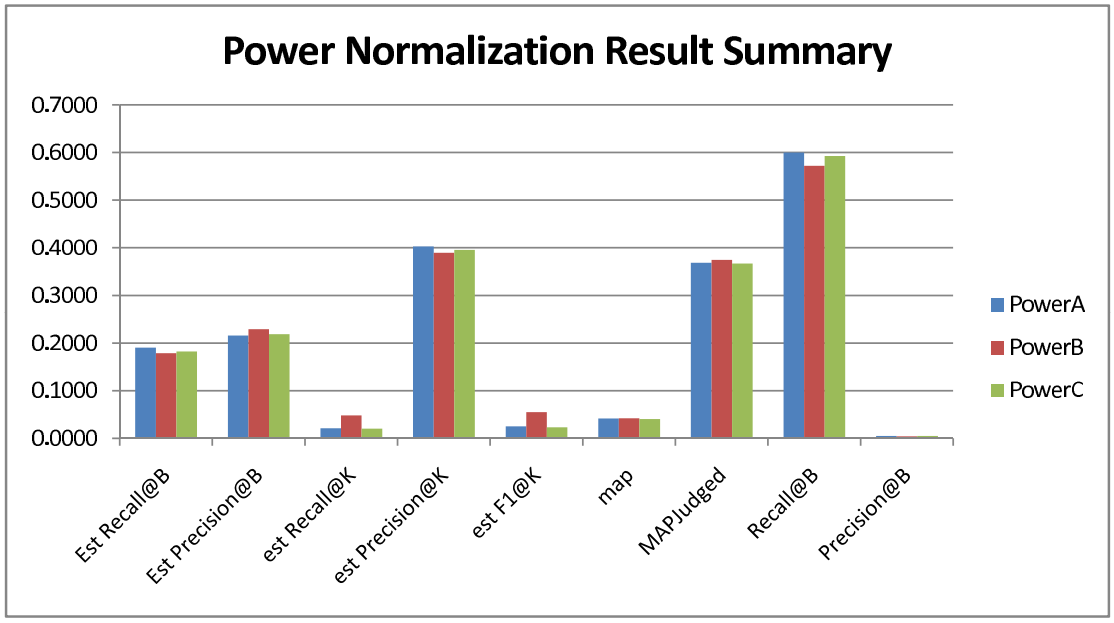
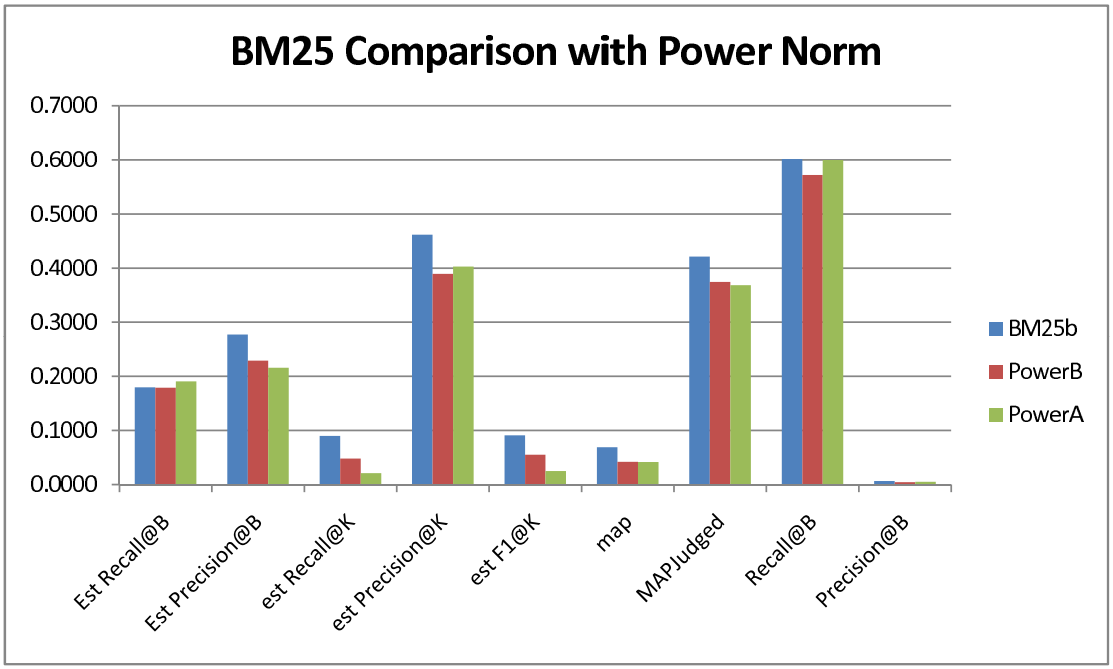


Fig. 6. Comparison of BM25 to Power Normalization



appears to accomplish this feat by simply retrieving more documents (average K is 15,225) and pays dearly in precision for this recall advantage. PowerA and PowerC manage to outperform BM25b on the statistic that was used for the competition is 2007; both of these schemes have a larger estimated recall at B.

8 Relevance Feedback runs using BM25 and Power Normalization

In this section we describe our submissions and results for the relevance feedback task.

8.1 Experimental Design

Our relevance feedback submissions leveraged our work on automatic relevance feedback. We decided to look at the top weighted terms from all documents that were judged relevant. Terms were added to the query only if their global term weight met a certain weight threshold. Some minimal training was done to determine the optimal number of terms and optimal term weight before we ran out of time. We used only the request text portion of each query for all runs.

Our relevance feedback runs are summarized below:

- UCRFBM25BL: BM25 baseline using the same parameters as UrsinusBM25a
- UCRFPwrBL: Power Norm baseline using the same parameters as UrsinusPwrA
- UCBM25T5Th5: BM25 with 5 terms over weight 5 added to each query
- UCBM25T10Th5: BM25 with 10 terms over weight 5 added to each query
- UCPwrT5Th5: Power Norm with 5 terms over weight 5 added to each query
- UCPwrT10Th5: Power Norm with 10 terms over weight 5 for each query

8.2 Defining K and K_h for Relevance Feedback

The $Thres_K$ and $Thres_{K_h}$ values from the ad hoc experiments were also used for the relevance feedback experiments.

8.3 Results

The first interesting feature that we noted is the use of the $Thres_K$ and $Thres_{K_h}$ values from the ad hoc experiments resulted in a far larger value of K for the relevance feedback runs. The basic power normalization run in the ad hoc submissions resulted in an average K value of 905; for the relevance feedback runs, the average K value was 1547. More strikingly, after the top 5 terms were added, the average K jumped to 11,880. When ten terms were added, the value increased to 13,233. For the BM25 submission the average K rose even faster. The baseline relevance feedback run had a K value of 564, which was significantly lower than the ad hoc average K (3157), but when five terms were added, this jumped

Fig. 7. Power vs. BM25 Relevance Feedback Run Comparison

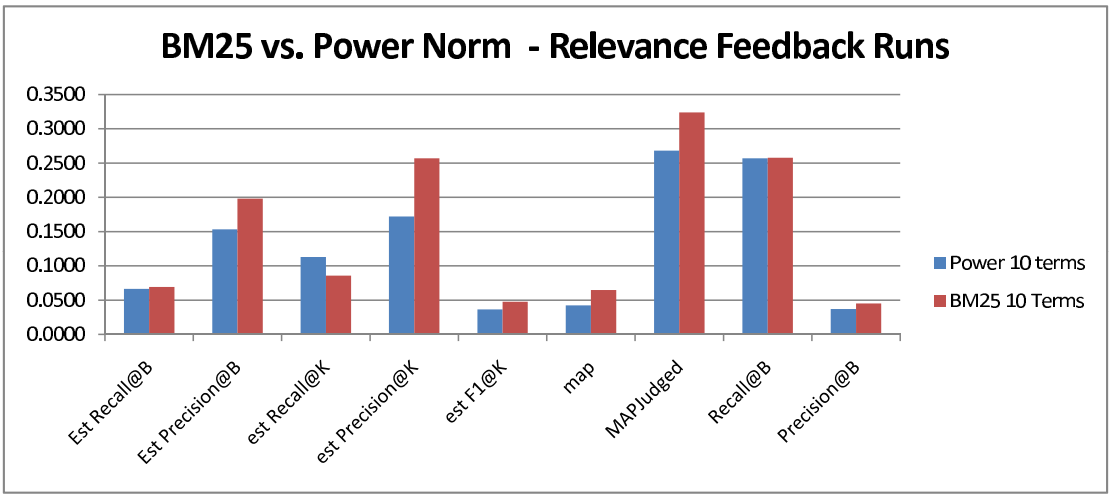
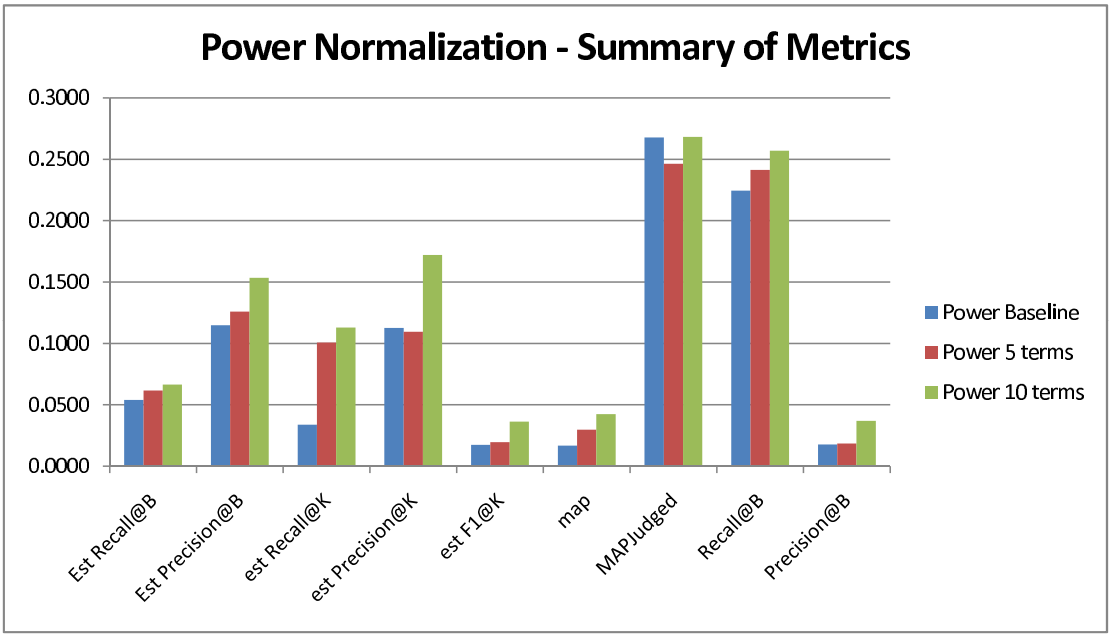


Fig. 8. Power Relevance Feedback Run Comparison - Adding additional terms

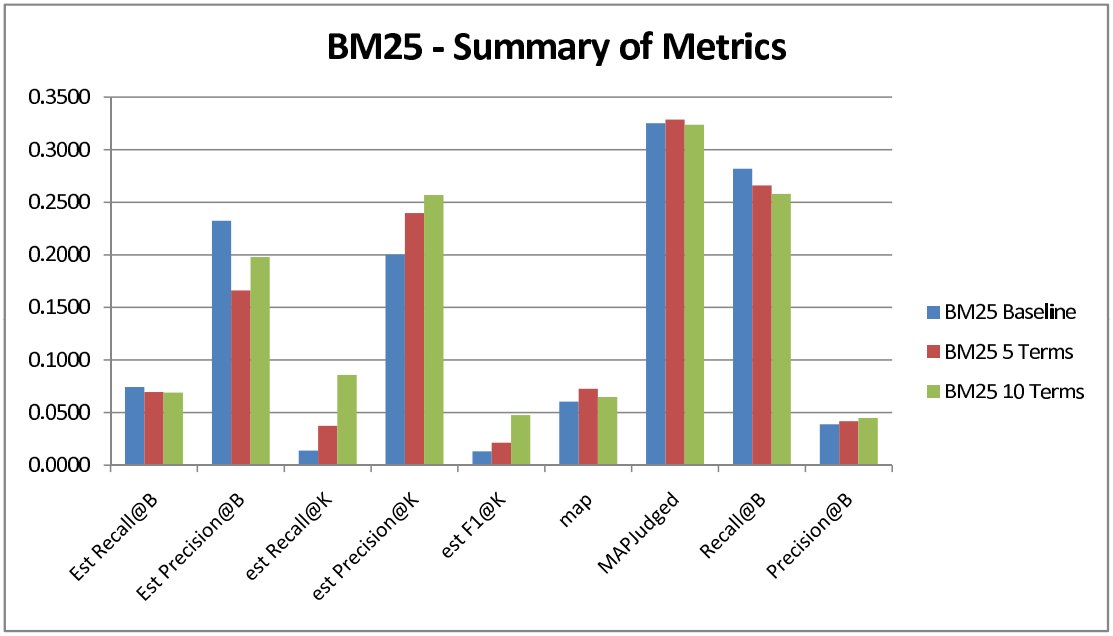


to 6517. When 10 terms were added, it took another leap to 21,406.

Figure 7 shows a comparison of the results for the top scoring BM25 and Power normalization relevance feedback runs. BM25 is clearly outperforming power normalization. It will be interesting to optimize the $Thres_K$ and $Thres_{K_h}$ values in post-hoc testing, to see if BM25 is getting its advantage by merely retrieving more documents. The strong showing on last year's metric (Estimate Recall at B) leads us to believe that BM25 will likely to continue to outperform power normalization after additional optimizations are run.

Figures 8 and 9 show the comparison for each weighting/normalization scheme as additional terms are added. We now detect a distinct difference between the power normalization runs and the relevance feedback runs. The power norm consistently does better as more terms are added. Adding five terms outperforms the baseline; adding ten terms outperforms adding only five terms. We plan post-hoc testing to take this strategy further to determine where the performance peaks and begins to degrade.

Fig. 9. BM25 Relevance Feedback Run Comparison - Adding additional terms



With BM25, however, we have a murkier situation. The baseline run actually outperforms the relevance feedback runs when the metrics from 2007 are used for comparison (Estimated Recall and Precision at B). It also strongly outperforms the other runs when Recall at B is measured. The run with ten terms added does best of the competition metrics for 2008 (F1, Recall, and Precision at K), but this could be due to the larger average K values. Perhaps most interesting of all is that the run with five terms added does best when the mean average precision metrics are observed.

9 Conclusions

Our experiments show that the Distributed EDLSI approach is promising, but it is not yet fully mature. It soundly outperformed the vector baseline approach, but it could not improve upon a basic BM25 system. Generally BM25 was the most consistent performing algorithm across all of the techniques we tested. BM25b outperformed power normalization on both the ad hoc and the relevance feedback tasks, and it outperformed Distributed EDLSI on the ad hoc task. Post analysis will be done to optimize the parameter settings for all three approaches.

References

[1] Michael W. Berry. Large-scale sparse singular value computations. *The International Journal of Supercomputer Applications*, 6(1):13–49, Spring 1992.

[2] Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):575–595, 1995.

[3] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[4] Susan T. Dumais. LSI meets TREC: A status report. In D. Harman, editor, *The First Text REtrieval Conference (TREC-1), National Institute of Standards and Technology Special Publication 500-207*, pages 137–152, 1992.

[5] April Kontostathis. Essential Dimensions of Latent Semantic Indexing (EDLSI). In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (CD-ROM)*, Kona, Hawaii, USA, 2007. Computer Society Press.

[6] April Kontostathis and Scott Kulp. The Effect of Normalization when Recall Really Matters. In *Proceedings of the 2008 International Conference on Information and Knowledge Engineering*, Las Vegas, NV, USA, 2008. CSREA Press.

[7] April Kontostathis and William M. Pottenger. A framework for understanding Latent Semantic Indexing (LSI) performance. *Information Processing and Management*, 42(1):56–73, 2006.

[8] April Kontostathis, William M. Pottenger, and Brian D. Davison. Identification of critical values in latent semantic indexing. In T.Y. Lin, S. Ohsuga, C. Liau, X. Hu, and S. Tsumoto, editors, *Foundations of Data Mining and Knowledge Discovery*, pages 333–346. Spring-Verlag, 2005.

[9] Scott Kulp and April Kontostathis. On Retrieving Legal Files: Shortening Documents and Weeding Out Garbage. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC2007)*, Bethesda, MD, 2008. NIST Special Publication 500-274.

[10] Todd A. Letsche and Michael W. Berry. Large-scale information retrieval with latent semantic indexing. *Information Sciences*, 100(1-4):105–137, 1997.

[11] Jane E. Mason and Raymond J. Spiteri. A New Adaptive Folding-up Algorithm for Information Retrieval. In *Proceedings of the 2008 Text Mining Workshop*, Atlanta, GA, 2008. SIAM.

[12] Gavin W. O"Brien. Information management tools for updating

an svd-encoded indexing scheme. Technical report, Knoxville, TN, USA, 1994.

[13] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at TREC. In *Text REtrieval Conference*, pages 21–30, 1992.

[14] Chunqiang Tang, Sandhya Dwarkadas, and Zhichen Xu. On scaling latent semantic indexing for large peer-to-peer systems. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 112–121, New York, NY, USA, 2004. ACM.

[15] Jane E. Tougas and Raymond J. Spiteri. Updating the partial singular value decomposition in latent semantic indexing. *Comput. Statist. Data Anal.*, 52(1):174–183, 2007.

[16] Jane E. Tougas and Raymond J. Spiteri. Two uses for updating the partial singular value decomposition in latent semantic indexing. *Appl. Numer. Math.*, 58(4):499–510, 2008.

[17] C.J. van Rijsbergen. *Information Retrieval*. Department of Computer Science, University of Glasgow, 1979.

[18] Hongyuan Zha and Horst D. Simon. On updating problems in latent semantic indexing. *SIAM J. Sci. Comput.*, 21(2):782–791, 1999.