Static Index Pruning for Information Retrieval Systems: A Posting-Based Approach

Linh Thai Nguyen
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616 USA
+1-312-567-5330
nguylin@iit.edu

ABSTRACT

Static index pruning methods have been proposed to reduce size of the inverted index of information retrieval systems. The goal is to increase efficiency (in terms of query response time) while preserving effectiveness (in terms of ranking quality). Current state-of-the-art approaches include the term-centric pruning approach and the document-centric pruning approach. While the term-centric pruning considers each inverted list independently and removes less important postings from each inverted list, the document-centric approach considers each document independently and removes less important terms from each document. In other words, the term-centric approach does not consider the relative importance of a posting in comparison with others in the same document, and the document-centric approach does not consider the relative importance of a posting in comparison with others in the same inverted list. The consequence is less important postings are not pruned in some situations, and important postings are pruned in some other situations. We propose a posting-based pruning approach, which is a generalization of both the term-centric and document-centric approaches. This approach ranks all postings and keeps only a subset of top ranked ones. The rank of a posting depends on several factors, such as its rank in its inverted list, its rank in its document, its weighting score, the term weight and the document weight. The effectiveness of our approach is verified by experiments using TREC queries and TREC datasets.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Index pruning, Search process.*

General Terms

Algorithms, Performance, Experimentation.

Keywords

Static Index Pruning, Document-centric Index Pruning, Term-centric Index Pruning, Posting-centric Index Pruning.

1. INTRODUCTION

Text information retrieval systems are based on an inverted index to efficiently process queries. The most important part of an inverted index is its inverted file, a file that contains posting list for each term in the text collection [1]. In general, a posting list of a term contains its posting entries (or index pointers), each in the form of <*docID*, *freq*>, where *docID* is the ID of a document that contains the term, and *freq* is its frequency in the document. For a multi-keyword query, all posting lists of query terms are retrieved from the inverted file, and document scores are accumulated for each document in the union of the posting lists, based on a specified weighting scheme. A list of documents in descending order of rank scores is presented to the user.

For a large text corpus, the inverted file is too large to fit into memory of the search server. Thus, query processing involves a lot of disk access, which increases query response time. For a text information retrieval system that has to process thousands of queries per second, it is critical to improve query processing performance.

Beside the parallel query processing approach that uses a cluster of servers to process queries, the index compression approach is widely used. The lossless compression approach uses data compression techniques to compress index data, thereby reducing the volume of data transferred from disk. The compressed index data is then decompressed in memory, and queries are processed based on the original index information. Common data compression technique used in information retrieval systems is variable length data coding [2]. In contrast, lossy compression approach opts for keeping only important information in the index, discarding other less important information [4][5][6][8][11]. Thus ranking quality of queries processed based on a lossy compressed index (i.e. a pruned index) might be affected.

In practice, a lossless compression technique can be applied on a lossy pruned index to further reduce index size. In addition, both types of compressed/pruned index can be used by an information retrieval system: a lossy pruned index is used to answer a large portion of user queries, and a lossless compressed index is used only if result quality is significantly hurt [4].

In this work, we concentrate on lossy index compression. Current state-of-the-art approaches include term-centric pruning [5] and document-centric pruning [6]. While term-centric pruning method considers each inverted list independently and removes less important postings from each inverted list, document-centric pruning considers each document independently and removes less important terms from each document. In other words, the term-centric method does not consider the relative importance of a posting in comparison with others in the same document, and document-centric method does not consider the relative importance of a posting in comparison with others in the same inverted list. The consequence is less important postings are not pruned in some situations, and important postings are pruned in some other situations.

We propose a posting-based pruning approach, which is a generalization of both the term-centric and document-centric approaches. Our approach ranks all postings and keeps only a subset of the top ranked ones, removing the others. We consider a couple of factors when ranking a posting, such as its rank in its posting list, its rank in its document, its weighting score, the normalized weight of the term, and the normalized weight of the document. Our experiments based on TREC queries and TREC datasets [22] show that posting-based pruning method outperforms both the term-centric and document-centric methods.

2. RELATED WORK

Lossless index compression techniques are well studied, for example, see Witten et al. [2][3]. Those techniques are mainly based on the fact that the frequencies of terms in documents, which are stored in the inverted index, follow a skewed distribution. In that case, variable length coding technique can be used to encode index information, consuming only a few bits for most of term frequency values. In general, this helps to reduce index size by about one-tenth. However, for large scale information retrieval systems, the compressed index is still too big to fit into memory. In addition, using a compressed index reduces time to access index data from disk, but does not reduce time to process the posting lists. Thus, using lossless index compression alone cannot significantly improve efficiency.

Lossy index compression techniques opt for discarding postings that are not informative. By removing a large number of postings from the inverted index, lossy index compression techniques not only significantly reduce index size, but also significantly reduce length of posting lists. Therefore, lossy index compression techniques can reduce both time to access index data from disk and time to process posting lists. However, as some index information is lost, lossy index compression techniques may lead to a drop in query ranking quality.

In [5], Carmel et al. introduced a term-centric approach to static index pruning. Entries in each posting list are sorted in descending order of a weighting scores. Only entries whose weighting scores are greater than a threshold value are kept in the pruned index. The threshold value can be the same for all terms (uniform pruning), or it can be different for each term (term-based pruning).

Buttcher and Clarke introduce a document-centric pruning technique [6]. Instead of posting list pruning, they propose document pruning. For each document, they keep only a small number of representative, highly-ranked terms in the pruned index. Terms in each document are ranked based on their contribution to the Kullback-Leibler divergence [16] between the document and the text collection. The intuition behind this is that those document-representative terms are powerful enough to distinguish the document from others. They also show experimentally that if the document is ranked high for a given query, it is very likely that query terms are among its representative terms. Thus indexing sets of representative terms is a good method to preserve ranking quality while reducing index size.

Other index pruning techniques (some are for distributed, peer-to-peer information retrieval systems) belong to either the term-centric approach or document-centric approach. Blanco et al. [8], and Shokouhi et al. [10], try to find terms whose posting lists can be completely removed. De Moura et al. [11] propose to index a set of representative sentences for each document. Lu and Callan [7] propose a number of methods to identify a representative term set for each document. Podna et al. [12] and Skobeltsyn et al. [13] propose to index term combinations to reduce the negative effect of posting list pruning to ranking quality. Blanco and Barreiro [9] improve the precision of term-centric pruning by considering a number of designs overlooked by the original work.

Looking at other aspect of static index pruning, Skobeltsyn et al. [14] point out that the use of results caching fundamentally affects the performance of a pruned index, due to the change in query pattern introduced by results caching. They then propose to combine results caching and index pruning to reduce the query workload of back-end servers.

3. TERM-CENTRIC PRUNING VERSUS DOCUMENT-CENTRIC PRUNING

3.1 Term-Centric Index Pruning

Term-centric pruning fits very well with the inverted index structure of information retrieval systems. As queries are processed based on inverted lists, it is natural to truncate inverted lists in order to reduce index size. Based on the inverted index structure, the “idealized, term-based” pruning technique proposed by Carmel et al. is well-formed and mathematically provable. This clearly shows that a pruned index, even though not containing all information, still can guarantee the ranking quality to some extent [5].

There are several properties that are specific to term-centric pruning. It preserves the collection vocabulary. For every term, there are always some entries in its inverted list in the pruned index. (The works of Blanco et al. [8] and Shokouhi et al. [10] are exceptions, as their work reduces the vocabulary size.) In contrast, term-centric pruning does not necessarily preserve the set of documents. As posting entries are removed, it is possible that some documents will be totally removed from the pruned index.

The fact that term-centric index pruning preserves the set of terms demonstrates its support for the possibility of all terms appearing in user queries. Due to this support, in order to guarantee the quality of top-K results for any queries, term-centric pruning must not prune any of the top-K entries in any posting list. Obviously, pruning any of these makes the pruned index unable to guarantee the top-K results of the query containing only that single term. In addition, term-centric pruning assumes (implicitly) that every term in the vocabulary is equally important. In contrast, for documents, term-centric pruning assumes that some are more important than others. This is inferred from the fact that term-centric pruning might totally removed some documents from the pruned index.

3.2 Document-Centric Data Pruning

Document-centric pruning does not make any assumption about the index structure and how queries are processed. Precisely, document-centric pruning should be considered as a "data" pruning technique instead of an index pruning technique, as what it actually does is to prune the documents, not an index structure.

In contrast to term-centric pruning, document-centric pruning preserves the set of documents, not the set of terms. While any document in the collection is represented by a subset of its terms (i.e., its set of representative terms), there is no guarantee that every term will be indexed. It is likely that there are terms that are always ranked low in any document and are removed by document-centric pruning.

The first assumption implied by document-centric pruning is that every document can be ranked first by some queries (one such query might be the query that contains all its representative terms). Due to this assumption, document-centric pruning opts for including every document in the pruned index. The second implied assumption is that terms are not equally important, and some terms can be totally removed from the pruned index.

4. POSTING-BASED INDEX PRUNING

As pointed out above, term-centric pruning prunes index elements, which are posting lists; while document-centric pruning prunes data elements, which are documents. Both approaches assume that all elements are equally important, and thus the pruned index should keep some information about every element, either they are posting lists or documents.

We first find that the decision to keep some amount of information for each posting list or document to be reasonable. Without any information about the user queries, we must assume any term can be used by users. Thus no term can be totally removed. Similarly, without any information about what users will search for, we also have to assume any document can be an answer (to some queries). Therefore, no document can be totally removed.

However, given the requirement of significantly reducing index size, it is not affordable to keep information for all posting lists and all documents. We believe that the pruned index should contain neither all terms, nor all documents, but only the most important postings, given the desired pruning level.

We suspect that non-informative terms are common in any large text collection. Non-informative terms are those terms that do not help to discriminate documents. One example of non-informative terms is a term that appears in every document, such as the term "abstract" in a collection of scientific papers. Those terms are expected to have similar weighting scores to every document. Therefore, eliminating those terms will not hurt ranking quality. Unfortunately, term-centric pruning tends not to prune any entries from the posting lists of those terms. The reason is, as entry scores are almost similar, all scores are likely to be greater than the threshold value computed by the term-based method proposed in [5].

We also suspect that there are many "rarely asked for" documents in any large text collection. A document is called "rarely asked for" if it does not appear in the top-ranked results of any real world query. In practice, users normally look at only the top 20 results, so any document that does not appear in the top-20 results of a large number of queries can be removed. Puppin et al. [17] observed that, for a collection of 5,939,061 documents and a set of 190,000 unique queries, around 52% of the documents were not returned among the first 100 top-ranked results of any query.

We propose a posting-based index pruning method. We choose neither posting lists, nor documents as our working elements. Instead, we choose postings, i.e. tuples of the form <term ID, document ID, term frequency>. Postings are contained in both index elements (as posting entries in posting lists) and data elements (as terms in documents). Also, choosing posting entries as working elements, we open the possibility of removing any document and any term's posting list from the pruned index. With this flexibility, our method is able to remove non-informative terms as well as "rarely asked for" documents.

4.1 Term Weighting

When building a pruned index, terms should not be treated equally. Non-informative terms appear in a large number of documents, results in long posting lists. However, non-informative terms do not help much in ranking documents. Thus, the pruned index should significantly prune the posting lists of non-informative terms and reserve places for other informative terms. Our posting-based pruning method assigns a weight to each term as its informativeness value. Blanco and Barreiro [8] have studied a number of term-weighting schemes for the purpose of posting list pruning. Their finding is that residual inverse document frequency (RIDF) is a good quantity to measure the informativeness of terms, among other schemes such as the classic inverse document frequency and the term discriminative value. We adopt RIDF to calculate term informativeness values. As specified in [8], the RIDF value of a term is

$$RIDF = -\log\left(\frac{df}{N}\right) + \log\left(1 - e^{-\frac{tf}{N}}\right) \qquad (1)$$

where df is the term's document frequency and N is the number of documents in the collection. As pointed out by Blanco, RIDF values can be computed efficiently.

4.2 Document Weighting

Documents in a large text collection are not equally important and therefore, in the pruned index, more terms should be kept for important documents. As for term weighting, we also assign a weight to each document in the collection to reflect how important it is. For a Web collection, the PageRank [18] or HITS [19] algorithm can be used to compute document important values. However, PageRank and HITS are not applicable for non-Web documents, as there is no link structure among documents. We adopt the approach of Buttcher and Clarke [6] for this purpose. For each document, we assign its Kullback-Leibler distance to the collection as its important value. Thus, "out standing" documents (i.e., documents which are very different from the collection) will be assigned high important values, while documents which are similar to others will be assigned low important values. Our important value for a document d is defined as

$$KLD(d \parallel C) = \sum_{t \in d} \frac{tf(t)}{|d|} \log\left(\frac{tf(t)}{|d|} \times \frac{|C|}{TF(t)}\right) \qquad (2)$$

where $tf(t)$ is the term frequency of term t, $TF(t)$ is the collection term frequency of term t, $|d|$ is the length of document d (i.e., the number of terms in *d*), and |*C*| is the length of the collection (i.e., the sum of document lengths).

4.3 Posting Ranking Function

Our static pruning method evaluates postings and assigns each a usefulness value. We then build the pruned index based on these values. Given a desired level of pruning, posting entries are selected based on their usefulness values and added into the pruned index until the pruned index size reaches its limit.

According to term-centric pruning, we should assign a high usefulness value to a posting entry which appears at the top of its posting list. According to document-centric pruning, we should not assign a high usefulness value to a posting whose term does not belong to the set of top-ranked terms of the document. In addition, as discussed above, we should assign low values to non-informative terms and "rarely asked for" documents, and vice versa. Also, we obviously want to assign high usefulness value to posting entries with high scores.

To rank postings, for each posting <*t*, *d*>, we compute the following quantities:

- *S*(*t*, *d*) = the score term *t* contributes to the rank score of document *d*.
- *RIDF*(*t*) = the informativeness value of term *t*.
- *KLD*(*d* ‖ *C*) = the important value of document *d*.
- *Rank*(*t*) = the rank of term *t* relatively with other terms in document *d*.
- *Rank*(*d*) = the rank of document *d* relatively with other documents in the posting list of term *t*.

Among the quantities above, *S*(*t*, *d*) is computed using a weighting scheme, such as the classic TFIDF weighting scheme, or the state-of-the-art BM25 weighting scheme; *RIDF*(*t*) is calculated as specified in (1); *KLD*(*d* ‖ *C*) is calculated according to (2); *Rank*(*d*) is the position of document *d* in the posting list of term *t*, where posting entries are sorted in descending order of its scores $S(t,d_i)$; and *Rank*(*t*) is the position of term *t* in document *d*, where terms are sorted in descending order of its "feedback" score [6], defined below:

$$Score_{DCP}(t) = \left(\frac{tf}{|d|}\right)\log\left(\frac{tf}{|d|}\times\frac{|C|}{TF}\right) \quad (3)$$

In this work, we use the BM25 weighting scheme [20] (given below) to calculate *S*(*t*, *d*) due to its widely use in other research works.

$$S_{BM25}(t,d) = \log\left(\frac{N-df+0.5}{df+0.5}\right)\times\frac{tf(k_1+1)}{tf+k_1\left((1-b)+b\frac{|d|}{avgdl}\right)} \quad (4)$$

where *avgdl* is the average length of documents, k_1 is set to its "standard" value of 1.2 and *b* is set to its "standard" value of 0.75.

In combination, our posting entry ranking function takes as parameters all the above quantities and returns a usefulness value. Note that we apply normalization and transformation to parameter values. First, RIDF(*t*) values are normalized so that they sum up to one. Similar normalization is applied to KLD(*d* ‖ *C*) values. Normalization step is necessary, as the range of RIDF(*t*) and KLD(*d* ‖ *C*) are different. Second, we use a sigmoid function to transform the term rank values and document rank values. This transformation is necessary, too. Using the term rank values makes it appears that the 10th term is ten time less important than the top ranked term, which does not seem right. Therefore, we use a non-linear function specified below (5) as a transform function. The parameter x_0 is used to shift the "transition" point, where the sigmoid function switches from high value state to low value state, and the parameter *a* is used to control the slope of the transition period of the function.

$$sigmoid(x) = 1 - \frac{1}{1+e^{(-x+x0)/a}} \quad (5)$$

The sigmoid function above returns a value between zero and one. Rank value close to zero (i.e., top ranked element) will be transformed to a value close to one, while other rank values will be transformed depend on two parameters x_0 and *a*. In Figure 1, we show the shape of the sigmoid function for several combinations of parameters.

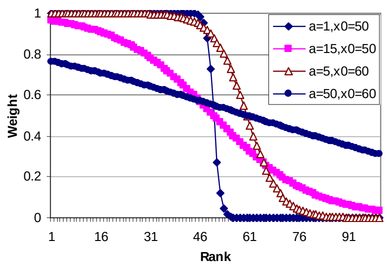

Figure 1. Sigmoid functions.

Our posting entry ranking function is given below:

$$f(\langle t,d\rangle) = S_{BM25}(t,d)\cdot\{\alpha\cdot RIDF(t)\cdot sigmoid(Rank(d)) + (1-\alpha)\cdot KLD(d\parallel C)\cdot sigmoid(Rank(t))\} \quad (6)$$

where α is a parameter taking value between zero and one.

4.4 Posting-Based Pruning versus Document-Centric Pruning and Term-Centric Pruning

We show that our posting entry ranking function generalizes document-centric index pruning method and term-centric pruning method.

If we set the value of α to zero, replace the document weighting function *KLD*(*d*‖*C*) with the unity function *u*(*d*) = 1, and set the parameter *a* of the sigmoid function to 1 so that the sigmoid function in (5) becomes a threshold function at x_0, then for each document *d*, our ranking function *f*() assigns a non-zero value to the posting entry <*t*, *d*> if *t* is ranked in the top-x_0 among all unique terms in *d*, otherwise *f*() returns a zero value. In this case, our posting-based pruning technique is equivalent to the "constant" document-centric pruning technique proposed by Buttcher and Clarke in [6], which select a fixed number of top ranked terms from each document. Obviously, the "relative" document-centric pruning technique proposed in [6] can be easily obtained from our posting entry ranking function by adjusting x_0 for each document according to its length.

Similarly, if we set the value of α to one, replace the term weighting function *RIDF*(*t*) with the unity function *u*(*t*) = 1, set the parameter *a* of the sigmoid function to 1, and set the parameter x_0 to the rank of the i-th posting entry in the posting list of term *t*

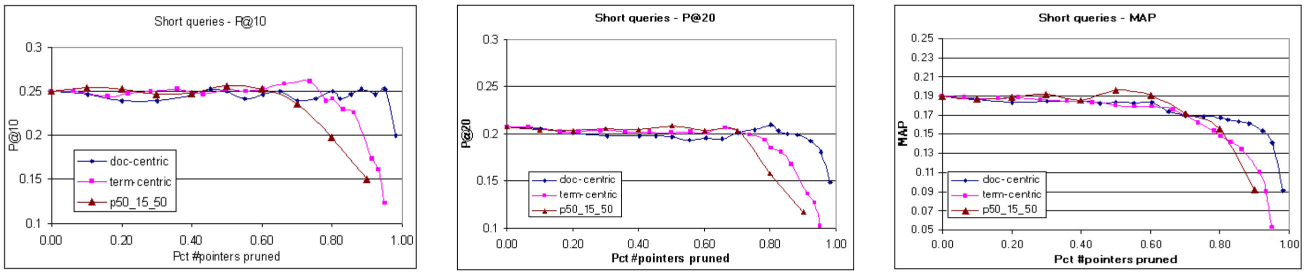

Figure 2: Querying effectiveness for short TREC queries.

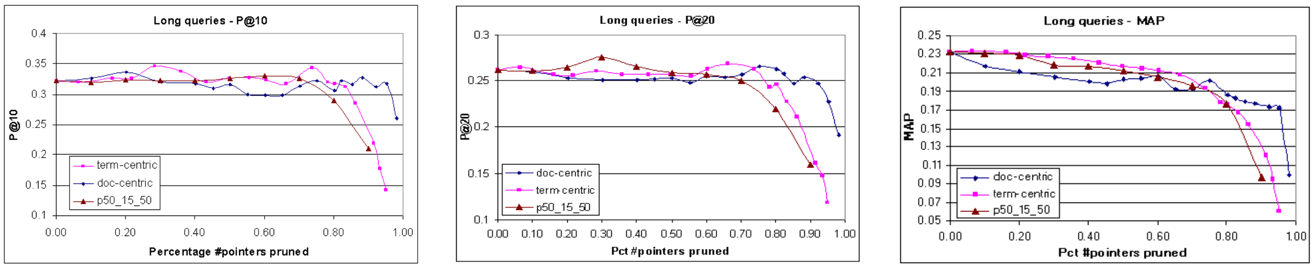

Figure 3: Querying effectiveness for long TREC queries.

such that $S(t, d_i) > \tau(t)$ and $S(t, d_{i+1}) \leq \tau(t)$, where $\tau(t)$ is the cut-off threshold proposed by Carmel et al. in [5], then our posting entry ranking function assigns a non-zero value to any posting entry selected by term-centric pruning technique, and a zero value to any non-selected posting entry.

Our posting-based pruning technique is different from term-centric and document-centric pruning techniques in several aspects. By using term weighting function and document weighting function other than the unity function, we allow the pruned index to include more information for informative terms and outstanding documents. By using a sigmoid function instead of a threshold function to transform the term and document ranks, we open the possibility that a posting entry which is ranked low in a posting list could be selected, if it is ranked high in the document, and vice versa.

5. EXPERIMENTAL SETTINGS

We use the WT10G collection as our data set. This collection contains 1,692,096 Web documents crawled from the Web. We use the Terrier platform [21] to index and rank queries, and we develop our posting-based index pruning technique based on Terrier.

We use the BM25 weighting scheme for both calculating term-document scores and query ranking. Potter stemming algorithm [23] and a standard list of stop-words are used for preprocessing documents. After stemming and stop-word removal, the vocabulary contains 3,161,488 unique terms. The un-pruned index contains 280,632,807 posting entries.

We use TREC [22] topics 451–500 as our query set. Precision is measured for each pruned index using the set of relevance judgment provided by TREC for topics 451–500. From TREC topics 451–500, we build two sets of queries: one set of long queries, wherein each query includes the title and the description fields from the TREC topic, and one set of short queries, wherein each query includes only the title field from the TREC topic.

We also implement term-centric index pruning and document-centric index pruning exactly as specified in their original works [5][6]. The only difference is that in [5], Carmel et al. used the SMART term weighting scheme for both index pruning and query ranking. We instead use BM25 term weighting scheme for query ranking, but still use SMART term weighting scheme for index pruning.

We conduct experiments to compare the effectiveness of our proposed posting-based pruning technique with the term-based, “score shifting” pruning technique proposed in [5], and the “relative” document-centric pruning technique proposed in [6]. In the next section, we report precision at 10, precision at 20, and average precision at each pruning level for each technique.

6. EXPERIMENTAL RESULTS

In Figure 2, we show the effectiveness of pruned indices for the set of short TREC queries; and in Figure 3, we show the effectiveness of pruned indices for the set of long TREC queries. Posting-centric pruned index is marked as “pXX_YY_ZZ”, where XX is the value of parameter α (percentage), YY is the value of parameter a, and ZZ is the value of parameter x_0. Figure 2 and Figure 3 show experiment results for a posting-centric pruned index with α = 50%, a = 15, and x_0 = 50. With the pruning level less than 70%, posting-centric pruning has similar performance as compare with document-centric pruning and term-centric pruning. However, for pruning level of 70% or more, posting-centric pruning is outperformed by document-centric pruning.

We turn our parameters for posting-centric index pruning technique. Table 1, Table 2, and Table 3 show experiment results for short queries of document-centric pruning technique and posting-centric pruning techniques with various combinations of

Table 1. Precision at 10 for short TREC queries of document-centric pruning technique and posting-centric pruning techniques of various parameter combinations.

Pct #posting entry-pruned	Document-centric	p50_15_50	p20_15_50	p50_35_200	p50_200_50	p80_200_1000	p50_300_1000	p80_300_1000	p80_400_1000
0	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
10	0.2458	0.2542	0.2542	0.2542	0.2542	0.2542	0.2542	**0.2563**	0.2521
20	0.2396	**0.2521**	0.2521	0.2542	0.2375	0.2333	0.2333	0.2354	0.2354
30	0.2396	**0.2458**	0.2458	0.2458	0.2292	0.2271	0.2292	0.2313	0.2292
40	0.2458	**0.2479**	0.2479	0.2271	0.2250	0.2271	0.2271	0.2250	0.2333
50	0.2500	**0.2563**	0.2542	0.2375	0.2292	0.2250	0.2312	0.2312	0.2208
60	0.2458	**0.2521**	0.2500	0.2250	0.2208	0.2104	0.2208	0.2167	0.2146
70	**0.2396**	0.2354	0.2354	0.2271	0.2396	0.2229	0.2354	0.2187	0.2187
80	**0.2500**	0.1979	0.2167	0.2021	0.2104	0.2063	0.2333	0.1979	0.2021
90	**0.2458**	0.15	0.1625	0.1625	0.1854	0.1958	0.1875	0.2021	0.2000

Table 2. Precision at 20 for short TREC queries of document-centric pruning technique and posting-centric pruning techniques of various parameter combinations.

Pct #posting entry-pruned	Document-centric	p50_15_50	p20_15_50	p50_35_200	p50_200_50	p80_200_1000	p50_300_1000	p80_300_1000	p80_400_1000
0	0.2073	0.2073	0.2073	0.2073	0.2073	0.2073	0.2073	0.2073	0.2073
10	**0.2052**	0.2042	0.2042	0.2052	0.2052	0.2052	0.2052	**0.2052**	**0.2052**
20	0.201	**0.2031**	0.2031	0.2021	0.1990	0.1938	0.1948	0.1948	0.1958
30	0.1979	**0.2052**	0.2063	0.1990	0.1854	0.1844	0.1854	0.1854	0.1844
40	0.1979	**0.2042**	0.2042	0.1917	0.1750	0.1740	0.1740	0.1750	0.1781
50	0.1969	**0.2083**	0.2073	0.1875	0.1771	0.1740	0.1781	0.1792	0.1719
60	0.1958	**0.2031**	0.2010	0.1802	0.1813	0.1813	0.1844	0.1813	0.1833
70	0.2010	**0.2031**	0.1990	0.1719	0.1802	0.1740	0.1802	0.1750	0.1750
80	**0.2094**	0.1583	0.1729	0.1573	0.1552	0.1635	0.1729	0.1635	0.1646
90	**0.1927**	0.1177	0.1229	0.1208	0.1354	0.1469	0.1448	0.1531	0.1500

Table 3. MAP for short TREC queries of document-centric pruning technique and posting-centric pruning techniques of various parameter combinations.

Pct #posting entry-pruned	Document-centric	p50_15_50	p20_15_50	p50_35_200	p50_200_50	p80_200_1000	p50_300_1000	p80_300_1000	p80_400_1000
0	0.1892	0.1892	0.1892	0.1892	0.1892	0.1892	0.1892	0.1892	0.1892
10	0.1864	0.1868	0.1871	0.1869	0.1879	0.1875	0.1878	0.1873	**0.1880**
20	0.1838	**0.1891**	0.1892	0.1889	0.1835	0.1818	0.1824	0.1835	0.1826
30	0.1846	**0.1914**	0.1913	0.1901	0.1816	0.1806	0.1807	0.1821	0.1825
40	0.1847	**0.1855**	0.1854	0.1880	0.1805	0.1822	0.1820	0.1827	0.1845
50	0.1837	**0.1958**	0.1958	0.1815	0.1834	0.1809	0.1839	0.1840	0.1809
60	0.1835	**0.1911**	0.1915	0.1804	0.1755	0.1743	0.1785	0.1747	0.1747
70	0.1693	**0.172**	0.1739	0.1698	0.1702	0.1619	0.1657	0.1692	0.1627
80	**0.1679**	0.1557	0.1571	0.1476	0.1373	0.1494	0.1575	0.1486	0.1440
90	**0.1533**	0.0919	0.1136	0.1238	0.1338	0.1404	0.1314	0.1378	0.1421

parameter values. Experiments with long queries have similar trend and therefore are omitted.

In Table 1, Table 2, and Table 3, the values in bold are the highest performance for a specific pruning level. The first observation is that posting-centric pruning is better at low and moderate pruning level, while document-centric pruning is better at higher pruning level. The second observation is that, even though posting-centric pruning is better than document-centric pruning at low and moderate pruning level, the differences are small. In contrast, at higher pruning level, the differences between the performance of posting-centric pruning and document-centric pruning are larger. In addition, none of the posting-centric pruning techniques outperforms document-centric pruning technique at high pruning level.

In all previous experiments of posting-centric pruning technique, parameters of the posting entry ranking function (6) are fixed for all posting entries. We consider the possibility of adaptively setting parameters for each posting entry, by adapting the slope of

Table 4. Performance at 90% pruning level of different posting-centric pruning techniques.

Pruning method	**Short TREC queries**			**Long TREC queries**		
	P@10	P@20	MAP	P@10	P@20	MAP
Document-centric	0.2458	0.1927	0.1533	0.3120	0.2470	**0.1731**
AP1: no term weighting, no document weighting, $\alpha = 0.5$, using two different sigmoid functions with parameters vary for each posting entry	0.2375	0.1688	0.1456	0.3020	0.2100	0.1561
AP2: similar to AP1, except that term-document scores are not used	0.1277	0.1106	0.0831	0.1660	0.1250	0.0903
AP3: similar to AP1, except that term weighting is used	**0.2604**	**0.1969**	**0.1592**	**0.3300**	**0.2500**	0.1714
AP4: similar to AP1, except that both term weighting and document weighting are used	0.2271	0.1573	0.1354	0.2740	0.1900	0.1479

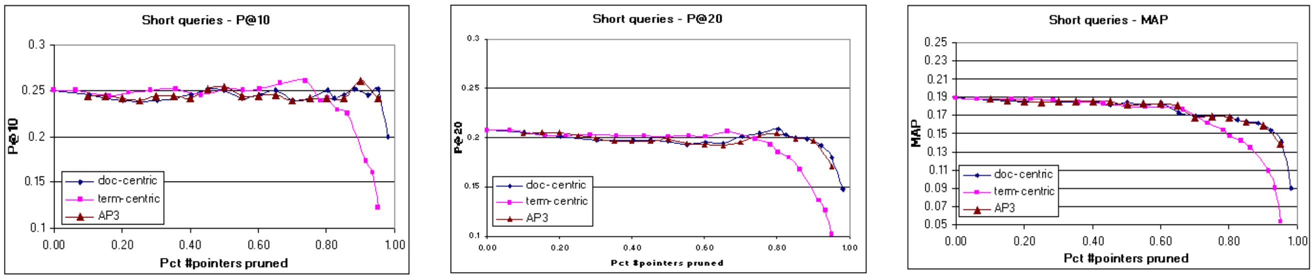

Figure 4: Querying effectiveness for short TREC queries.

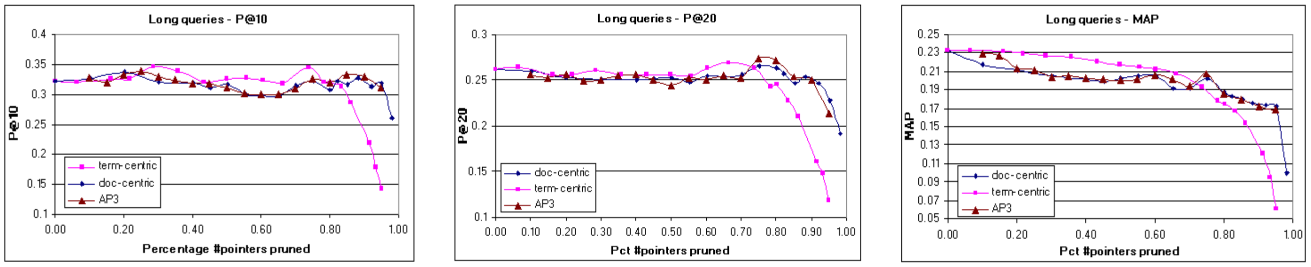

Figure 5: Querying effectiveness for long TREC queries.

the sigmoid function as follows. For a posting entry <*t*, *d*>, we use two sigmoid functions, one for the rank of term *t* in the document *d*, the other for the rank of document *d* in the posting list of term *t*. We call the former document-oriented sigmoid function, and the later term-oriented sigmoid function. For the term-oriented sigmoid function, its x_0 parameter is set according to the pruning level (i.e., if the pruning level is 90%, x_0 is equal to 10% of the length of the term posting list). Similarly, for the document-oriented sigmoid function, if the pruning level is 90%, its x_0 parameter is set to 10% of the number of unique terms in the document. For both sigmoid functions, the parameter *a*, which control the slope of the function, is set so that the function returns a value close to 1 for input value which is less than $0.9 \times x_0$ and returns a value close to 0 for input value which is greater than 1.1 $\times$ x_0. We also explore the performance of several alternatives, which may or may not use term-weighting and/or document-weighting (refer to the ranking function (6) in Section 4.3).

In Table 4, we report the performance of different alternatives at the approximately 90% pruning level. Document-centric pruning performance is included for reference. For each column, the best value is in bold. From the results reported in Table 4, we can see that:

(i) Term-document scores should be used in the posting entry ranking function (6), which is revealed by the significant differences between the performance of AP1 and AP2.

(ii) Term-weighting is useful, which is confirmed by the differences between the performance of AP1 and AP3.

(iii) Document weighting seems to be harmful, which hurt the performance of AP4, compare to the performance of AP3 and AP1.

Among all alternatives, AP3 is the best, which is only slightly outperformed by document-centric pruning for long queries according to MAP. We therefore consider it as the best among our alternatives. Below, we report its performance in comparison with document-centric pruning and term-centric pruning at various level of pruning in Figure 4 and Figure 5.

By adapting the sigmoid functions to posting entries, the performance of our posting-centric pruning technique is much better, as good as the performance of document-centric pruning technique (posting-centric pruning is better than document-centric pruning at some pruning level, while the inverse is true at other pruning levels).

7. CONCLUSIONS AND FUTURE WORK

We evaluate document-centric and term-centric static index pruning based on the WT10G corpus and TREC query sets. Based on our experimental results, term-centric index pruning is better than document-centric index pruning at low and moderate pruning level (i.e., less than 70% pruning, according to our results), while document-centric index pruning is better at higher pruning level.

We propose posting-centric index pruning technique, which ranks each posting entry (i.e., a term-document pair) based on a set of features such as the rank of the term in the document and the rank of the document in the inverted list of the term. We show that posting-centric index pruning generalizes both document-centric and term-centric pruning, and therefore, the solution space of term-centric pruning covers the solution spaces of both document-centric and term-centric index pruning. This implies that, by exploring this larger solution space, better solution can be found.

We explore the solution space of posting-centric pruning by studying a family of posting entry ranking functions. We discover that term weighting based on RIDF is useful, while document weighting based on KL-divergence is harmful. We also notice that parameters of the sigmoid function, which we use to transform the rank of a term/document to its score, should be adapted to each posting entry. Fixing these parameters makes posting-centric pruning less effective than document-centric pruning.

Other term weighting and document weighting methods are possible. We are evaluating a method of weighting terms and documents based on user queries and the PageRank algorithm applying on the graph of terms and documents. Our goal is to discover important terms and documents by analyzing the relationship among terms and documents given the context of user queries. Once the important terms and the important documents are discovered, their information is kept in the pruned index, while information about others, less important terms and documents, can be partially removed or totally discarded.

8. REFERENCES

[1] D. Grossman and O. Frieder, “Information Retrieval: Algorithms and Heuristics,” Springer, 2nd ed, 2004.

[2] I. H. Witten, A. Moffat, and T. C. Bell, “Managing Gigabytes,” Morgan Kaufmann, 2nd ed, 1999.

[3] J. Zobel and A. Moffat, “Inverted Files for Text Search Engines,” in ACM Computing Surveys, 38(2), 2006.

[4] A. Ntoulas and J. Cho, “Pruning Policies for Two-Tiered Inverted Index with Correctness Guarantee,” in Proc. ACM SIGIR, 2007.

[5] D. Carmel et al., “Static Index Pruning for Information Retrieval Systems,” in Proc. ACM SIGIR, 2001.

[6] S. Buttcher and C. L. A. Clarke, “A Document-Centric Approach to Static Index Pruning in Text Retrieval Systems,” in Proc. ACM CIKM, 2006.

[7] J. Lu and J. Callan, “Pruning Long Documents for Distributed Information Retrieval,” in Proc. ACM CIKM, 2002.

[8] R. Blanco and A. Barreiro, “Static Pruning of Terms in Inverted Files,” in Proc. ECIR, 2007.

[9] R. Blanco and A. Barreiro, “Boosting Static Pruning of Inverted Files,” in Proc. ACM SIGIR, 2007.

[10] M. Shokouhi et al., “Using Query Logs to Establish Vocabularies in Distributed Information Retrieval,” In Int'l Journal on Information Processing and Management, 2007.

[11] E. S. de Moura et al, “Improving Web Search Efficiency via a Locality Based Static Pruning Method,” in Proc. ACM WWW, 2005.

[12] I. Podna, M. Rajman, T. Luu, F. Klemn, and K. Aberer, “Scalable Peer-to-peer Web Retrieval with Highly Discriminative Keys,” in Proc. IEEE ICDE, 2007.

[13] G. Skobeltsyn, T. Luu, I. P. Zarko, M. Rajman, and K. Aberer, “Web Text Retrieval with a P2P Query Driven Index,” in Proc. ACM SIGIR, 2007.

[14] G. Skobeltsyn, F. Junqueira, V. Plachouras, and R. Baeza-Yates, “ResIn: a Combination of Results Caching and Index Pruning for High-performance Web Search Engines,” in Proc. ACM SIGIR, 2008.

[15] C. Tang, and S. Dwarkadas, “Hybrid Global-local Indexing for Efficient Peer-to-peer Information Retrieval,” in Proc. NSDI, 2004.

[16] S. Kullback, “The Kullback-Leibler Distance,” The American Statistician, 41:340-341, 1987.

[17] D. Puppin, F. Silvestri, and D. Laforenza, “Query-Driven Document Partitioning and Collection Selection,” in Proc. INFOSCALE, 2006.

[18] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank Citation Ranking: Bringing Order to the Web,” Technical Report, Stanford University.

[19] J. Kleinberg, “Authoritative Sources in a Hyperlinked Environment,” in Journal of the ACM, 46(5), 1999.

[20] S. E. Robertson. S. Walker, and M. Hancock-Beaulieu, “Okapi at TREC-7,” in Proc. of the Seventh Text Retrieval Conference, 1998.

[21] TERabyte RetrIEveR, http://ir.dcs.gla.ac.uk/terrier/

[22] TREC (WT10G, TREC-9)

[23] M. F. Porter, “An Algorithm for Suffix Stripping,” in Program, 14(3), 1980.