Locality Based Pruning Methods for Web Search

Edleno Silva de Moura and Celia Francisca dos Santos and Bruno dos Santos de Araujo
and
Altigran Soares da Silva
Federal University of Amazonas
and
Pavel Calado
IST/Inesc-ID
and
Mario A. Nascimento
University of Alberta

This paper discusses a novel approach developed for static index pruning that takes into account the locality of occurrences of words in the text. We use this new approach to propose and experiment simple and effective pruning methods that allow a fast construction of the pruned index. The methods proposed here are specially useful for pruning in environments where the document database changes continuously, such as large scale web search engines. Extensive experiments are presented showing that the proposed methods can achieve high compression rates while maintaining the quality of results for the most common query types present in modern search engines, i.e. conjunctive and phrase queries. In the experiments, our locality based pruning approach allowed reducing search engine indices to 30% of their original size, with almost no reduction in precision at the top answers. Furthermore, we conclude that even an extremely simple locality based pruning method can be competitive when compared to complex methods that do not rely on locality information.

Categories and Subject Descriptors: H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Indexing methods*; H.3.2 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms: Search, pruning

Additional Key Words and Phrases: pruning, indexing, search engines, web search, information retrieval

1. INTRODUCTION

As the number of people interested in, and capable of, using the Web grows at a fast pace, the demand for precise and fast search mechanisms becomes greater than ever. At the same time, the amount of information available on the Web is also growing fast. This situation is not likely to change in the future as ordinary

users, that once were mere information consumers, are now becoming active content producers. This can be observed, for instance, in *blogs*, whose pages are nowadays massively created [Rosenbloom 2004].

There is, therefore, a constant need to improve the efficiency of search engines, without compromising the quality of the results. To this effect, new strategies are constantly being proposed. Examples of such strategies can vary from the intelligent selection of which web pages to collect [Aggarwal et al. 2001; Cho and Garcia-Molina 2003], thus avoiding the spurious growth of the search engine database, to the application of distributed algorithms [Melnik et al. 2001] for highly time-efficient search solutions.

One alternative to improve search engine performance, and the focus of this paper, is the use of *index pruning methods* [Persin et al. 1996; Baeza-Yates and Ribeiro-Neto 1999; Carmel et al. 2001; de Moura et al. 2005]. Pruning methods can be seen as a lossy compression technique for web search engine indices, in the sense that although the search engine database is reduced, some loss of information can occur. The core strategy of these methods lies in trying to provide the advantages of data volume reduction, while maintaining the loss of information at an acceptable level.

Almost all data needed by a search engine to process user queries is stored in data structures known as *indices* [Witten et al. 1999]. Currently, pruning methods are targeted at the *frequency index*, which contains, for each term t, its frequency on each document where it occurs. This list of frequencies is called the *inverted list* of t and it is used to measure the relative importance of terms in both the stored documents and in the queries. Pruning techniques determine which term entries should and should not be present in the index. Each term entry is examined and, if it does not provide useful information, it is discarded.

However, modern search engines also deploy a *positional index*. Positional indices contain, for each term t, information about the occurrence positions of t in each document. These indices are used by search systems to process positional queries, such as phrase or proximity queries. As expected, they are the most expensive kind of index to maintain and use. Traditional pruning techniques cannot be applied to positional indices. By discarding terms independently of each other, the meaning of whole sentences is lost. Thus, dependencies between terms must be taken into consideration.

In fact, recent evidences have suggested that users are now using more phrase queries than before, reflecting their acquaintance with the way search engines work [Silverstein et al. 1999; Broder 2002; Rose and Levinson 2004]. More importantly, when analyzing the correlation between terms occurring in queries, it was found that they are often part of phrases, even if the user did not explicitly specify them as such, for instance, by putting quotes around them [Silverstein et al. 1999].

This paper discusses a novel approach developed for index pruning that takes into account the locality of occurrences of words in the text. By using it, we can achieve high compression rates while maintaining the quality of results for conjunctive and phrase queries, typical query types present in modern search engines and widely appreciated by users.

The effectiveness of our approach was first verified in [de Moura et al. 2005], where a preliminary version of a method described here was presented. Although effective, this initial method, called *LBPM* (Locality based Pruning Method) requires knowledge about the search engine ranking function and, depending on such function, it also may require the pruning process to be performed after indexing the document database. This requirement has two main disadvantages: first, it increases the index construction time, and second, it prevents direct updates from being made to the pruned indices.

To overcome these problems, we present new locality based pruning methods in which the pruning process is carried out simultaneously to index construction or updating. These are specially useful for pruning in environments where the document database frequently changes, as in large scale web search engines.

The rest of this paper is organized as follows. Section 2 discusses the motivation for pruning search engine indices and presents reference work related to this topic. Section 3 presents the general structure of our pruning approaches, while Section 4 presents details about the operation of each proposed method. Section 5 presents the test collections, query sets and performance evaluation metrics used in our experiments. Section 6 presents experiments and discusses the results obtained by our methods. Finally, Section 7 presents our conclusions and suggestions for future work.

2. RELATED WORK

A common, and successful, approach to reduce the index size on a web search engine is the use of data compression techniques. These techniques can be classified as *lossless*, where no information is ever discarded, or *lossy*, where the index size is reduced by discarding information that is deemed not useful at query processing time. Lossless methods have been extensively studied and experimented in the literature [Baeza-Yates and Ribeiro-Neto 1999; Witten et al. 1999; Bell et al. 1990; de Moura et al. 2000; Navarro et al. 2000] and can be considered as safe, since the original files can always be obtained from their compressed version. They have the disadvantage, however, of imposing upper bounds on the compression rates, which represents a practical limit to the performance improvements and reduction in storage space that can be achieved.

Lossy methods are of interest as an attempt to overcome such bounds. A trivial example of such methods is the simple stop-word removal procedure, which removes from the index entries related to very common terms. Latent Semantic Indexing (LSI) [Deerwester et al. 1990] can also be regarded as a lossy compression method. In this case, documents are represented by independent "artificial concepts" that replace several individual terms that are found to be closely related (e.g., synonyms). In this way, most of the redundancy in the characterization of documents due to use of semantically related terms is removed, resulting in a effectively compressed version of the database.

A more effective lossy compression alternative for search engine indices is the use of *pruning methods* [Carmel et al. 2001; de Moura et al. 2005]. The rationale of such methods is to remove, from the indices, entries whose potential contribution to the relevance score of the documents in which they occur is small, i.e., their

removal will have little effect in the final ranking position of these documents. The expected impact of this removal is a noticeable reduction in storage requirements, I/O operations and computational overhead. The methods we discuss in the present paper fit in this category.

Pruning methods can be classified as *dynamic* [Persin et al. 1996; Baeza-Yates and Ribeiro-Neto 1999] and *static* [Carmel et al. 2001; de Moura et al. 2005]. Dynamic methods maintain the index completely stored on disk and use heuristics to avoid reading unnecessary information at query processing time. In this case, the amount of pruning performed varies according to the user queries. That represents an advantage, since they can be better adapted to each specific query. In contrast, static methods try to predict, at index construction time, the entries which will not be useful at query processing time. These entries are then removed from the index. For this reason, static methods can be seen as lossy compression methods. Static methods offer the advantage of both reducing the disk storage costs and time to process each query. A system that uses both static and dynamic methods can also be implemented to take advantage of the two types of pruning options.

We have proposed in a previous work, a static pruning method that takes into account the co-occurrence of terms in the documents at the pruning process [de Moura et al. 2005], being useful for web search systems that allow filters on queries, such as conjunctive and phrase queries. These types of queries may have a significant impact on the design and performance of the pruning methods. For instance, to achieve a significant reduction in the time to process a phrase query, it is necessary to remove entries not only from the frequency index, but also from the positional index.

Although effective, this method has two disadvantages: (1) it uses information from the search engine ranking function to guide the pruning process, which may require a pre-processing of the document database to create non-pruned indices before actually pruning the indices, slowing down the index construction process; and (2) it prevents updates on the document database to be directly reflected on the indices. In this paper, we propose alternative locality based pruning solutions that solve these problems, while retaining the advantages of being well suited for processing typical search engine queries.

Another alternative way to reduce the search engine index sizes is by using partial document representations. This can be accomplished by using text summarization methods [Edmundson 1968], which consist in condensing a source text while preserving its information content and maintaining readability. Some text summarization methods in the literature do not focus on maintaining readability of the indexed documents, but rather on not losing information contained in the documents [Sakai and Sparck-Jones 2001; Nomoto and Matsumoto 2001; Mallett et al. 2004].

These methods can be easily adapted to act as static pruning methods, since the removal of some inverted list entries is a natural consequence of indexing document summaries, instead of the original documents. However, there is a slight, but important, difference between text summarization methods and pruning methods. Pruning methods remove inverted list entries, maintaining the original statistical information about the document database. On the other hand, when dealing with

text summarization methods, it is common not to use any previous statistical information obtained from the original database. As a consequence, important information such as the original term frequencies, the original term occurrence positions and the original term *idf* [Salton and McGill 1983] values are lost at summarization time. This type of information is desirable for obtaining query results closer to those obtained with the original database. In fact, in this paper we adapt the *Full Coverage* [Mallett et al. 2004] summarization method to work as a locality based pruning method, that is, we make sure that the original statistics are maintained. This method was proposed to summarize search engine text databases, trying to preserve the usefulness of the summary to search engine users. Since its goal is to generate summaries for search engines, we have decided to adapt it to act as a pruning method.

3. STATIC PRUNING BASED ON LOCALITY INFORMATION

Pruning methods aim at removing index entries without changing the quality of answer results provided by search engines. Some types of queries submitted to search engines, such as conjunctive queries and phrases, which are popular among users [Bahle et al. 2002; Silverstein et al. 1999; Broder 2002; Rose and Levinson 2004], require pruning methods to preserve index entries for documents that occur in the inverted lists of different terms, i.e., documents where two query terms occur together. Since most of the methods in the literature [Persin et al. 1996; Carmel et al. 2001] consider information *individually* from each term to prune the entries, the resulting pruned index may not hold this property. As a consequence, important documents may not be present in the final ranking.

Consider, for instance, a common search engine index, where the inverted lists contain the documents in priority order. This problem is illustrated in Figure 1, where the ranked list R_A for term A starts with documents 3, 15, 1, 8 and 14, and finishes with documents 2, 31, 4 and 13. The ranked list R_B for term B starts with documents 2, 1, 7, 9 and 43 and finishes with documents 25, 16, 3 and 21. Suppose that the shadowed areas correspond to portions pruned in both lists. After the pruning, a query requiring documents that contain both terms A *and* B would include document 1 in the answer, but would not include documents 2 and 3. However, these last two documents would be present in the answer before pruning. Further, documents 2 and 3 could be important to the query, since they appear on the top positions of the individual rankings for terms A and B, respectively. This example shows that a pruning solution which does not take into account co-occurrences of terms within documents may fail to preserve important results for conjunctive and phrase queries.

We here make the assumption that query terms are usually related to each other and may appear in the same context within relevant documents. In case of phrases this assumption is obviously true. In case of conjunctive queries, our experiments indicate that this assumption also holds.

Based on this assumption, we propose a family of methods that aims at predicting what set of terms may occur together in queries and use this information to preserve documents common to the inverted lists of different terms. The basis of these methods is the generic algorithm described in Figure 2.

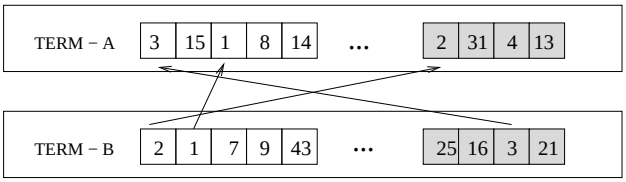


Fig. 1. Loss of information caused by pruning term inverted lists without considering co-occurrence of terms across documents.

```
1  Generic Locality Based Pruning
3  begin
4    let D be a document;
5    let S(D) be the set of fragments in D;
6    let p be the final desired percentage;
9    /* FragmentSelection */
10   S'(D) ← ∅;
11   size ← 0;
12   Setup();
13   do
14       S_important ← Important(S(D));
15       S'(D) ← S'(D) ∪ S_important;
16       size ← size + |S_important|;
17       S(D) ← S(D) − S_important;
18   until (size ≥ p|D|) or S(D) = ∅
20   /* FrequencyIndexPruning */
22   foreach term entry t_e ∈ document D
23     if (t_e ∉ any fragment from S'(D))
24       remove t_e from the frequency index
25     end
26   end
28   /* PositionalIndexPruning */
29   foreach s ∈ S(D) do
30     foreach term occurrence t_o ∈ s do
31       remove t_o
32       from the positional index
33     end
34   end
35 end
```

Fig. 2. Generic locality based pruning algorithm.

The algorithm views the document as a set of fragments. In this paper these fragments will be sentences from natural language texts, a unit that has been successfully adopted in our previous work [de Moura et al. 2005]. Nonetheless, it should be noted that our proposal is orthogonal to the definition of fragments. The algorithm, described in the following, is divided in three distinct phases: fragment selection, frequency index pruning, and positional index pruning.

Assume that we want to reduce the search engine's indices to an approximate percentage p of its original size, and let $S(D)$ be a set of fragments extracted from a document D in the document database. The first phase selects fragments from the document D until there are no more fragments to select (i.e., $S(D)$ is empty) or

until the size, in number of terms, of all the selected fragments reaches a percentage p of the size of D. Each iteration of the loop adds to the set of *significant fragments* of D, $S'(D)$, the fragment $S_{important}$ from $S(D)$, which is the most important fragment according to the heuristic used for the function *Important*. The *size* variable is then incremented with the size of $S_{important}$ and $S_{important}$ is removed from $S(D)$.

Function *Important* is at the core of the methods we propose here and determines the differences between them. Each of the proposed pruning methods follows the generic algorithm of Figure 2 relying on a different heuristic to measure the importance of the extracted fragments. As each heuristic can use different information about the fragments to measure their importance, we also have the auxiliary procedure *Setup* which is used to declare and set the variables necessary for the correct operation of function *Important*

Once the significant fragments have been selected, the algorithm starts the pruning phase, first for the frequency index and next for the positional index. For the frequency index, entries that represent the frequency of a term t in a document D are preserved only if t occurs at least once in at least one of the selected fragments in $S'(D)$. All the remaining entries are removed. In the positional index, only term occurrences in the fragments selected as important are preserved, the remaining are also removed.

4. LOCALITY BASED STATIC PRUNING METHODS

In this section, we propose three alternative methods for pruning that adopt the generic locality based pruning algorithm presented in Figure 2.

4.1 LBPM

The LBPM algorithm selects fragments by using a method proposed by Carmel et al. [2001], which we hereafter refer to as *Carmel's method*. The idea behind Carmel's method is to use the search engine's ranking to compute the importance of each inverted list entry, thus determining which index entries can be removed. This information is obtained by taking the individual terms from the collection vocabulary and submitting each one as a single-term query to the search system. A resulting document list R_t for each term t is obtained, containing the documents that include t, sorted in decreasing order of importance, according to the search engine's ranking criteria.

Carmel's method takes only the top portion of R_t to guide the pruning process. The top portion of R_t is determined according to a criterion called δ-*top*, and is denoted as $R_t(\delta)$. This criterion preserves only entries that correspond to documents in R_t whose score is at least δ times the highest score of all documents in R_t, where $\delta \in [0, 1]$. For instance, if $\delta = 0.7$, each document with a score of at least 70% of the top score will be in $R_t(\delta)$. The value of δ is typically chosen via experiments. Each index entry that represents the occurrence of a term t in a document D is removed from the index if D is not present in $R_t(\delta)$.

The LBPM algorithm starts by using Carmel's method to determine which term occurrences are individually important to each document. This information is then

used to rank the fragments of text present in a document and to determine the significant fragments. More specifically, LBPM determines the *significant terms* of each document D by computing, for each term t in the database vocabulary $\mathcal{V}$, the ranked list $R_t(\delta)$, exactly as Carmel's method. We thus define the significant terms of a document D as:

$$T(D) = \{t | D \in R_t(\delta), \forall t \in \mathcal{V}\}$$

The significant terms of a document D, denoted by $T(D)$, are all terms t whose set $R_t(\delta)$ contains D.

LBPM works according to the generic locality based pruning algorithm showed in Figure 2, using functions *Setup* and *Important* as detailed in Figure 3. The procedure *Setup* is responsible for declaring $T(D)$, the set of significant terms of the document D, and creating the global variable $T'(D)$, which is used to store the significant terms that were not already covered by the fragments selected. These variables are then manipulated in function *Important*

```
1  Setup
2  begin
3    let T(D) be the set of significant terms for D;
4    let T'(D) be a global variable with non-covered terms from T(D);
5    T'(D) ← T(D);
6  end

1  Important
2  begin
3    input: S(D) (parameter with the set of fragments in D);
4           T(D) (global variable with the set of significant terms for D);
5           T'(D) (global variable with non-covered terms from T(D));
6    output: S_common (most significant fragment from S(D));
7    S_common ← common(S(D), T'(D));
8    T'(D) ← T'(D) − {x|x ∈ S_common ∧ x
9    ∈ T'(D)};
10   if T'(D) = ∅ then T'(D) ← T(D);
11   return S_common;
12 end
```

Fig. 3. The procedure *Setup* and the function *Important* for LBPM

Function *Important* starts by calling function $common(S(D), T'(D))$, which returns the fragment in $S(D)$ that has more terms in common with $T'(D)$. LBPM considers this fragment, called S_{common}, the most important fragment in $S(D)$. Next, the terms in S_{common} are removed from $T'(D)$. The removal of terms from $T'(D)$ allows performing each fragment selection based on different sets of significant terms. Each execution of the function *Important* updates the global variable $T'(D)$ so that a next call can select a new fragment based only on the non-covered terms, i.e, those remaining in $T'(D)$. This process has the goal of preserving as many significant terms as possible in the selected fragments. If the fragments already selected cover all significant terms, $T'(D)$ becomes empty, which makes the algorithm assign $T'(D) \leftarrow T(D)$ (line 9), starting a new round of choices.

4.2 The Full Coverage Method

The *full coverage* pruning method computes the importance of each fragment by adapting a successful approach for text summarization proposed by Mallet et al. in [Mallett et al. 2004], the *full coverage summarizer*. The idea behind such summarizer is to select fragments that "fully cover" the concept-space of a document, by iteratively measuring the similarity between each fragment and the whole document, striking-out words that have already been covered. We have adopted this same idea in the full coverage locality based pruning method.

The procedure *Setup* and the function *Important* for full coverage are presented in Figure 4. The procedure *Setup* declares and initializes a query Q with the entire document D. The variable Q is global and is accessed by function *Important* Function *Important* treats each fragment in $S(D)$ as a document within the overall "collection" of D itself. The function $sim(S(D), Q)$ computes the ranking of the fragments by using the vector space model [Salton and McGill 1983]. It computes the similarity between each fragment and the current value of Q and returns the fragment with the highest similarity score. After selecting the best fragment, the terms present in S_{sim} are removed from Q. Such removal of terms aims to minimize concept redundancy in the set of sentences selected by the algorithm.

```
1  Setup
2  begin
3    let Q be a global query string;
4    Q ← D;
5  end

1  Important
2  begin
3    input: S(D) (parameter with the set of fragments in D);
4           Q (global variable with the query string);
5    output: S_sim (most important fragment from S(D));
7    S_sim ← sim(S(D), Q);
8    Q ← Q − {x|x ∈ S_sim ∧ x ∈
9    Q}; (updates Q)
10   return S_sim;
11 end
```

Fig. 4. The procedure *Setup* and the function *Important* for full coverage.

Each call to the function *Important* returns the fragment in $S(D)$ that received the highest ranking among the fragments from $S(D)$. In addition, it updates Q so that the next call to the function can perform the selection of a fragment based on the non-covered terms.

The method allows a fast generation of the pruned index, since the index is directly built from the documents and there is no need to build a previous non-pruned index. Notice that some global information, such as the term idfs, needs to be computed anyway by a previous pass through the text. However, such pass can be performed with a cost far smaller than the cost of building the whole index.

4.3 Top Fragments Method

The *top fragments* method selects the initial fragments from each document. The number of sequences selected from each document is defined according to a desired pruning rate. This strategy assumes that the first fragments of each document are those that can better describe it, i.e., it assumes that the main idea of each document is concentrated on its top fragments. This idea is also used with relative success in summarization methods [Brandow et al. 1995; Hovy and Lin 1998]. In this case, the function *Important* which selects the fragments from each document based on a desired pruning rate, takes the sentences that appear at the top of the document as the most important.

4.4 Random Method

Finally, the *random* method randomly selects k fragments from each database document. The value of k is again defined according to a desired pruning rate. This method assumes that important information is equally distributed among the document's fragments. Therefore, function *Important* selects sentences randomly from the document. While this is a quite naive approach, it serves as a practical yardstick for comparing the methods presented above.

5. EXPERIMENTAL ENVIRONMENT

In this section we present the experimental environment adopted for evaluating all the proposed methods.

5.1 Document Collections

All experiments with the pruning methods presented here were carried out on two collections. The Los Angeles Times (LAT) collection from TREC [Hawking et al. 1999] and the TodoBR collection, which is a set of documents extracted from a real search engine database. The LAT collection contains about 132,000 documents (467 MB). For this collection, the queries were selected from the ad-hoc tasks for TREC 8, for topics from 401 to 450. We have used the titles of the topics as short queries. These titles were then applied to the search system as both disjunctive and conjunctive queries. We have decided to use LAT with the goal of discovering possible differences between running pruning methods on a web collection and on a non-web collection. We have chosen only short queries since long conjunctive queries yielded very little or no results in LAT.

The TodoBR collection contains 10,077,722 web pages collected from the Brazilian web. It was chosen to present the experiments in a real case search engine environment, in order to better validate the ideas presented here. We have used in the experiments queries extracted from a log of 3 million real user queries submitted to TodoBR. This log is one of the main reasons we have chosen TodoBR collection for our experiments, since it provides useful information about search engine user preferences.

All the indices adopted, including the original, i.e., non-pruned indices, were

compressed using *Elias-δ* lossless methods for coding document numbers, frequencies and positions, as described in Witten et al. [1999]. The TodoBR index sizes after compression with *Elias-δ* are 8.4 Gb for the positional index and 1.4 Gb for the frequency index, totalizing 9.8 Gb of compressed indices. The pruning rates (compression rates) computed for the pruning methods consider these index sizes as the baseline. For instance, a pruning rate of 80% in the experiments with TodoBR means a total index size of 1.96 Gb. We have decided to combine lossless and lossy compression to show a scenario with fully compressed indices. The TodoBR indices with no compression at all require 31.7 Gb of disk storage, when representing the frequency entries with 8 bits, the document numbers with 28 bits and the occurrence positions with 28 bits. In the case of the LAT collection, the indices with no compression require 738 MB of disk storage and the index sizes after compression with *Elias-δ* require 184.5 MB.

5.2 Ranking Strategies

An important information about the experiments concerns the ranking strategies applied. As in every large scale search engine, the ranking algorithm applied on TodoBR was computed using not only the document texts, but also other auxiliary sources of evidence. For this ranking algorithm we have combined three different sources of information: document contents, anchor text concatenation and the authority value of each document. The ranking function to combine all the evidences was implemented as suggested by Calado et al. [2003].

The final similarity score of each document is given by:

$$s(d,q) = [1 - (1 - s_c(d,q)) \times (1 - s_a(d,q)) \times (1 - s_h(d,q))] \quad (1)$$

where s_c is the vector-space similarity [Salton and McGill 1983; Baeza-Yates and Ribeiro-Neto 1999] of the query q to the contents of document d (this evidence is computed based on the frequency inverted index that we prune), s_a is the similarity of q to the anchor text concatenation associated with d, and s_h is the authority value of d, computed as shown in [Kleinberg 1998; Calado et al. 2003].

We have decided to adopt a known solution proposed for ranking results on search engines in order to have more realistic results in the experiments. As search engines usually take other information to compute the ranking, it would not be fair to perform experiments just with only one source of evidence. An important detail about the ranking used is that the main cost in both disk storage and computational effort comes from the indices we are pruning. The indices for the additional evidences account for roughly 10% of the size of the main positional and frequency indices.

To check the isolated impact of the pruning methods on the textual evidence and make the experiments more thorough, we also performed experiments using such evidence alone.

For LAT, since it is not a web collection, all queries were processed using only the traditional vector space model to rank the query answers [Salton and McGill 1983; Baeza-Yates and Ribeiro-Neto 1999].

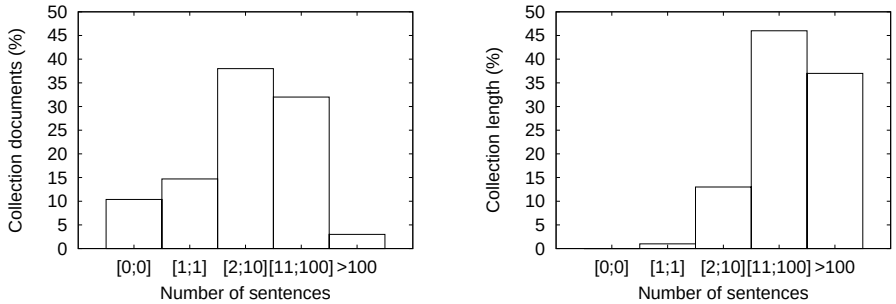
5.3 Time Efficiency

The experimental environment for evaluating the time efficiency of the systems comprises two machines running the GNU/Linux operating system, kernel version 2.4.21. The search engine server runs on a Pentium 4 2.4 GHz machine with 1 Gb of main memory, and a 200 Gb IDE disk. The search engine client runs on another machine with the same configuration. The two machines are connected directly (using a crossover cable) by an 100-megabit fast Ethernet connection.

5.4 Atomic Documents

In this work, text fragments represent sentences in natural language. Although all methods try to prune each document equally, there is a set of documents that is not affected by our methods, namely empty documents, which are returned as results of a query due to other sources of evidence (authority values and anchor texts), plus documents that have only one sentence, since the proposed methods have the feature of maintaining at least one fragment for each document. We have called these documents as *atomic documents*. Despite the fact that atomic documents have little or no textual information, they are frequent in the results and usually relevant to the user, since they might contain, for instance, images, scripts that produce useful textual information, animations and in many cases are entry points to web sites.

Figure 5(a) shows how the documents of the TodoBR collection are distributed over the number of sentences. About 10% of the TodoBR collection documents are empty and almost 15% have their content composed by only one sentence. Thus, approximately 25% of the TodoBR collection documents are atomic, i.e., are not affected by the pruning methods. This is not a problem, however, since such documents represent together less than 2% of the collection size, as illustrated in Figure 5(b).

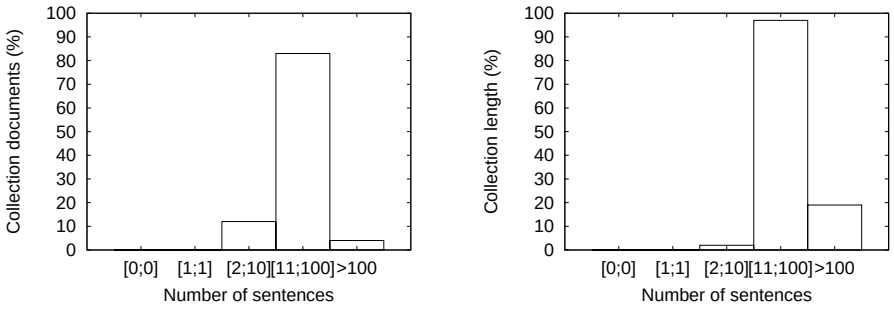


(a) (b)

Fig. 5. (a) Percentage number and (b) percentage size of the TodoBR collection documents according to the number of sentences in the documents.

Figure 6(a) shows how the documents of the LAT collection are distributed in

relation to the number of sentences. Differently from TodoBR, LAT collection does not have atomic documents. Its documents are newspaper stories published in the Los Angeles Times and, as such, they are usually well-written and have more than one sentence. The percentage size of the LAT collection documents according to the number of sentences is shown in Figure 6(b).



(a) (b)

Fig. 6. (a) Percentage number and (b) percentage length of the LAT collection documents according to the number of sentences in the documents.

Atomic documents can strongly influence the results obtained by the pruning methods, since their scores will not be affected by the pruning. Queries that originally return many atomic documents as result may have almost no change in their ranking after the pruning and, consequently, almost no loss in their result quality, independently of the pruning method performed.

The large number of atomic documents makes pruning strategies even more attractive on the Web, since methods can adopt simpler or more aggressive pruning strategies without affecting the atomic documents. That is, the final loss due to pruning is attenuated by the atomic documents on the Web.

5.5 Query Types

Search engines usually provide rich query options to their users. For instance, Google[1] allows users to include query options like inserting phrases in the query or making a query term mandatory. In general, at least three types of queries are important for search engine environments: conjunctive, disjunctive, and phrase. Conjunctive queries require that all the terms in the query appear in the documents provided as answers. For instance: searching for the documents that have the words 'goiaba' (guava) and 'fungos' (fungi) would return only documents containing these two words. A disjunctive query allows documents that have at least one of the query words in the answer. For instance: searching for documents that have the words 'manga' (mango) or 'mamao' (papaya) would return documents containing

[1] `http://www.google.com`

at least one of these words. A phrase query restricts the answers to documents that have the query terms appearing consecutively in the text and preserving the order of the query terms, for instance 'Luis Inacio Lula da Silva' (the name of the current Brazilian president) would find only documents where this sequence of words appear in the text. The main difference between conjunctive and phrase queries considered here is that in conjunctive queries the position occurrences of terms are not taken in account, unlike in phrase queries. Besides the boolean requirements, search engines sort the document answers according to their ranking criteria.

Notice that, as the web users usually are interested in the first answers provided by the systems [Silverstein et al. 1998] and not in the whole set of documents that match the query, static pruning strategies can be applied to all these query types. For instance, the phrase query "Monteiro Lobato", which asks for pages that talk about one of the most famous Brazilian writers, results into more than 930,000 answers when submitted to Google[2]. No one would expect that a user that typed this query would require to read all these pages. Hence, for all practical purposes, a pruning method that removes entries without changing the top answers would not affect most of the users.

In search engines like Google, Altavista[3] and also in TodoBR, queries seem to be taken as conjunctive by default and, because of this, conjunctive queries are very common. In fact, in TodoBR log, roughly 81.1% of the queries with more than one term are conjunctive. Phrase and disjunctive queries are much less common. For instance, in TodoBR they form respectively 18.3% and 0.6% of the submitted queries with more than one term. Nonetheless, we also will investigate these types of query due to their possible importance in other search systems.

Queries with one term can also be seen as a particular case. Our experiments show that the impact of pruning methods on queries with one term is different from the impact on conjunctive and phrase queries. In TodoBR, queries with one term correspond to 34% of the submitted queries. Figure 7 shows the distribution of TodoBR queries according to the number of terms. As we can see, more than 80% of queries submitted to the TodoBR have at most three terms.

To understand the impact of the proposed pruning methods on all query types, four sets of queries were built, one for each type. Each set contains 1000 queries randomly selected from the TodoBR log. To evaluate the impact of the proposed pruning methods on queries of different lengths (i.e., different number of terms), experiments were performed using the best performing pruning method, submitting, for each specific query length, a set of 1000 queries extracted from the TodoBR log.

For the LAT collection, we have expressed the collection's topics titles as conjunctive and disjunctive queries. Phrase queries were not used since most of them resulted in empty or very small result sets. Few queries with one term were found and, for this reason, this query type also was not included in the experiments.

[2] http://www.google.com

[3] http://www.av.com

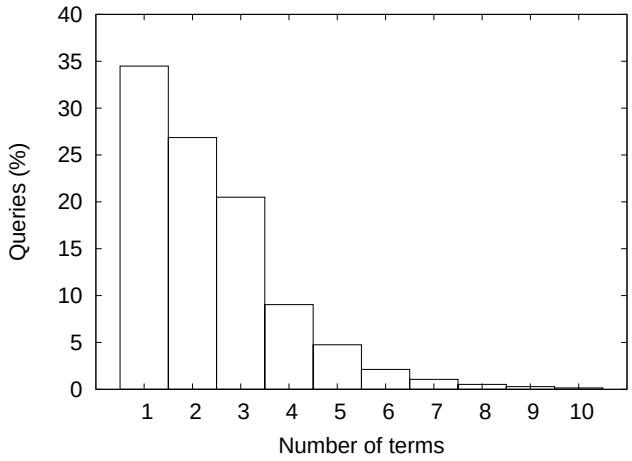


Fig. 7. Percentage number of queries as the number of terms increases.

5.6 Performance Evaluation

To study the effect of the pruning methods on the quality of the results, we evaluated two distinct aspects: the distance between the original ranking and the ranking obtained by the pruned indices and answer quality.

The ranking distance is computed here using a variation of Kendall's *tau* method, proposed in Fagin et al. [2003] and extended in [de Moura et al. 2005] to compare the top k answers of two different rankings. Kendall's *tau* method compares two top k lists and produces a score that indicates how similar these two lists are. The results vary from 0, meaning completely different rankings, to 1, meaning equal rankings. In our experiments, we use $k = 20$. Details about how to compute the Kendall's *tau* measure adopted here can be found elsewhere [de Moura et al. 2005].

The second aspect evaluated was how the pruning methods affects the ranking quality. For the TodoBR collection, we have determined the set of relevant documents for each query using the pooling method employed for the Web-based collection of TREC [Hawking et al. 1998; Hawking et al. 1999]. For constructing the pools we have evaluated the first top 20 answers of each ranking for each query. Queries were evaluated by 15 students from the Federal University of Amazonas that had no knowledge of the proposed methods. For the LAT collection, the already available standard relevancy judgments were used.

For presenting the results, we adopted the following measures: precision @5, precision @10 and MAP. See [Baeza-Yates and Ribeiro-Neto 1999] for more details about such measures. To check the significance of the results, we applied t-test [Anderson and Finn 1997] and the approach presented in [Sanderson and Zobel 2005].

6. EXPERIMENTAL RESULTS

This section presents the experimental results obtained with all the proposed methods. As a baseline for comparison, we use the results obtained with Carmel's

method, as proposed in [Carmel et al. 2001]. Experiments were divided in three parts. First, we study the similarity between rankings before and after pruning. Next, we study the impact of pruning on ranking quality. Finally, we measure the time efficiency of the pruning methods, including the time for building the indices and the time for processing queries. All the compression rates in the experiments were computed considering only the indices required to process the specific query type experimented. Thus, the positional indices were taken in account only when the results include phrase queries. Since the original Carmel's method was not designed for positional indexes, results for Carmel's method on phrases were obtained by removing all the positional entries related to frequency entries removed [de Moura et al. 2005].

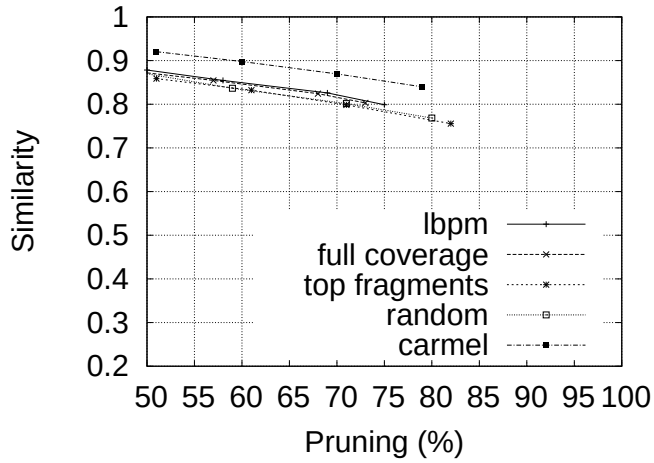
6.1 Ranking Similarity

The charts in this section show the similarity between the ranking obtained after pruning and the original rankings as the pruning rate increases. Four pruning rates of approximately 50%, 60%, 70% and 80% were tested. Note that the exact pruning rates can not be completely controlled by any of the methods, thus being impossible to achieve always the same exact rates. However, the actual obtained rates are close enough to allow a fair comparison. By applying statistical significance tests we had verified that all the differences between methods presented for the TodoBR collection are significant ($p < 0.001$), a consequence of the large amount of queries experimented.

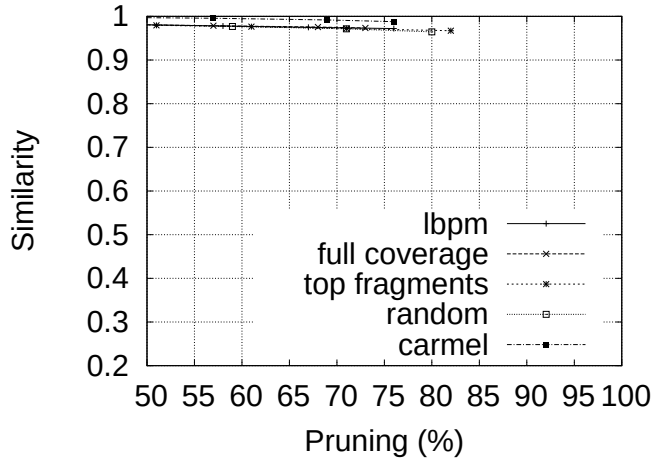
Figure 8 presents results for the TodoBR collection, on a set of randomly selected disjunctive queries when using the textual evidence alone (a) and when using the combined sources of evidence (b). In both cases, all the methods produced results very close to the original system. The similarity obtained by each method, at all levels of pruning, was superior to 0.90 in the combined ranking and superior to 0.75 when using only text. That means the ranking does not change much in all cases. This was expected for disjunctive queries, since the probability of conserving at least one of the query terms in a document after the pruning is larger than that of conserving all the query terms.

As a consequence we can say that all the experimented methods have a good performance when dealing with disjunctive queries. Notice that when using the combined evidences, all the experimented methods presented results close to 1 and the performance in all compression ratios is quite close. This is a natural consequence of the inclusion of additional non-pruned evidences in the ranking computation, such as the anchor text concatenation.

Figure 9 presents the results for the TodoBR collection obtained with conjunctive queries when using only textual evidences (a) and using the combined evidences (b). Recall that conjunctive queries represent the most popular query type. In this case, all the locality based methods (LBPM, full coverage, top fragments and random methods) yielded slightly better results than Carmel's method. One important result is that even the simpler locality based strategies, such as top fragments and random are quite competitive, which shows the importance of locality as a heuristic to guide the pruning process. Notice that full coverage and LBPM have almost similar performance when using only the textual source of evidence. This result



(a) only textual evidence

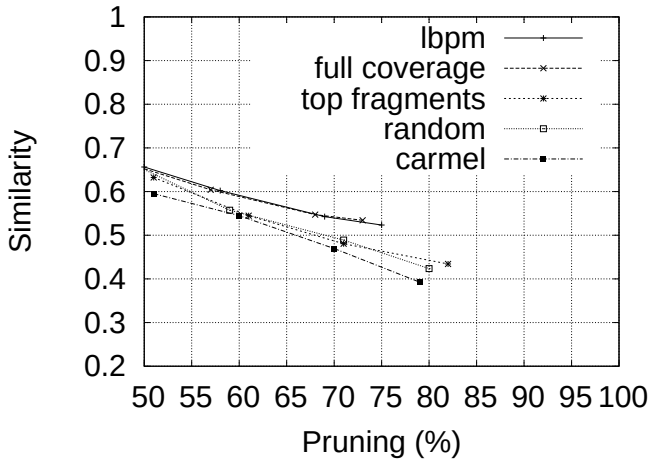


(b) combined sources of evidence

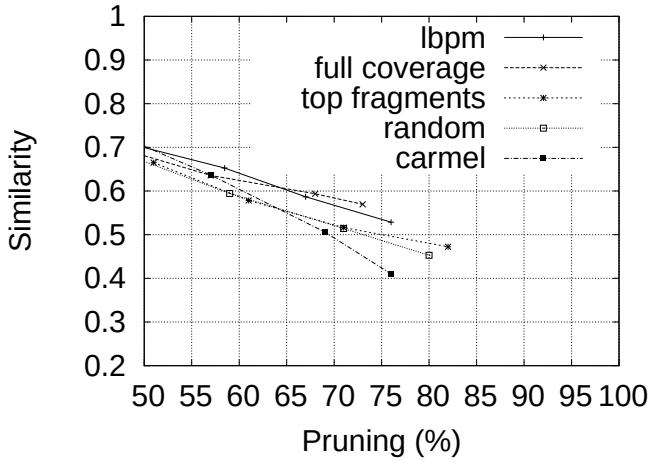
Fig. 8. Kendall's *tau* ranking similarity obtained by LBPM, full coverage, top fragments, random and Carmel's methods in the TodoBR collection, using only disjunctive queries selected randomly. Results include a ranking using only textual evidence (a) and the combined set of sources of evidence (b).

was due to the similarity between the selection algorithm of full coverage and the vector space model adopted in the ranking. When combining other evidences, the selection algorithms become more different, since LBPM uses information extracted from all the evidences, while full coverage does not change with the ranking strategy.

In Figure 10, we present the results obtained with phrase queries for the TodoBR



(a) only textual evidence



(b) combined sources of evidence

Fig. 9. Kendall's *tau* ranking similarity obtained by LBPM, full coverage, top fragments, random and Carmel's methods in the TodoBR collection, using only conjunctive queries selected randomly. Results include a ranking using only textual evidence (a) and the combined set of sources of evidence (b).

collection again when using the textual evidence alone (a) and all the combined evidences (b). In this case, both the positional and frequency indices are pruned and included in the experiments. Notice that again all the locality based methods performed quite close to each other. The worse results obtained by Carmel's method in this case were expected, since it was not designed for pruning positional indices.

While we believe the inclusion of Carmel's method in this case is not fair, we believe it is important to investigate its behavior in this scenario, since it is the best baseline we found for static pruning in literature.

In general, the similarity levels obtained with phrases are better than the ones obtained with conjunctive queries, since a phrase, which usually occurs on a sentence, is more likely to be preserved than a conjunctive query, which can be distributed among many sentences.

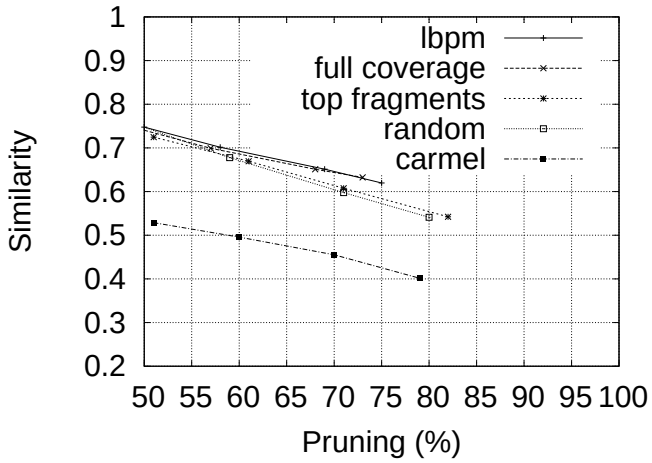
We also performed experiments including single term queries. The results obtained are quite similar to the ones obtained for disjunctive queries, with Carmel's method being slightly superior when using only textual evidences and all the methods obtaining similar performance when combining evidences.

In summary, results for the TodoBR collection show that the locality based pruning methods indeed produce competitive results when compared to Carmel's method for all query types. The results obtained for disjunctive queries and single term queries show that all the experimented methods are very suitable for this query type, yielding results almost equal to the original ranking even for 80% compression rate when using combined sources of evidences.

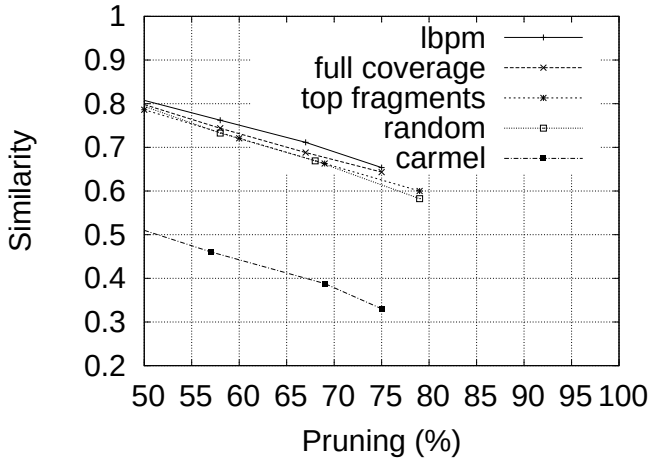
When considering conjunctive queries and phrases, which represent together more than 65% of the queries submitted for TodoBR, the locality based methods presented slightly superior performance. This is important not only because these query types are frequent, but also because these are cases in which all the experimented pruning methods exhibit the worse results in ranking similarity.

The fact that the results obtained by the locality based methods are close can be explained in part by the number of atomic documents (Section 5.4) present in the query results used in the experiments. For instance, more than 40% of the query results of each conjunctive query tested was composed by atomic documents. This suggests that preserving atomic documents may be an interesting additional heuristic for pruning in Web collections. In fact, other methods, such as Carmel's method, could be modified in order to preserve such entries when pruning. In our experiments with TodoBR queries, Carmel's method preserved only 30% of the atomic documents of the original conjunctive queries at roughly 80% pruning rate in the experiment with combined sources of evidence, while the locality based methods preserved 100% in all compression rates.

Another important information about the pruning methods is the impact that they have over queries as the number of query terms increases. In Figure 11, we present the similarity levels obtained by the five methods in the TodoBR collection at several pruning rates, using sets of 1000 queries, each with a specific number of terms. We have experimented using queries with at most five terms, which correspond to 95% of the queries tested. As expected, for all the methods the similarity level falls rapidly as the number of terms increases. However, locality based methods are less sensible to the increasing in the number of terms, which is a consequence of the fact that Carmel's method was not designed to deal with several terms in conjunctive and phrase queries. This can be illustrate by looking back at the example in Figure 1. The situation depicted in that figure deteriorates with the increasing in the number of terms, since the likelihood of missing important terms grows.



(a) only textual evidence



(b) combined sources of evidence

Fig. 10. Kendall's *tau* ranking similarity obtained by LBPM, full coverage, top fragments, random and Carmel's methods in the TodoBR collection, using only phrase queries selected randomly. Results include a ranking using only textual evidence (a) and the combined set of sources of evidence (b).

It is worthy noticing, however, that the similarity levels obtained with lpbm for the pruning rates of 50% and 60% are acceptable, especially for at most three terms, which corresponds to approximately 82% of the queries in the TodoBR log.

The results with ranking similarity over TodoBR collections suggests the use of an adaptive pruning strategy, varying the pruning rate according to the type of the

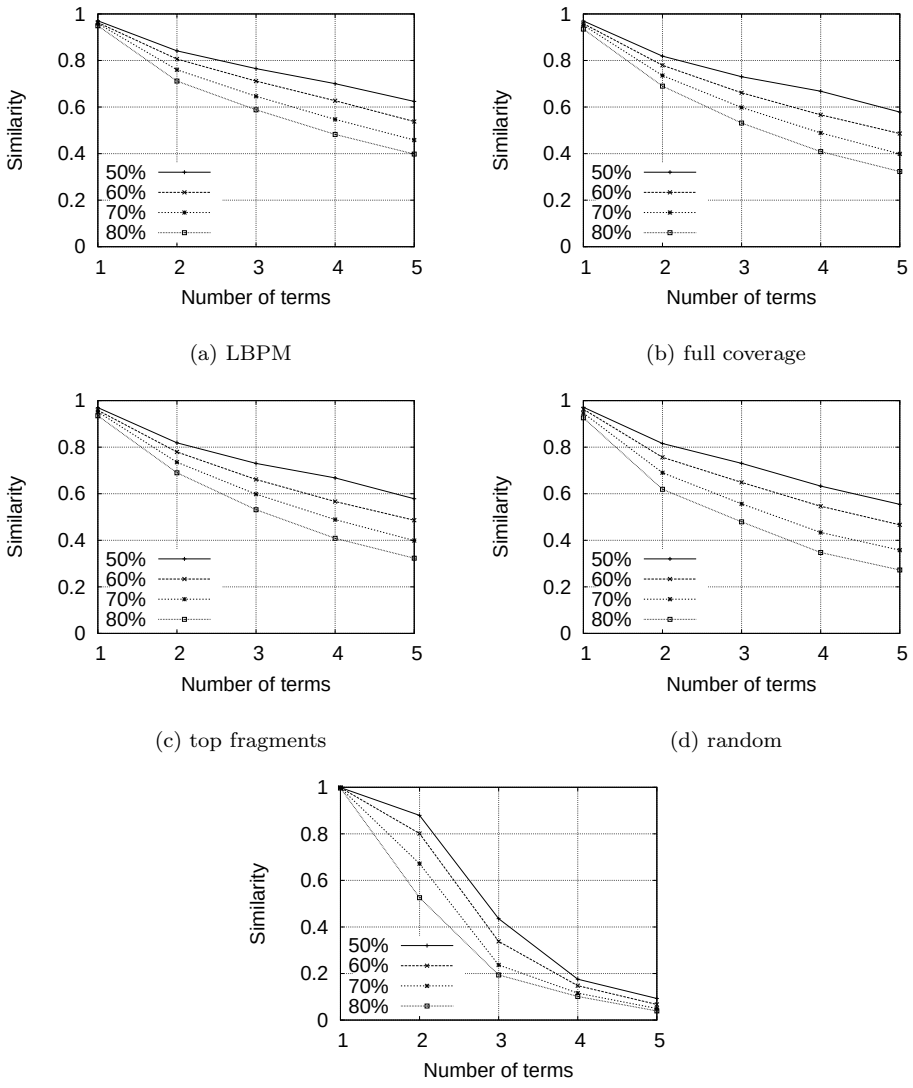
query and its number of terms. In such a strategy, instead of simply removing the index entries, we could create a cascade indexing, with the inverted lists entries splitted in different internal partitions, each one corresponding to a pruning rate. The amount of partitions used for each query would be then selected according to the query type and number of query terms. In this case, the final pruning rate would be a function of the accepted ranking similarity levels expected for the worst case query type. This is an approach we leave for future work.

Figure 12 presents results for the LAT collection using 50 short queries extracted from topic titles and submitted as disjunctive queries and conjunctive queries. For disjunctive queries, Figure 12(a), we observe that results are slightly different from those obtained for the TodoBR collection. In this case, Carmel's method obtained better results for disjunctive queries with a significant difference. Further, results produced by the more naive methods degrade considerably as the pruning level increases. For instance, at an 80% pruning level in the TodoBR collection, the top fragments method yielded a similarity level of 0.77, using only textual evidence. On the other hand, at the same pruning level in the LAT collection, the top fragments method yielded a similarity level of only 0.58.

In general, the similarity levels obtained for the LAT collection are smaller than those obtained for TodoBR. One of the reasons for this difference is that LAT does not have atomic documents. Another reason is that many terms in LAT queries have very short inverted lists, i.e., they are rare terms, unlike the terms in the TodoBR query log.

Another reason for the different results may be the disparity in the collection sizes. We created three random samples of TodoBR, each of them with the same size as of LAT. We repeated the experiments using only textual evidence in these three samples. Regarding the relative performance of the experimented methods, the conclusions were exactly the same obtained for the whole TodoBR collection, with Carmel's method being the best method for disjunctive queries and the locality based methods being better for conjunctive and phrase queries. For instance, at 50% pruning rate, LBPM achieved an average similarity of 0.48 for conjunctive queries, 0.88 for disjunctive queries and 0.57 for phrase queries, while Carmel's method achieved 0.28 for conjunctive queries, 0.97 for disjunctive queries and 0.31 for phrase queries. However, as in LAT, the Kendall's *tau* similarity values are also smaller than in TodoBR, which suggests that the collection size may also affect the static pruning quality.

We have also experimented using the TREC collection after removing rare terms from queries and the overall results obtained were slightly better for all methods, specially for the locality based ones. For instance, removing the queries with terms that occur in less than 200 positions, the comparative results were quite close to the ones obtained using the TodoBR collection, with the locality based methods being slightly better for conjunctive queries and slightly worse for disjunctive. This behavior is a consequence of the way Carmel's method prunes inverted lists of rare terms, preserving their entries while LBPM and the other locality based methods tend to prune more entries of lists of rare terms. For instance, while at roughly 70% pruning rates the Carmel's method keeps 96.5% of the entries for terms with lists with less than 200 elements, LBPM keeps only 46% of such entries. The removal of



(a) LBPM

(b) full coverage

(c) top fragments

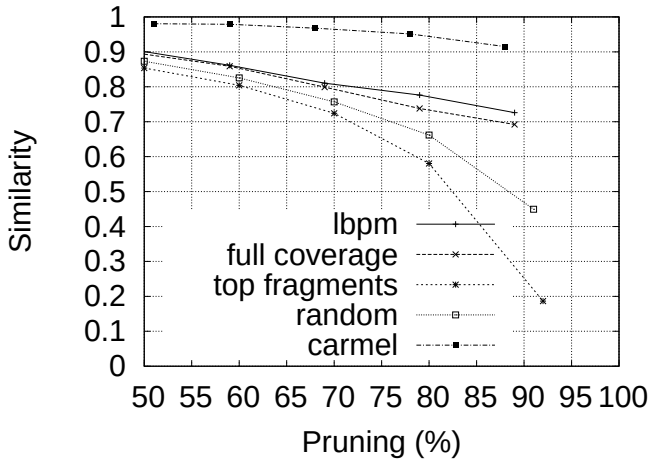
(d) random

(e) carmel

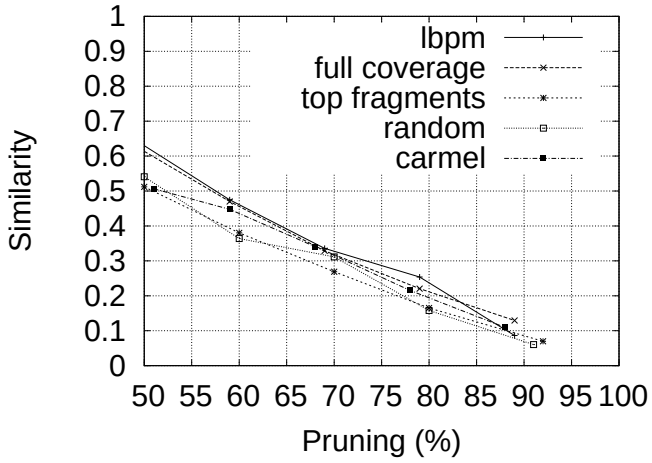
Fig. 11. Kendall's *tau* ranking similarity obtained by the five experimented methods in the TodoBR collection according to the number of terms in the queries.

rare terms with less than 200 occurrences from TodoBR samples did not change the relative results, but again the larger improvements were obtained by the locality based methods.

These results suggest locality based methods can be competitive even for TREC if processing rare terms is done as a special case. This strategy would not cause a significant loss in compression, since these terms have small inverted lists.



(a) disjunctive



(b) conjunctive

Fig. 12. Kendall's *tau* ranking similarity obtained by LBPM, full coverage, top fragments, random and Carmel's methods in the TREC collection, comparing the top 20 results. Figure (a) and (b) show the results for, respectively, disjunctive and conjunctive queries.

6.2 Ranking Quality

In order to illustrate the impact of changes in the ranking when using the investigated methods, we have performed further experiments with the TodoBR collection by submitting 40 queries extracted from the TodoBR logs. We randomly selected 14 queries with one term, 21 conjunctive queries and 5 phrases to follow the distribution of queries in the whole log. Query results were evaluated by the group of

15 students we mentioned in Section 5.6. The average number of answers in each query pool was 33 and the average number of relevant answers was 17.3.

Figure 13 shows the average precision as the pruning rate increases, when using pruned indices generated by our methods and Carmel's method. Average precision is computed by taking the average results of the eleven precision points for each ranking. LBPM values are closer to those obtained by the original indices (i.e., at 0% pruning) for pruning rates up to 70% and is the best method in terms of average precision. The remaining locality based methods present results closer to each other. These results reinforce the results obtained with ranking similarity experiments, indicating that the locality is an useful heuristic for static pruning.

We applied significance tests and the conclusion is that most differences from LBPM and other methods are all significant up to 70% compression rate, the exception is 50% compression rate against random. On the other hand, differences among all the other experimented methods is not significant, thus the performance results of these methods can be considered as a tie.

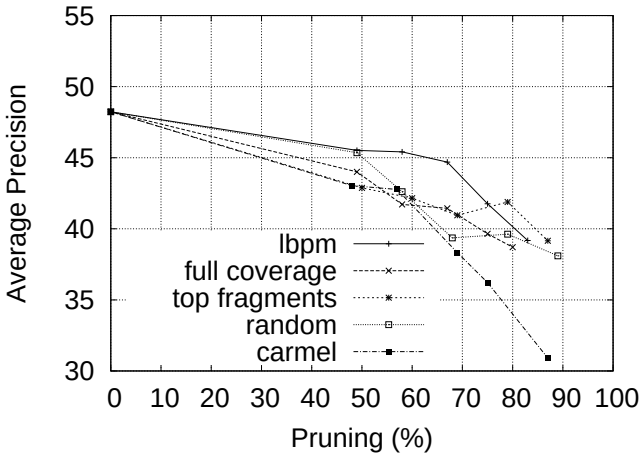
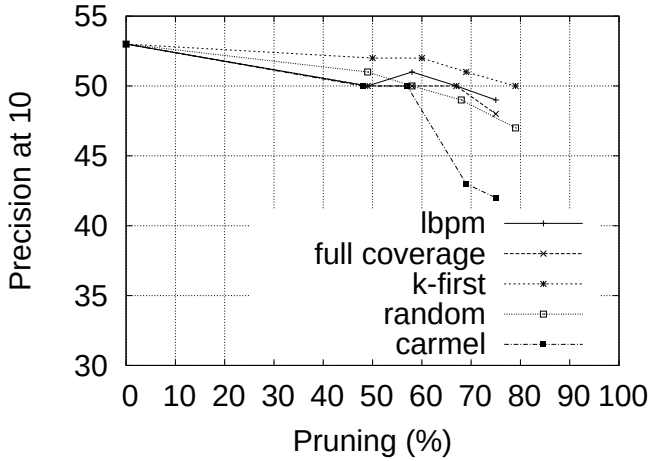


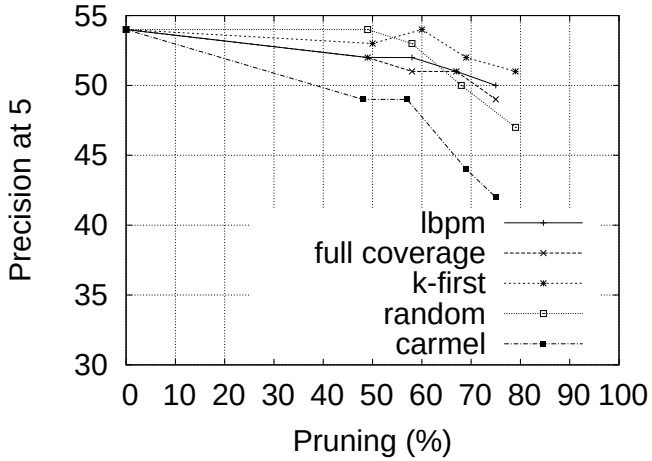
Fig. 13. Average precision obtained when processing queries with indices pruned by LBPM, full coverage, top fragments, random and Carmel's methods at different pruning rates.

Another important ranking quality metric for search engines is the precision of the methods at the top results. Figure 14 shows the performance of the methods when considering the P@10 (a) and P@5 results (b). Applying significance methods, the results can be considered as a tie between all the methods, except for Carmel's method at all compression rates greater than 70% in both cases. Another important observation is that all the locality based methods give precision results close to the system without pruning. For instance, the relative difference between the system without pruning and the system pruned at a rate 80% with top-sentences is smaller than 2% in P@10.

These experiments reinforce that results obtained by locality based methods are extremely competitive. Precision values obtained are also slightly higher for top sentences, indicating that it is better to preserve top results when compared to the random heuristic. Previous results with summarization methods have also indicated that selecting the top sentences of a text may be a good strategy for capturing the most important terms of a text, since authors tend to present the main ideas of a text in the first sentences [Hovy and Lin 1998; Edmundson 1968].



(a) precision @10



(b) precision @5

Fig. 14. P@10 (a) and P@5 obtained when processing queries with indices pruned by LBPM, full coverage, top fragments, random and Carmel's methods at different pruning rates.

These results allow us to conclude that, although the pruning process causes changes in the query rankings, as shown by Kendall's *tau* similarity, changes in precision are, in fact, smaller. In practice, this implies that we can significantly reduce search engine index storage costs without compromising the users' satisfaction with the results. For instance, results in P@10 and P@5 obtained by the top fragments method is almost the same obtained by the search engine without pruning.

As the number of queries in the ranking quality experiments is small and considering that the main goal of a pruning method is to not change the search results of the original system, the most important conclusions about the investigated methods are the ones obtained in the ranking similarity experiments performed in the previous section.

6.3 Time Performance

One of the advantages of using methods based only on the locality (top fragments and random) is that their implementation is simple and is not affected by the search engine ranking method. Further, the index can be implemented requiring no previous indexing step, which allows an speed up in the indexing times. For instance, Figure 15 shows the time necessary to prune the TodoBR indices at several pruning levels using the methods based only on locality. Time is here expressed as a percentage of the time necessary to create the non-pruned index and consider only the time necessary to prune the frequency indexes. As it can be seem, there is a significant gain in performance. The final index construction time for these two methods were almost proportional to the reduction achieved in the final index size.

Efficient implementations of the Carmel's method, full coverage and LBPM may also result in indexing construction speed up when compared to no pruning. In case of Carmel's, and consequently LBPM, such time depends on the possibility of estimating the importance of each entry in the inverted list at index construction times. For instance, it is easy to perform such estimation when using only the vector space model in the ranking. On the other hand, when adding other evidences, such us anchor text information and authority, this estimation can be harder and a preliminary indexing step may be required. In any case, the final indexing times will be superior or close to the ones obtained by the simpler methods, i.e., top fragments and random. We also performed experiments including the time to prune the positional indexes and the conclusions about the relative performance of the methods was the same.

Another important advantage of pruning is the reduction in the query processing times. Figure 16 shows the results obtained when processing queries at different pruning rates using LBPM in TodoBR collection. Time is given as a percentage of the time for processing the queries using the original non-pruned indices. Notice that the pruning method has reduced only the size of positional and frequency indices constructed for the text in the documents' body. Other index components of the search engine, such as the indices for anchor text, were not pruned.

Obviously, as the pruned indices are reduced, the time for processing queries is also reduced. This reduction followed an almost linear behavior in our experiments. However, it is important to notice that the impact of pruning on query

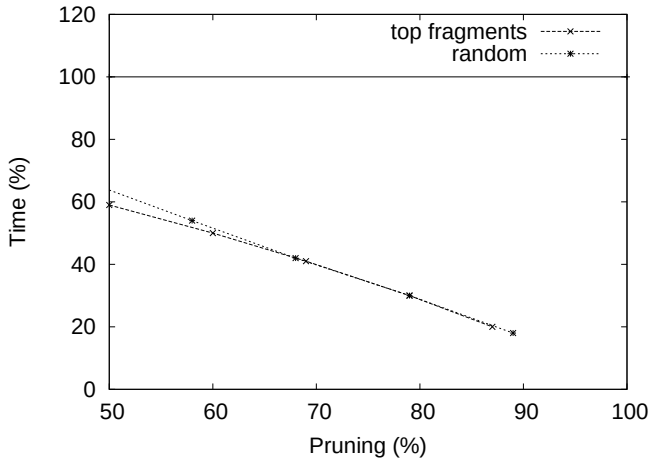


Fig. 15. The percentage of time in function of the total time to create the non-pruned indices.

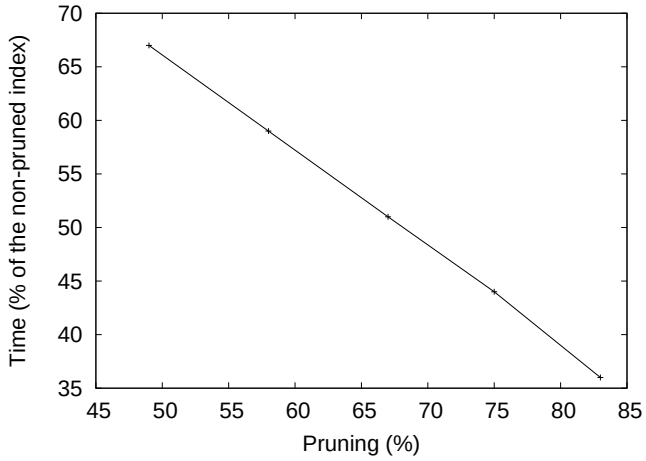


Fig. 16. Time for processing a total of 100,000 queries extracted from the TodoBR query log, at different levels of pruning, using the LBPM method. Time is given as a percentage of the time obtained for processing queries with the non-pruned indices.

processing times depends on the system implementation, hardware available and on the amount of extra information processed by the search system. Therefore, results presented on Figure 16 serve to merely illustrate that pruning may have a significative impact on query processing times.

7. CONCLUSIONS AND FUTURE WORK

We have proposed and experimented new static pruning methods specifically designed for web search engines. Experiments indicate that word locality is essential for designing pruning methods for the web, although not as important for more traditional information retrieval collections, such as LAT TREC. Even an extremely simple locality based pruning method is superior to methods that do not rely on locality information. For instance, the simple top fragments method was superior to Carmel's method when applied to a web search engine.

Based on the observed characteristics of the web collections, such as the necessity of preserving relationships among terms and the presence of atomic documents, we proposed a family of locality based pruning methods that achieve competitive results when compared to the best static pruning method found in the literature.

For the *random* and *top fragments* methods, this performance is obtained with the additional advantage of reducing the index construction times and making index updates easy. When compared to LBPM, a locality based method that combines locality with the similarity heuristic adopted by Carmel's method, these two naive methods methods have a slightly worse result in ranking quality, but this loss is compensated by their simplicity and flexibility in updates.

The use of static pruning methods opens the possibility of improvements in the ranking through the addition of more sophisticated ranking strategies. This is usually possible since the amount of data managed at query processing is smaller after the pruning, saving time for use in the ranking strategy. The decision of adopting a pruning method in a search engine depends on the system designer and is a tradeoff between performance for building or updating the index and quality of results at query processing time.

As future work, we plan to study the impact of the fragment selection process on the final results. In spite of the good results obtained here, previous work on how the fragment selection affects the retrieval quality have pointed that the fragment type may play an important role in the retrieval effectiveness of information retrieval systems [Salton et al. 1993; Callan 1994; Liu and Croft 2002]. Therefore, we plan to study how the fragment type selection affects the effectiveness of our locality based pruning method, trying to determine the fragment type that maximizes the final ranking quality achieved after pruning.

Finally, another important issue is to determine the amount of pruning rate based on the collection statistics. The experimental results presented here indicate that different pruning levels should be adopted for each query type and each collection.

In case of the query types, it is possible to create a cascade index scheme, where each inverted list is divided in blocks of different pruning levels. The number of blocks necessary for processing a query may be selected according to the query type. For instance, a disjunctive query may require only a block obtained at a 90% compression rate, while a disjunctive query may use these entries plus another block obtained at a 50% compression rate. Using this strategy, each query type would take advantage of the its best tradeoff between pruning rate and ranking quality. This idea can be extended to a per query analysis of the most appropriate pruning level, thus obtaining a combination of static and dynamic pruning. We also intend to study this possibility in future work.

Acknowledgments

This work was partially supported by projects SIRIAA (CNPq/CT-Amazônia 55.31 26/2005-9), GERINDO (CNPq/CT-INFO 552.087/02-5), FCT (POSC/EIA/58194/2004) and ADAPTINF (GRICES/CNPq bilateral cooperation), CNPq individual grant 303576/2004-9 (Edleno S. de Moura), CNPq individual grant 303032/2004-9 (Altigran S. Silva) . Celia Francisca was supported by CAPES, Brazil, and Mario A. Nascimento was partially supported by NSERC, Canada.

REFERENCES

Aggarwal, C. C., Al-Garawi, F., and Yu, P. S. 2001. On the design of a learning crawler for topical resource discovery. *ACM Transactions on Information Systems(TOIS) 19,* 3 (July), 286–309.

Anderson, T. W. and Finn, J. D. 1997. *The New Statistical Analysis of Data*, 1st ed. Springer-Verlag.

Baeza-Yates, R. and Ribeiro-Neto, B. 1999. *Modern Information Retrieval*. Addison-Wesley.

Bahle, D., Williams, H. E., and Zobel, J. 2002. Efficient phrase querying with and auxiliary index. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 215–221.

Bell, T. C., Cleary, J. G., and Witten, I. H. 1990. *Text compression*. Prentice Hall.

Brandow, R., Mitze, K., and Raul, L. F. 1995. Automatic condensation of electonic publications by sentence selection. *Information Processing and Management 31,* 5, 675–685.

Broder, A. 2002. A taxonomy of web search. *SIGIR Forum 36,* 2, 3–10.

Calado, P. P., de Moura, E. S., Ribeiro-Neto, B., Silva, I., and Ziviani, N. 2003. Local versus global link information in the web. *ACM Transactions on Information Systems (TOIS) 21,* 1, 42–63.

Callan, J. P. 1994. Passage-level evidence in document retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. 302–310.

Carmel, D., Cohen, D., Fagin, R., Farchi, E., Herscovici, M., Maarek, Y. S., and Soffer, A. 2001. Static index pruning for information retrieval systems. In *Proceedings of the 24th*

Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New Orleans, Louisiana, USA, 43–50.

Cho, J. and Garcia-Molina, H. 2003. Effective page refresh policies for web crawlers. *ACM Transactions on Database Systems (TODS) 28,* 4 (December), 390–426.

de Moura, E. S., dos Santos, C. F., Fernandes, D. R., Silva, A. S., Calado, P., and Nascimento, M. A. 2005. Improving web search efficiency via a locality based static pruning method. In *Proceedings of the 14th International World Wide Web Conference*. Chiba, Japan, 0–1.

de Moura, E. S., Navarro, G., Ziviani, N., and Baeza-Yates, R. 2000. Fast and flexible word searching on compressed text. *ACM Transactions on Information Systems(ACM TOIS) 18,* 2 (April), 113–139.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. 1990. Indexing by latent semantic analysis. *Journal of American Society for Information Sciency 41,* 6, 391–407.

Edmundson, H. P. 1968. New methods in automatic extraction. *Journal of the ACM 16,* 2, 264–285.

Fagin, R., Kumar, R., and Sivakumar, D. 2003. Comparing top k lists. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. 28–36.

Hawking, D., Craswell, N., and Thistlewaite, P. B. 1998. Overview of TREC-7 very large collection track. In *The Seventh Text REtrieval Conference (TREC-7)*. Gaithersburg, Maryland, USA, 91–104.

Hawking, D., Craswell, N., Thistlewaite, P. B., and Harman, D. 1999. Results and challenges in web search evaluation. *Computer Networks 31,* 11–16 (May), 1321–1330. Also in Proceedings of the 8th International World Wide Web Conference.

Hawking, D., Voorhees, E., Bailey, P., and Craswell, N. 1999. Overview of trec-8 web track. In *Proc. of TREC-8*. Gaithersburg MD, 131–150.

Hovy, E. H. and Lin, C.-Y. 1998. *Automated Text Summarization in SUMMARIST*. Mit press, Chapter Intelligent Scalable Summarization Text Summarization, 81–94.

Kleinberg, J. M. 1998. Authoritative sources in a hyperlinked environment. In *Proceedings*

of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms. San Francisco, California, USA, 668–677.

Liu, X. and Croft, W. B. 2002. Passage retrieval based on language models. In *Proceedings of the eleventh international conference on Information and knowledge management (CIKM)*. 375–382.

Mallett, D., Elding, J., and Nascimento, M. A. 2004. Information-content based sentence extraction for text summarization. In *ITCC '04: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2*. IEEE Computer Society, 214.

Melnik, S., Raghavan, S., Yang, B., and Garcia-Molina, H. 2001. Building a distributed full-text index for the web. *ACM Transactions on Information Systems(TOIS) 19,* 3 (July), 217–241.

Navarro, G., de Moura, E. S., Neubert, M., Ziviani, N., and Baeza-Yates, R. 2000. Fast and flexible word searching on compressed text. *Information Retrieval Journal 3,* 1, 49–77.

Nomoto, T. and Matsumoto, Y. 2001. A new approach to unsupervised text summarization. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM Press, 26–34.

Persin, M., Zobel, J., and Sacks-Davis, R. 1996. Filtered document retrieval with frequency-sorted indexes. *Journal of the American Society of Information Science 47,* 10 (Oct.), 749–764.

Rose, D. E. and Levinson, D. 2004. Understanding user goals in web search. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*. ACM Press, New York, NY, USA, 13–19.

Rosenbloom, A. 2004. The blogosphere: Introduction. *Communications of the ACM 47,* 12, 30–33.

Sakai, T. and Sparck-Jones, K. 2001. Generic Summaries for Indexing in IR. In *SIGIR Conference on Research and Development in Information Retrieval*. 190–198.

Salton, G., Allan, J., and Buckley, C. 1993. Approaches to passage retrieval in full text

information systems. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 49–58.

Salton, G. and McGill, M. J. 1983. *Introduction to Modern Information Retrieval*, 1st ed. McGraw-Hill.

Sanderson, M. and Zobel, J. 2005. Information retrieval system evaluation: effort, sensitivity, and reliability. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 162–169.

Silverstein, C., Henzinger, M., Marais, H., and Moricz, M. 1998. Analysis of a very large altavista query log. Tech. Rep. 14, Systems Research Center Laboratory. October.

Silverstein, C., Marais, H., Henzinger, M., and Moricz, M. 1999. Analysis of a very large web search engine query log. *SIGIR Forum 33,* 1, 6–12.

Witten, I. H., Moffat, A., and Bell, T. C. 1999. *Managing Gigabytes: Compressing and Indexing Documents and Images*, 2nd ed. Morgan Kaufmann Publishers.

...