

# Revisiting Globally Sorted Indexes for Efficient Document Retrieval

Fan Zhang<sup>2\*</sup>, Shuming Shi<sup>1</sup>, Hao Yan<sup>3\*</sup>, Ji-Rong Wen<sup>1</sup>

<sup>1</sup> Microsoft Research Asia

<sup>2</sup> Nankai University, China

<sup>3</sup> Polytechnic Institute of New York University

{shumings, jrwen}@microsoft.com, v-fazhan@microsoft.com, hyan@cis.poly.edu

## ABSTRACT

There has been a large amount of research on efficient document retrieval in both IR and web search areas. One important technique to improve retrieval efficiency is early termination, which speeds up query processing by avoiding scanning the entire inverted lists. Most early termination techniques first build new inverted indexes by sorting the inverted lists in the order of either the term-dependent information, e.g., term frequencies or term IR scores, or the term-independent information, e.g., static rank of the document; and then apply appropriate retrieval strategies on the resulting indexes. Although the methods based only on the static rank have been shown to be ineffective for the early termination, there are still many advantages of using the methods based on term-independent information. In this paper, we propose new techniques to organize inverted indexes based on the term-independent information beyond static rank and study the new retrieval strategies on the resulting indexes. We perform a detailed experimental evaluation on our new techniques and compare them with the existing approaches. Our results on the TREC GOV and GOV2 data sets show that our techniques can improve query efficiency significantly.

## Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: Search process;

H.3.4 [Systems and Software]: Performance evaluation (efficiency and effectiveness)

## General Terms

Algorithms, Performance, Experimentation

## Keywords

Top- $k$ , Dynamic index pruning, Globally-sorted index

## 1. INTRODUCTION

Current commercial search engines have to answer thousands of queries per second on billions of documents. Such performance challenges have led to a lot of research on how to improve the efficiency of query processing or document retrieval, using techniques such as massive parallelism, caching, inverted index

compression and early termination. We focus on one important class of optimizations, early termination (or called dynamic query pruning), which has been widely used in large search engines [31]. The common goal of various early termination techniques [1, 2, 3, 7, 8, 12, 15, 16, 19, 23, 24, 27, 30, 31, 32, 34, 35] is to speed up document retrieval by avoiding processing all indexes that are relevant to the given query.

To better understand early termination techniques for search engines, we first look at the basic index structure [4, 36, 31]: inverted index, which consists of many *inverted lists*. Each inverted list is a sequence of postings, each of which contains a document ID (docID), plus additional information such as the term frequency in the document, the exact positions of the occurrences, and their context (e.g., in title, in URLs, etc). Typically all postings in each inverted list are sorted by their docIDs. To process a query, search engines need to traverse the entire inverted lists for all relevant terms and calculate relevance scores for all documents in the involved lists, and finally return the top- $k$  (e.g.,  $k = 10$ ) documents having the highest scores. However, since the typical sizes of inverted lists are very long especially for common terms in large collections, such exhaustive evaluation requires significant computing resources and may slow down the query response greatly.

To overcome the above problem, many early termination techniques have been proposed [1, 2, 3, 7, 8, 12, 15, 16, 19, 23, 24, 27, 30, 31, 32, 34, 35]. The basic idea underlying is to first sort the postings in the inverted lists in such a way that the most promising documents are skewed towards the beginning of the inverted lists (and thus postings are often not ordered by document IDs), and then adopt appropriate evaluation strategies on the resulting lists such that the documents with the highest scores will be identified without scanning the entire lists. The way to sort an inverted list plays a vital role in these early termination techniques since it affects the location where the following early termination condition is satisfied: the minimum score in the current top- $k$  document list is greater than the maximal possible score of all documents to be processed.

In terms of their ways to organize the inverted lists, most existing early termination techniques can be roughly divided into the following categories: many approaches sort their inverted lists by term-dependent information, e.g., term frequency in the document [24], or the IR scores or the impact [3], while other approaches do so by combining them with other term-independent information, e.g., static rank [6, 19], or simply sort them by document IDs [31]. We call the term-independent information global scores (GS) since such scores can be determined globally without considering the term information for each document, and correspondingly call the term-dependent information local scores (LS). For simplicity

\* This work was done when Fan Zhang and Hao Yan were interns at Microsoft Research Asia.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WSDM'10, February 4-6, 2010, New York City, New York, USA.

Copyright 2010 ACM 978-1-60558-889-6/10/02...\$10.00.

of the later discussion, we call the above two kinds of approaches LS methods and GS-LS methods respectively. Besides these two kinds of approaches, it is also interesting to see if sorting postings only by their GS scores, e.g., static rank, can also achieve effective early termination (we call such methods GS methods).

It is interesting to study the GS methods due to the following reasons: first, current search engines often integrate a variety of other features (including static rank) beyond term IR scores [6, 19, 34, 35] into an overall scoring function using machine learning techniques; second, the resulting indexes of the GS methods can be easily transformed into the typical indexes, where postings are sorted by docIDs, as long as docIDs are reassigned in the order of their documents' global scores. As a result, most index processing techniques that are suitable for the typical indexes, e.g., index maintenance and compression, and some efficient query evaluation techniques, e.g., document-at-a-time (DAAT) [7, 17, 36], can be directly applied to the transformed indexes; more interestingly, [6, 19, 34, 35] has shown that the global information may be integrated together with the local information, where they are orthogonal to each other, to achieve the overall better query processing performance than the LS methods. Thus, it is possible that we may further improve the query processing performance on such kind of indexes by integrating a better GS method. Although it has been shown through experiments [19, 34, 35] that the GS methods based solely on static rank do not perform well in practice, the underlying reasons of the ineffectiveness in such cases and other GS methods considering more global information beyond static rank are still to be explored.

Therefore, in this paper, we focus on new techniques for organizing inverted indexes based solely on the global information besides static rank, to achieve effective early termination and thus efficient document retrieval for search engines. The main challenges are to choose the appropriate global information and then to integrate them together to achieve effective early termination. In particular, we first perform theoretical analysis about the relationship between early termination and the distribution of various scores. We then propose new approaches to organize each inverted list by the maximum of the static rank of the document and the upper bound of the term IR scores (UBIR) for all terms contained. In this way, the documents with either higher static rank values or a larger bound of term IR scores are located around the beginning of the inverted lists. The major advantage of our method is that the UBIR may greatly reduce the estimated values of term IR scores for those documents to be processed during query processing and thus the early termination condition can be satisfied much earlier than the method without using it. Our experimental results show significant improvements in terms of effective early termination.

The remainder of this paper is organized as follows. The next section offers some technical background and discusses related work. Section 3 describes our contributions in more detail. Section 4 provides the theoretical analysis of the relationship between the early termination and the distributions of scores. Section 5 proposes our main algorithms, while Section 6 evaluates query efficiency of our algorithm and compares our approaches with other methods. Finally, Section 7 describes concluding remarks and future work.

## 2. BACKGROUND AND RELATED WORK

In this section, we first give some background on ranking functions, then discuss in Subsection 2.2 the previous efficient document retrieval techniques and in Subsection 2.3 those GS techniques. Finally in Subsection 2.4 we introduce the data sets used in our experiments. Please refer to [4, 31, 36] for the basic background knowledge on indexing and query processing.

### 2.1 Ranking Functions

IR systems often use term-based ranking functions [31] to calculate document scores for a given query, while current search engines usually integrate into the ranking functions other link-based factors [5, 6, 19, 31], e.g., Pagerank [6], or even with many other factors achieved from machine learning techniques. Therefore, the overall score of a document for search engines is generally a combination of the term-based score and the linked-based score. For simplicity, we call the term-based scores IR scores and linked-based scores static rank (SR) scores for the rest of this paper (although linked-based scores may often also consider other linked-based factors other than SR scores).

There is not much research on using the integrated ranking function, which considers both SR and IR scores, for efficient document retrieval in search engines, although there are a few of them [19, 34, 35]. Their common way to combine both kinds of scores is to use the weighted sum of them as follows:

$$S(d, q) = \alpha \cdot SR(d) + \beta \cdot IR(d, q) \quad (2.1)$$

where  $S(d, q)$  is the overall score of the document  $d$  with respect to the query  $q$ ,  $SR(d)$  is the SR score of the  $d$ ,  $IR(d, q)$  is the IR score of the document  $d$  with regard to the query  $q$ , and  $\alpha$  and  $\beta$  are two non-negative parameters ( $\alpha + \beta = 1$ ). Normally both SR and IR scores are normalized into the range  $[0, 1]$ . One popular way to calculate the IR score is the BM25 formula used in [26], which has been widely used in IR and search engine areas and thus adopted in the later sections of this paper, as follows:

$$IR(d, q) = \sum_{t \in q} \omega_t \cdot \frac{(1+k_1)tf}{k_1((1-b)+b\frac{dl}{avdl})+tf} \quad (2.2)$$

where  $tf$  is the term frequency (i.e., the number of occurrences) in the document,  $dl$  is the document length,  $avdl$  is the average length of all documents in the collection,  $k_1$  ( $\geq 0$ ) and  $b$  ( $\in [0, 1]$ ) are two parameters, and  $w_t$  is the inverse document frequency weight of the term  $t$  and can be computed as follows:

$$\omega_t = \log \frac{N-n_t+0.5}{n_t+0.5} \quad (2.3)$$

where  $N$  is the total number of all documents in the collection while  $n_t$  is the number of documents containing the term  $t$ . From the above formula (2.2) and (2.3), we can see that the rare terms in the collection will have higher  $w_t$  values than the common terms, while the frequently-used terms in a document tend to have higher scores for the second factor in the formula (2.2) than those who appear in the document only a few times. Thus, a document will be assigned a high IR score if a rare term appears in it frequently.

The ranking functions of practical search engines also take into consideration the context of terms' occurrences, e.g., in the title, or in the URL, for efficient query processing [6, 34]. In practice, such fields should be treated differently due to their distinct contributions to the overall score. In our algorithms to be

proposed in later sections, we will distinguish the following four different contexts (we call them fields) of a web document: title, URL, anchor (text) and body fields [6, 34], where the anchor text refers to the visible, clickable text (in other pages) in a hyperlink, while the body field refers to the rest parts of a web page rather than the other three fields. The new ranking function considering such information will be discussed in more details in later sections.

We note that the document score may also depend on the distance between the query terms in the document, called term proximity [25, 27, 34, 35] and terms occurring close to each other may result in a higher score. The ranking function needs be changed if term proximity is taken into account and so does the corresponding retrieval strategy. However, in the preliminary version of this paper, we do not consider such information in our ranking function and will discuss it in the final version.

## 2.2 Efficient Document Retrieval

The goal of the document retrieval is to efficiently find the top  $k$  documents with the highest scores for a given query. One important way to improve retrieval efficiency is to use early termination techniques [1, 2, 3, 7, 8, 12, 15, 16, 19, 23, 24, 27, 30, 31, 32, 34, 35] which in the ideal cases terminate the query processing immediately once the top  $k$  results have been found, skipping the rest part of the lists. It can be described more formally as follows: if we define  $S_k$  as the  $k$ th largest score of the current top- $k$  result list and  $S_T$  as the maximal possible score of all the unprocessed (or unseen) documents in the rest of the inverted lists. The early termination can be executed once the following condition is satisfied:

$$S_k \geq S_T \quad (2.4)$$

However, in practice search engines often require that the  $k$  documents in the result list (achieved by the early termination techniques) must be returned in the same order as those results that are achieved when the entire lists are processed without the early termination. To achieve this goal, the following condition has to be satisfied before the early termination is executed:

$$S_i^{min} \geq S_{i+1}^{max} \quad (2.5)$$

where  $S_i^{min}$  is the minimum possible value of the  $i$ th document and  $S_{i+1}^{max}$  is the maximal possible value of the  $(i+1)$ th document in the top- $k$  result list respectively, both of which are estimated right before the execution of the early termination.

As mentioned in Section 1, most early termination techniques in IR and web search areas are often based on certain index structures, which are achieved by sorting postings according to the term frequency [24], IR score or term impact [3], or the combination of IR score and Pagerank [6, 19], or simply the docIDs. They then apply on the resulting indexes appropriate document evaluation strategies to achieve efficient document retrieval. The ways to achieve various index structures have been discussed in Section 1. Therefore we only discuss the document evaluation algorithms in this subsection.

Many evaluation strategies [3, 7, 8, 16, 17, 19, 21, 30, 33, 34, 35, 36] have been proposed in IR and web search areas and they can be roughly divided into the following three categories: document-at-a-time (DAAT) [7, 17, 19, 30, 33, 34, 35, 36], term-at-a-time (TAAT) [8, 16, 21, 30, 36] and score-at-a-time (SAAT) [3]. DAAT evaluates a document considering the contributions of all query terms before it deals with the next document; TAAT

evaluates all documents in the inverted list of a term before it does so for the next term; SAAT is only suitable for indexes sorted by impacts [3]. It first splits an inverted list into a few segments such that all documents of a segment have the same quantized term scores (called impacts). It sorts the segments by their impacts in decreasing order and then evaluates all documents in the first segments of all terms before it evaluates the second segments of these terms. While TAAT is widely used in the traditional IR systems and SAAT can achieve good performance in certain cases [36], DAAT has been shown to be able to achieve very good query performance in many cases especially with certain optimizations [7, 17, 19, 30, 33, 34, 35, 36]. DAAT often requires a smaller run-time memory size while the other two methods need more memory to maintain intermediate scores during query processing. In addition, DAAT is more efficient than the other two methods especially when the term position (term proximity) information is considered into the ranking functions [25, 33, 34, 35]. Please refer to [3, 7, 36] for the detailed comparison among those strategies. We also note that, many retrieval algorithms have also been proposed in the database areas, e.g., the Fagin's Algorithm (FA) [13], Threshold Algorithm (TA) and No Random-access Algorithm (NRA) strategies [12]. FA was first introduced while the TA and NRA algorithms are proposed later and attract more research attention. The main difference between TA and NRA is that the former allows random access on the inverted list while the latter one only allows sequential access on it. Please refer to [14] for a survey of these methods.

We now briefly discuss some other related work. First, the retrieval strategies with early termination can either return the exact top- $k$  results of all relevant documents in the collection, or just return the approximate top- $k$  results [12, 19, 33] as long as certain retrieval precision can be reached. Second, the above early termination techniques are essentially dynamic pruning techniques since the early termination condition, according to either the formula (2.4) or (2.5), is judged dynamically at the run time of document retrieval. In contrast, static pruning techniques [9, 10, 22] often predict and discard the unimportant indexes during the process of building the indexes, resulting in much less amount of indexes to be processed during the document retrieval. We note that such methods achieve the retrieval efficiency by sacrificing the search quality since the discarded indexes will never be checked against the query. Another well-studied way to achieve retrieval efficiency is pre-aggregation techniques [11, 18, 20], which first create additional short lists, e.g., the intersections of some lists, and then exploit such lists to speed up the retrieval process. Finally, the early termination strategies are also affected by the caching policies and the techniques for dealing with both of them have been studied in [20, 28, 29]. However, in this paper, we only focus on the dynamic pruning techniques and do not consider pre-aggregation and caching policies.

## 2.3 Document Retrieval with the GS Methods

The GS methods sort postings purely by the global information. The ordering of the indexes achieved by the GS methods often depends on the way the docIDs are assigned, e.g., at random, in the order the documents are crawled or indexed, or based on global measures of page quality (such as Pagerank [2]). [19, 34, 35] have shown that the pure GS methods based only on Pagerank or static rank cannot lead to the effective early termination (although the methods using the combination of GS and LS information can achieve impressive retrieval efficiency).

However, we will see from the later sections that as long as other appropriate global information is properly integrated with static rank to sort inverted lists, we can still achieve efficient documents retrieval. To make our algorithms (to be discussed in later sections) more understandable, we describe in this subsection the basic retrieval process (i.e., the retrieval strategies) of the GS methods as follows.

Given a query, the query processor first locates the next document to be evaluated in a DAAT manner, and then computes its overall score, containing both IR and SR scores. If the score is larger than that of the  $k$ th document in the current top- $k$  result list, the document will be added into the list, otherwise it will be discarded. The early termination condition is checked periodically and once it is satisfied, all top- $k$  results will be returned by the processor. From the retrieval process, we can see that we do not need to keep the auxiliary data structures to store the intermediate scores that are required by other documents evaluation algorithms such as TAAT and SAAT.

### 3. CONTRIBUTIONS OF THIS PAPER

In this paper, we study and evaluate efficient document retrieval techniques under the index structures achieved by sorting inverted lists according to only the term-independent information. Our main contributions are as follows:

- (1) We perform a detailed theoretical analysis for the relationship between early termination and the distribution of SR and IR scores. We find that as long as the IR scores are uniformly distributed, effective early termination can always be achieved. However, the real distribution of IR scores often does not conform to the uniform distribution and the skewed distribution is one of the important reasons for the ineffectiveness of early termination for the GS methods.
- (2) We propose new algorithms to first organize inverted lists by the global information beyond static rank and then perform query processing in a DAAT manner on the resulting lists. In particular, we use as the global information the maximal value of the document's static rank and the upper bound of its IR scores (UBIR) for all terms contained in it. In this way, the estimated IR scores for all the documents to be processed can be greatly reduced such that the early termination conditions can be satisfied quickly. We also present several additional approaches based on other global information.
- (3) We evaluate our algorithms on the TREC GOV and GOV2 data sets, and compare with the existing GS methods. Our experimental results show that our algorithms can improve query efficiency significantly.

### 4. PROBLEM ANALYSIS

In this section, we first perform a theoretical analysis on the relationship between the early termination and the distribution of SR and IR scores. We then verify it through our simulation on the artificial data and the experiments on the real data sets. Our main observations are as follows: On one hand, we can always achieve early termination on the artificial indexes on the GS indexes where inverted lists are sorted by the SR scores, as long as the IR scores are uniformly distributed; while on the other hand, we cannot achieve early termination on the similar GS indexes of the real large scale data sets, where the IR scores are not uniformly distributed.

From above, we can see that the IR scores (in the ranking function (2.1) and (2.2)) play an important role on the efficient document retrieval especially when the indexes are the GS indexes. The main challenge is that the precise estimation of the maximal possible IR score for the documents to be evaluated is hard to achieve for the GS indexes that are based only on static rank. This motivates us to integrate the IR score into the global information to sort inverted lists (It will be desirable that the integrated score is still term-independent due to the advantages of the GS indexes discussed in Section 1).

To perform our theoretical analysis, we make the following assumptions: first, both the SR (static rank) score and the IR score of a document conform to the uniform distributions in  $[0,1]$ ; second, each inverted list is sorted by SR scores; third, the weight of the SR score, i.e. the  $\alpha$  in the formula (2.1), always satisfy  $\alpha \leq 0.5$ , which has been shown to be a reasonable range; finally, we assume the evaluation is performed in a DAAT manner.

Given a query  $q$ , we want to compute the expected earliest location on the intersected list where the early termination condition, i.e., the formula (2.3), can be satisfied. Formally, suppose  $n$  is the number of documents contained in the intersected list  $L_{inter}$  of all inverted lists relevant to  $q$ , while  $M$  is a random variable representing the number of documents that have been processed in the  $L_{inter}$  at the time point when early termination condition is satisfied. Our above goal can then be expressed as the expected value of  $M$  with regard to a known  $n$  as follows:

$$E(M) = \sum_{m=1}^n m \cdot P(M = m) \quad (4.1)$$

where  $P(M=m)$  is the probability that the query processing is terminated at the  $m$ th document on  $L_{inter}$  ( $1 \leq m \leq n$ ). Through the detailed deduction shown in the appendix, the value of  $E(M)$  can be described as follows under our above assumptions:

$$E(M) = 1 + n \cdot \int_0^1 \left(1 - \frac{\alpha \cdot x^2}{2\beta}\right)^{n-1} dx \quad (4.2)$$

where  $\alpha$  and  $\beta$  are the parameters also used in the formula (2.1). The detailed description of the above formula is shown in the appendix. However, another interesting intermediate result is shown as follows:

$$P(d.\text{score} \geq S_T) = \alpha(1 - r)/2\beta \quad (4.3)$$

where  $d$  is the last document that has been processed at the current time point,  $S_T$  is the maximal possible overall score of all documents on the  $L_{inter}$  that are after  $d$  (i.e., all document to be processed). The  $P(d.\text{score} \geq S_T)$  is the probability that early termination will happen at the current time point. The  $r$  is the static rank value of the document  $d$ . We note that  $r$  is in the decreasing order along the list since the indexes are the GS indexes. From the formula (4.3), we can see that the probability of early termination is increasing when more documents have been processed. Thus we can expect to always achieve early termination under assumption that the IR score is uniformly distributed.

In order to verify the above conclusion, we perform several simulations and experiments on the artificial data sets and real data sets, where the inverted lists are always sorted by the SR scores while the SR and IR scores may conform to different kinds of distributions. We show the results as the average percentage of accessed documents when the query processing stops and use as the input the number of relevant documents  $n$ . The basic retrieval algorithm discussed in Subsection 2.3 is used in the simulations

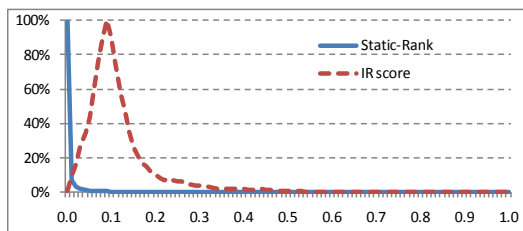
and experiments. Please refer to Subsection 6.1 for the experimental setup.

In Table 4-1, we compare the average percentage of accessed documents achieved for the theoretic analysis and the simulations, where both the artificial SR and IR scores are uniformly distributed. From Table 4-1, we can see that the simulation results match our theoretical analysis perfectly.

**Table 4-1. Average percentage of accessed documents when the query processing algorithm stops, a comparison between theoretic analysis and simulation ( $k=1$ )**

$n$	$E(m)/n$ (theoretic analysis)			$Avg(m)/n$ (simulation)		
	$\alpha=0.1$	$\alpha=0.3$	$\alpha=0.5$	$\alpha=0.1$	$\alpha=0.3$	$\alpha=0.5$
10	94.7%	69.3%	50.0%	89.0%	68.0%	50.0%
100	38.6%	20.2%	13.5%	38.7%	20.2%	13.5%
1000	12.0%	6.15%	4.06%	11.9%	6.15%	4.06%
10000	3.77%	1.92%	1.26%	3.75%	1.90%	1.27%
100000	1.19%	0.61%	0.40%	1.20%	0.60%	0.40%

However, the SR and IR scores of the real web data sets are often not uniformly distributed. For example, we show in Figure 4-1 the distribution of SR and IR scores of all documents in the TREC GOV data set that are relevant to the 2003np and 2004mixed query sets. From Figure 4-1, we can see that neither SR nor IR scores is uniformly distributed. The similar observation can be achieved on the GOV2 dataset and we do not show it here due to the space limitation.



**Figure 4-1. The distribution of static-rank and IR scores of documents (Dataset: GOV; query-set: 2003np+2004mixed). The x-axis is the score values for both SR and IR scores, while the y-axis is the number of documents for every possible score and is shown as the percentage of their maximal value**

Thus, it is interesting to see whether we can still achieve effective early termination when the SR and IR scores are not uniformly distributed. In particular, we consider four different combinations of the distributions for the SR and IR scores (shown in Table 4-2), and we compare the average percentage of accessed documents achieved for each of the four cases in Table 4-3 (where we only return the top-1 document with the highest score, i.e.,  $k=1$ ) and in Table 4-4 (where we return the top-10 documents). From both tables, we can see that effective early termination can hardly be achieved for the real data sets where the IR scores are not uniformly, although we can do so when the IR scores are uniformly distributed.

**Table 4-2. Four settings about the static-rank distribution and IR score distribution of documents ( $\alpha=0.3$ )**

IR score distribution \ Static-rank distribution	Uniform	Real
Uniform	I	III
Real	II	IV

**Table 4-3. Simulation results of the average percentage of processed documents when our query processing algorithm stops ( $\alpha=0.3, k=1$ )**

Case \ $n$	I	II	III	IV
100	20.2%	74.4%	99.6%	100%
1000	6.15%	7.38%	95.6%	100%
10000	1.90%	0.23%	78.3%	99.9%
100000	0.60%	0.026%	61.4%	99.5%

**Table 4-4. Simulation results of the average percentage of processed documents when our query processing algorithm stops ( $\alpha=0.3, k=10$ )**

Case \ $n$	I	II	III	IV
100	68.8%	100%	100%	100%
1000	21.4%	99.8%	100%	100%
10000	6.74%	3.49%	100%	100%
100000	2.12%	0.216%	81.8%	100%

## 5. OUR ALGORITHMS

In this section, we describe our algorithms for efficient query processing. Our algorithms contain two major phrases: in the first phrase, we build the new indexes by sorting the inverted lists according to only the global (term-independent) information; while in the second phrase, we use appropriate retrieval strategies in a DAAT manner to process the resulting indexes. The main idea of our method is to use the global information available in the new indexes to reduce the estimated values of the maximal possible overall score for all documents to be processed, such that the early termination conditions can be quickly satisfied during query processing. Our algorithms are motivated by the GS method discussed in [19], where all documents in each segment of an inverted list are sorted by their SR (static rank) scores. For the simplicity of the discussion later, we denote the typical indexes, where inverted lists are sorted by docIDs, as the TPI indexes, while we denote the GS indexes, where inverted lists are sorted only by the SR (static rank) values, as the TSR indexes. In the rest of this section, we will first discuss the ranking function used in our algorithms and then propose our methods to build the new indexes and the corresponding retrieval strategies.

### 5.1 The Ranking Function

As mentioned in Section 2, we integrate the context information (or the structured information) of term occurrences into our ranking function considering the different contributions from the following four fields of a web page: title (T), URL (U), anchor text (A) and body fields (B). In particular, we use the weighted sum to combine their separate IR scores into a unified IR score, i.e., the IR score in the ranking function (2.1). In other words, the ranking function (2.1) is changed in this case as follows:

$$S(d, q) = \alpha \cdot SR(d) + \beta \cdot \sum_{i \in \{T, U, A, B\}} w_f \times IR(d, q, i) \quad (5.1)$$

where  $w_i$  is the weight for the  $i$ th field and  $IR(d, q, i)$  is the term IR score for the field. The  $IR(d, q, i)$  can be calculated using the formula (2.2) except that the values of all variables in the formula are determined with respect to the field instead of to the entire document. For example, the  $tf$  value will be the term frequency of the term in the  $i$ th field instead of in the document.

## 5.2 Building New Indexes

We build the new indexes using three different methods, all of which sort inverted lists by the global information, except that they are based on different kinds of global information

**MSI:** Recall that the UBIR score is the maximal value of the term IR scores for all terms contained in the document. That is, the term IR scores for all terms in the document are not greater than the UBIR score. To build the new indexes, we first calculate the UBIR score for each document in the collection and then combine it with the SR (static rank) score together as a global score (GS), which is then used to sort all inverted lists. In particular, the GS score is  $= \max(SR, \lambda \times UBIR)$ , i.e., the maximal value of the SR score and the weighted UBIR score (we call this method MSI). We note that although the term IR score is term-dependent, both UBIR and MSI values are term-independent.

**SSI:** This method also evaluates the GS score as the combination of the SR and UBIR scores. However, unlike MSI, it calculates the GS score as  $GS = \alpha \times SR + (1 - \alpha) \times UBIR$ . We note that the formula is actually the formula (2.1) and thus the resulting GS score is directly the upper bound of the overall score for all to-be-processed documents. We call this method SSI since the GS score is the weighted sum of SR and UBIR scores.

**MST:** The main difference of this method with the above two methods is that it combines the term frequency, instead of the URIR score, with the SR score to evaluate the GS score. In particular, it uses the maximal value of the term frequency (TF) for all terms in the document as the upper bound of the TF values (UBTF) and we call it MST. It then assigns the GS value as  $GS = \max(SR, \gamma \times UBTF)$ , where  $\gamma$  is a non-negative parameter. We note that, the TF value is actually the sum of the TF values in the three fields of T, U and A (The reason why the body field is not considered here will be discussed soon). Compared to MSI and SSI, the advantage of MST is that it does not need to create new indexes again even if the parameter values of the ranking functions, e.g., the values of  $k_1$  and  $b$  in the formula 2.2, are changed, while the other two methods need to create additional indexes in such cases. However, the disadvantage of this method is that it cannot reduce the upper bound of the overall score for all to-be-processed documents as much as the other two methods (The details will be discussed soon).

## 5.3 Retrieval Strategies

In this subsection, we discuss the retrieval strategies for the indexes achieved by the above three methods. We note that although the resulting index structures are different, a common property of these methods is that a much tighter upper bound of the possible scores for all to-be-processed documents is explicitly or implicitly embedded into their indexes. Therefore, their retrieval strategies are very similar to each other and can be described as a unified strategy shown in Figure 5-1. In contrast, if the upper bound information is not available at run time, the estimated maximal scores of those unseen documents,  $S_T$ , will be often greatly over-estimated such that it is very hard to realize the early termination. In fact, the only difference of the retrieval strategies among the above three methods is they use different ways to estimate  $S_T$ , which corresponds to line with bold fonts, i.e., “Update  $S_T$ ”, in Figure 5-1 and are discussed in more details as follows:

**For MSI:** Given the document  $d$  that was just evaluated during query processing, its GS score,  $GS_d$ , is then available in our

indexes and can be used to predict the value of  $S_T$  as follows: since  $GS_d = \max(SR, \lambda \times UBIR)$ , the maximal possible value of the SR score must not be greater than  $GS_d$ , while the UBIR must not be greater than  $\frac{GS_d}{\lambda}$ . Therefore, in terms of the ranking function (2.1) or (5.1), the upper bound of  $S_T$  can be described as  $S_T \leq \alpha \times GS_d + (1 - \alpha) \times \frac{GS_d}{UBIR}$ . We note that such an upper bound of  $S_T$  may be much less than that achieved on the TPI or TSR indexes. For example, for the TSR indexes, since we do not know any information about the IR score of the to-be-processed documents, we have to estimate it as the maximal possible value that can be achieved from the formula (2.2) particularly when the  $tf$  value is infinite. Obviously this method over-estimates the real  $tf$  values and thus the resulting IR score may be much larger than  $\frac{GS_d}{\lambda}$ . As a result, the estimated overall score  $S_T$  will be also much larger such that the early termination condition  $S_K \geq S_T$  won't be satisfied soon. One thing to note is that the  $S_T$ -values evaluated in MSI always considers the contributions of all four fields of T, U, A and B.

**Algorithm:** Document retrieval strategy for our algorithms

**Input:** Inverted lists  $L_1, \dots, L_{|Q|}$ , for the query  $Q$

**Output:** Top- $k$  documents

```

R = empty; //R: the current top-k result list
SK = 0; //SK: the score of the kth document in R
loop
  d = NextDoc();
  if (d is empty) return R;

  Compute d.score;
  if(|R| < k OR d.score > SK)
    R.insert(d)
    Update SK
  end-if

  //update the maximal possible score for all unseen docs
  Update ST;
  if(|R| ≥ k AND SK ≥ ST)
    return R;
end-loop
return R

```

Figure 5-1 The retrieval strategy of our algorithms

**For SSI:** It is quite easy in this case to evaluate the  $S_T$  since it is directly the value of  $GS_d$ .

**For MST:** It is more complicated to evaluate the  $S_T$  in this case than the above two cases. The main challenge is that we only know the sum of TF values for the three fields of T, A, and U, while we have to estimate the partial UBIR score of those fields (The evaluation of that of the body field will be discussed soon). We solve this problem by creating a table (offline) where for each possible combination of the TF values in these three fields, a partial UBIR score is calculated and stored. We create such a table only for small UBTF values. Since most of TF values in the three fields are very small values, say, from 0 to 4, the table size won't be very large. In contrast, the TF value of the body are often much larger such that the table size will be too large if it is included into the UBTF. During query processing, given a sum of TF values, we first look up the table and then return the corresponding UBIR score if a matched record can be found. To evaluate the partial TR score of the body field, we have to assume

the TF in that field is infinite and then use the formula (2.2) to evaluate it. Once we get the whole UBIR score of all four fields, we can then evaluate  $S_T$  in a similar way as the above two other methods.

## 6. EXPERIMENTS

### 6.1 Experimental Setup

**Datasets and query sets:** For our experiments, we use the following two data sets: TREC GOV and GOV2. The former consists of about 1.25 million web pages crawled from web sites in the gov domain in 2002, while the latter contains around 25.2 million web pages crawled from the gov domain in 2004. These two data sets are widely used in IR research community. For the evaluation on the GOV data set, we use the *2004mixed* query set, which contains 225 queries with 3.43 terms per query on average, and the *2003np* query set, which contains 300. For the evaluation on the GOV2 data set, we also use the *2004mixed* query set.

**Hardware and software environment:** For the GOV dataset, experiments are carried out on a single machine with Dual 3.06 GHz Intel Xeon CPU, 4G RAM and 600 GB local SATA disk. While for corpus GOV2, we distributed its documents to 5 machines (via URL hashing), with each machine having about 5 million documents indexed.

**Evaluation metrics:** We evaluate the search efficiency of our algorithm by the average percentage of processed documents when the early termination happens (abbreviated as *doc# ratio*), and the ratio of average query processing time between an approach and the baseline (abbreviated as *time ratio*). For approximate top- $k$  approaches, we also report the *error rate*, i.e., the average percentage of different results in top- $k$  (compared to the accurate results). We always tune the parameters of the ranking function for the appropriate experiments such that the optimal MAP (mean average precision) can be achieved.

### 6.2 Experimental Results

#### 6.2.1 Overall performance comparison

In Table 6-1, we compare the average percentage of the processed documents (until the early termination happens) on the TREC GOV data set using the TSR index (where documents are sorted only by the SR scores) and the three kinds of indexes we proposed: MSR, SSI, and MST. In each cell of the table, the upper-left and bottom-right numbers are respectively *doc# ratios* and *time ratios*. We have the following observations: First, we can always achieve early termination on all of our three kinds of indexes while we cannot do so for the TSR indexes since our approaches consider the impact of the IR scores on the index organization. Second, both MSI and SSI can outperform MST greatly since the maximal possible score of the future documents can be estimated by the upper bound of IR scores more precisely than by the term frequency. However, one of the disadvantages of the MSI and SSI is that, once the parameters of their ranking functions are changed, the UBIR scores must be recalculated and then the entire indexes have to be reorganized (We note that the MST indexes is not limited to this). Third, as we may expect, the search efficiencies of all of the four index structures become worse when the value of  $k$  (i.e., the number of documents to be returned) is increased. Similar results on the GOV2 corpus are shown in Table 6-2, where *time ratio-2* means the time ratio calculated for queries with more than 3000 relevant results.

**Table 6-1. Performance comparison of various GS-indexes (dataset: GOV;  $\alpha=0.3$  for 2003np and 0.2 for 2004mixed)**

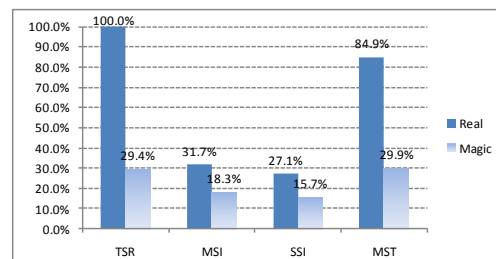
Query set	Index	$k=1$	$k=3$	$k=5$	$k=10$
2003 np	TSR	100% / 100%	100% / 100%	100% / 100%	100% / 100%
	MSI	15.5% / 54.8%	32.8% / 76.0%	39.7% / 81.8%	48.1% / 87.5%
	SSI	5.9% / 44.5%	19.1% / 65.7%	24.3% / 71.7%	32.0% / 81.1%
	MST	21.5% / 65.1%	47.3% / 89.7%	56.4% / 95.4%	63.5% / 95.8%
2004 mixed	TSR	100% / 100%	100% / 100%	100% / 100%	100% / 100%
	MSI	16.9% / 63.5%	26.2% / 80.8%	31.7% / 86.0%	41.8% / 90.7%
	SSI	11.8% / 60.1%	21.4% / 78.1%	27.1% / 83.5%	37.3% / 88.2%
	MST	49.8% / 94.9%	77.8% / 99.5%	84.9% / 99.8%	91.0% / 99.4%

**Table 6-2. Performance comparison of various GS-indexes (metric: ratio; dataset: GOV2; query set: 2004mixed;  $\alpha=0.2$ )**

Index	$k=1$			$k=5$		
	Doc# Ratio	Time Ratio	Time Ratio-2	Doc# Ratio	Time Ratio	Time Ratio-2
TSR	100%	100%	100%	100%	100%	100%
MSI	12.2%	63.3%	37.0%	20.7%	82.0%	64.9%
SSI	10.7%	62.9%	33.4%	18.8%	82.1%	60.7%
MST	70.9%	97.5%	96.8%	88.9%	99.7%	99.2%

#### 6.2.2 The potential of various kinds of indexes

We compare in Figure 6-1 the least number of documents to be evaluated (i.e., we assume that we magically know where the top- $k$  documents are in the inverted lists) on the GOV data set using our three kinds of indexes. Since the locations of all top- $k$  documents are magically known, the query processing can be immediately terminated once the last occurrence of all top- $k$  documents has been scanned and evaluated. Therefore, the results show the potential that the best early termination techniques can achieve under each of the index organizations. We also show in Figure 6-1 the corresponding results achieved from the real early termination (where we do not know the locations of the top- $k$  documents).



**Figure 6-1. The real and magic performance of various GS-indexes (metric: doc# ratio, dataset: GOV; query set: 2004mixed;  $\alpha=0.2$ ;  $k=5$ )**

From Figure 6-1, we can see that early termination can always be achieved (for all of the four kinds of indexes) as long as an Oracle tells us the locations of all occurrences of the top- $k$  documents on the inverted lists, since the top- $k$  documents have been organized in such a way that they are skewed towards the beginning of the inverted lists and thus can be evaluated much earlier than most of the other documents on the lists. However, from Figure 6-1, we

observe that such an important fact cannot be detected on the TSR indexes and as a result we still evaluate the entire inverted lists. In contrast, our MSI and SSI detect this very early and therefore can achieve effective early termination (we note that the query performance of them is even comparable to that of their counterparts with the magic information). As has been discussed, the main reason is that MSI and SSI can reduce the estimated values of the maximal possible scores for the future documents much more than TSR and MST. Therefore, they can predict more precisely the point at which early termination happens.

### 6.2.3 Experimental results for different parameters

We show in Figure 6-2 the impact on the query efficiency of the static-rank weight  $\alpha$  especially for the MSR and SSI indexes. We vary  $\alpha$  in  $[0.1, 0.5]$  in the Figure. We can see that better query performance can be achieved with a larger  $\alpha$  value, which indicates that the SR scores can still affect the early termination greatly. However, we note that this is not true when  $\alpha > 0.5$  (we do not show the results in those cases for the same reasons in [34, 35]). For example, when  $\alpha = 1$ , i.e., the overall document score is equal to the SR score, we cannot achieve early termination at all (e.g., the results on the TSR indexes shown in Figure 6-1).

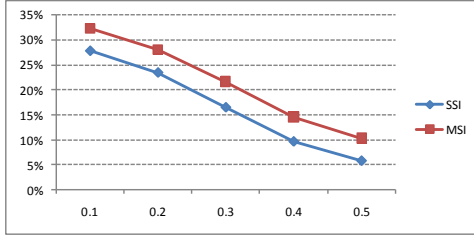


Figure 6-2. Performance of SSI and MSI, on various static-rank values (dataset: GOV; query set: 2004mixed;  $k=3$ )

### 6.2.4 Approximation query processing

As we have discovered in Figure 6-1, the real early termination is always later than the magic early termination for all of the four indexes. This is because in the former cases we always overestimate the maximal possible score of the future documents (while the magic methods never do so), i.e., the value of  $S_T$ , such that the early termination condition  $S_K \geq S_T$  cannot be satisfied very early. To overcome this problem, we can force to reduce the estimated value of  $S_T$  such that  $S_K \geq S_T$  can be satisfied early. However, in such cases, we can no longer achieve the exact top- $k$  documents but only the approximate top- $k$  documents. We show the query efficiency and error rate of query results in such cases in Table 6-3 and Table 6-4 respectively, where we stop the query whenever the following condition is satisfied,

$$S_K \geq \theta \cdot S_T \quad (6.1)$$

where  $\theta < 1$ . That is, we force to reduce the estimated  $S_T$  to  $\theta S_T$ . The results for different  $\theta$  values are shown in Table 6-3 and Table 6-4. In Table 6-3, the bolded results are the best efficiency we can achieve when the corresponding error rate is below 1.0%.

Table 6-3. Results of theta-approximation (metric: ratio, dataset: GOV; query set: 2004mixed;  $\alpha=0.2$ ;  $k=5$ )

Index	$\theta=0.8$	$\theta=0.85$	$\theta=0.9$	$\theta=0.95$	$\theta=1.0$
TSR	100%	100%	100%	100%	100%
MSI	16.7%	20.0%	<b>23.6%</b>	28.4%	31.7%
SSI	12.8%	15.9%	19.6%	24.1%	<b>27.1%</b>
MST	40.2%	49.6%	<b>58.7%</b>	66.9%	84.9%

Table 6-4. Error rate of the theta-approximation (dataset: GOV; query set: 2004mixed;  $\alpha=0.2$ ;  $k=5$ )

Index	$\theta=0.8$	$\theta=0.85$	$\theta=0.9$	$\theta=0.95$	$\theta=1.0$
TSR	0%	0%	0%	0%	0%
MSI	9.20%	6.25%	0.04%	0.01%	0%
SSI	9.20%	7.05%	4.11%	1.25%	0%
MST	4.20%	1.88%	0.71%	0.09%	0%

### 6.2.5 Analysis by query categories

In Figure 6-3, we compare the query performance on the SSI indexes for queries with different lengths. We can see that short queries can benefit more from the SSI indexes than long queries.

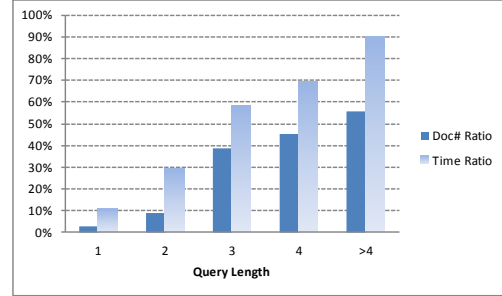


Figure 6-3. Results on the SSI indexes for queries with different lengths (index: SSI; dataset: GOV; query set: 2004mixed;  $\alpha=0.2$ ;  $k=3$ )

In Figure 6-4, we compare the query performance on the SSI indexes for the queries with different numbers of relevant documents. It is reasonable that early stopping can easily be achieved for the queries with more relevant results. In the extreme case that only  $k$  documents are relevant, no methods would achieve early termination.

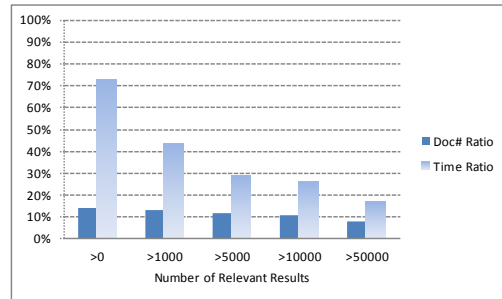


Figure 6-4. Query Performance on the SSI indexes for queries associated with different number of relevant documents (index: SSI; dataset: GOV; query set: 2003np;  $\alpha=0.3$ ;  $k=3$ )

## 7. CONCLUSION AND FUTURE WORK

In this paper, we have studied early termination techniques for the efficient document retrieval on large-scale data sets. We proposed new techniques to build new inverted indexes based only on the term-independent information, in particular, the combination of static rank and UBIR or UBTF. We also discussed the new retrieval strategies on the resulting indexes. Our experimental results show that although the previous methods based solely on the static rank has been shown to be in effective to the early termination, our methods can achieve effective early termination and therefore significantly improve query efficiency.

There are still a few interesting open problems. First, we are currently studying to improve query efficiency when the term position information, or the term proximity, is considered in the ranking function. Second, it will be very interesting to study how to combine our methods with the methods based on term-dependent information to achieve the best overall query performance. Finally, we want to study the impacts on our methods of other factors, such as query features and caching policies.

## 8. REFERENCES

- [1] V. Anh, O. de Kretser, and A. Moffat. Vector-space ranking with effective early termination. In SIGIR 2001.
- [2] V. Anh and A. Moffat. Compressed inverted files with reduced decoding overheads. In SIGIR 1998.
- [3] V. Anh and A. Moffat. Pruned query evaluation using pre-computed impact scores. In SIGIR 2006.
- [4] R. Baeza-Yates and B. Ribeiro-Neto. Modern information retrieval. Addison Wesley, 1999.
- [5] A. Borodin, G. Roberts, J. Rosenthal, and P. Tsaparas. Finding authorities and hubs from link structures on the world wide web. In Proc. of the 10th Intl. Conf. on World Wide Web, 2001.
- [6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In Proc. of the 7th Intl. Conf. on World Wide Web, 1998.
- [7] A. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien. Efficient query evaluation using a two-level retrieval process. In Proc. of the 12th Conf. on Information and Knowledge Management, pages 426–434, Nov 2003.
- [8] C. Buckley and A. F. Lewit. Optimization of inverted vector searches. In Proc. of the 8th Annual SIGIR Conf. on Research and Development in Information Retrieval, pages 970–110, 1985.
- [9] S. Büttcher and C. Clarke. A document-centric approach to static index pruning in text retrieval systems. In Proc. of the 15th ACM international Conference on information and Knowledge, 2006
- [10] D. Carmel, et al. Static index pruning for information retrieval systems. In Proc. of the 24th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval, pages 43–50, New Orleans, Louisiana, USA, September 2001.
- [11] G. Das, D. Gunopulos, N. Koudas, and D. Tsirogiannis. Answering top-k queries using views. In VLDB 2006.
- [12] R. Fagin, A. Lotem, and M. Naor. Optimal Aggregation Algorithms for Middleware. JCSS, 66(4):614–656, 2003.
- [13] R. Fagin. Combining fuzzy information from multiple systems. JCSS, 58(1):83–99, 1999.
- [14] R. Fagin. Combining fuzzy information: an overview. *SIGMOD Rec.*, 31(2):109–118, 2002.
- [15] U. Güntzer, W. Balke, and W. Kiebling. Optimizing multi-feature queries for image databases. In Proc. of the 26th Int. Conf. on Very Large Data Bases, pages 419–428, 2000.
- [16] D. Harman and G. Candela. Retrieving records from a gigabyte of text on a mini-computer using statistical ranking. JASIS, 41(8):581–589, 1990.
- [17] M. Kaszkiel, J. Zobel, and R. Sacks-Davis. Efficient passage ranking for document databases. ACM Transactions on Information Systems, 17(4):406–439, Oct. 1999.
- [18] R.Kumar, K. Punera, T. Suel and S. Vassilvitskii, Top-k aggregation using intersections of ranked inputs, In WSDM 2009.
- [19] X. Long and T. Suel. Optimized query execution in large search engines with global page ordering. In Proc. of the 29th Int. Conf. on Very Large Data Bases, September 2003.
- [20] X. Long and T. Suel. Three-level caching for efficient query processing in large web search engines. In Proc. of the 14th Intl. Conf. on World Wide Web, pages 257–266, 2005
- [21] A. Moffat and J. Zobel. Fast ranking in limited space. In Proc. of the 10th IEEE Intl. Conf. on Data Engineering. Houston, TX, February 1994.
- [22] E. de Moura et al. Improving web search efficiency via a locality based static pruning method. In Proc. of the 14th Intl. Conf. on World Wide Web, pages 235–244, 2005.
- [23] S. Nepal and M. V. Ramakrishna. Query processing issues in Image (multimedia) Databases. In Proc. of the 15th ICDE, pages 22–29, 1999.
- [24] M. Persin, J. Zobel, and R. Sacks-Davis. Filtered document retrieval with frequency-sorted indexes. JASIS, 47(10):749–764, 1996.
- [25] Y. Rasolofoa and J. Savoy. Term proximity scoring for keyword-based retrieval systems. In ECIR 2003.
- [26] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In Proc. of the 3rd Text Retrieval Conference (TREC), Nov 1994
- [27] R. Schenkel, A. Broschart, S. Hwang, M. Theobald and G. Weikum. Efficient text proximity search. In SPIRE 2007
- [28] G. Skobeltsyn, F. Junqueira, V. Plachouras, R. Baeza-Yates: ResIn: a combination of results caching and index pruning for high-performance web search engines. In SIGIR 2008.
- [29] Y. Tsegay, A. Turpin, and J. Zobel. Dynamic index pruning for effective caching. In Proc. of the ACM 16th Conference on Information and Knowledge Management, 2007
- [30] H. Turtle and J. Flood. Query evaluation: strategies and optimizations. Information Processing and Management, 31(6):831–850, 1995.
- [31] I. Witten, A. Moffat, and T. Bell. Managing gigabytes: compressing and indexing documents and images. Morgan Kaufmann, second edition, 1999
- [32] W. Wong and D. Lee. Implementation of partial document ranking using inverted files. Information Processing and Management, 29(5):647–669, 1993.
- [33] H. Yan, S. Ding and T. Suel. Compressing term positions in web indexes, In Proc. of the 32nd Annual SIGIR Conf. on Research and Development in Information Retrieval, Boston, July, 2009
- [34] M. Zhu, S. Shi, M. Li, and J.-R. Wen. Effective top-K computation in retrieving structured documents with term-proximity Support. In CIKM 2007.
- [35] M. Zhu, S. Shi, N. Yu, J.-R. Wen. 2008. Can phrase indexing help to process non-phrase queries? In CIKM 2008.
- [36] J. Zobel and A. Moffat. 2006. Inverted files for text search engines. ACM Computing Surveys. Vol. 38, No 2, Jul. 2006.

## APPENDIX

Proof of the Formula 4.2 in Section 4:

For a query  $q$ , denote  $n$  as the number of documents contained in the intersection of the inverted lists corresponding to the query. Define  $M$  to be the random variable representing the number of documents accessed when the query processing algorithm stops. Our objective is to compute the average number of documents accessed before the algorithm stops. In other words, we need to estimate  $E(M)$ , the expected value of  $m$  given a fixed  $n$ ,

$$E(M) = \sum_{m=1}^n m \cdot P(M = m) \quad (A1)$$

where  $P(M=m)$  is the probability of the random variable  $M$  taking value  $m$ . To compute the probability, consider the time when we have just processed  $m$  documents and are about to test the early-stopping condition (Figure A-1). Denote  $R_m$  to be the random variable representing the static-rank of the  $m$ -th document in the intersection of the inverted lists. Assume that the static-rank of the last processed document is  $r$ . Let's compute the probability of the early-stopping condition being satisfied given  $m$  and  $r$ , represented by  $P(S_k \geq S_T | M = m, R = r)$  or  $P(S_k \geq S_T | m, r)$  in short.

As the documents are ordered by static-rank, the static-rank value of every un-accessed document should be at most  $r$ , therefore,

$$S_T = \alpha \cdot r + \beta \cdot 1.0 \quad (\text{A2})$$

For a document  $d$  in the first processed  $m-1$  documents (not including the last processed one), define its static-rank and IR score to be  $g$  and  $s$  respectively. According to our assumptions,  $g$  is uniformly distributed in  $[r, 1]$ , and  $s$  is uniformly distributed in  $[0, 1]$ . According to Formula A2 and Figure A-2, we have,

$$\begin{aligned} P(d.\text{score} \geq S_T) &= P(\alpha g + \beta s \geq \alpha r + \beta) \\ &= \alpha(1-r)/2\beta \end{aligned} \quad (\text{A3})$$

For the last accessed document  $d^*$ ,

$$P(d^*.\text{score} \geq S_T) = P(\alpha r + \beta s \geq \alpha r + \beta) = 0 \quad (\text{A4})$$

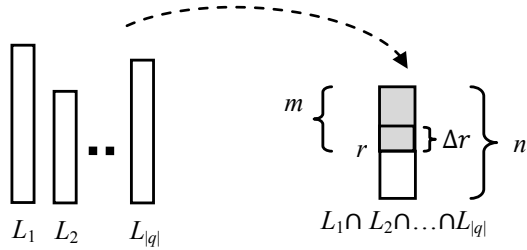


Figure A-1. Theoretical analysis

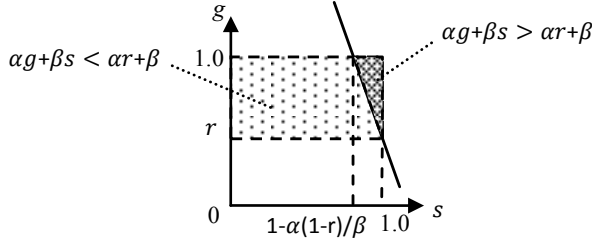


Figure A-2. Computing the probability of  $\alpha g + \beta s > \alpha r + \beta$

For  $k=1$  (i.e., computing the top-1 results), the query processing algorithm stops if and only if there exists at least one processed document whose score is not less than  $S_T$ . According to Formulas A3 and A4, the probability of the early-stopping condition being satisfied is,

$$\begin{aligned} P(S_k \geq S_T | m, r) &= 1 - \left( \prod_{i=1}^{m-1} P(d.\text{score} < S_T) \right) \\ &\quad \cdot P(d^*.\text{score} < S_T) \\ &= 1 - \left( 1 - \frac{\alpha(1-r)}{2\beta} \right)^{m-1} \end{aligned} \quad (\text{A5})$$

So the probability of the early stopping condition *NOT* being satisfied is,

$$P(S_k < S_T | m, r) = \left( 1 - \frac{\alpha(1-r)}{2\beta} \right)^{m-1} \quad (\text{A7})$$

For a given  $m$  value, the probability that the static-rank of the  $m$ -th document is in a small range  $[r, r+\Delta r]$  (where  $\Delta r \ll 1$ ) is,

$$P(r \leq R_m \leq r+\Delta r) = \binom{n}{n-m} \cdot r^{n-m} \cdot \binom{m}{1} \cdot \Delta r \cdot (1-r)^{m-1} \quad (\text{A8})$$

Therefore the probability density function of  $R_m$  is,

$$f(r) = m \cdot \binom{n}{n-m} \cdot (1-r)^{m-1} \cdot r^{n-m} \quad (\text{A9})$$

It can be verified that  $\int_0^1 f(r) dr = 1.0$ .

Let  $F(m)$  denote the probability that the early-stopping condition is *NOT* satisfied just after processing the  $m$ -th document, then

$$F(m) = \int_0^1 P(S_k < S_T | m, r) f(r) dr \quad (\text{A10})$$

Given the definition of  $F(m)$ , the probability that the algorithm stops *exactly* at the  $m$ -th document is,

$$P(M = m) = \begin{cases} 1 - F(1) & \text{if } m = 1 \\ F(m-1) - F(m) & \text{if } 1 < m < n \\ F(n-1) & \text{if } m = n \end{cases} \quad (\text{A11})$$

We have the above formula because the algorithm stops exactly at  $m$  ( $1 < m < n$ ) if and only if the early-stopping condition does *not* hold after processing the  $(m-1)$ 'th document but holds after the  $m$ 'th document. By substituting Formula A11 into Formula A1, we have,

$$\begin{aligned} E(m) &= \sum_{m=1}^n m \cdot P(M = m) \\ &= 1 + \left( \sum_{m=2}^{n-1} m \cdot F(m-1) \right) - \left( \sum_{m=1}^{n-1} m \cdot F(m) \right) + n \cdot F(n-1) \\ &= 1 + \sum_{m=1}^{n-1} F(m) \end{aligned} \quad (\text{A12})$$

By substituting Formulas A7, A9, and A10 into Formula A12 and letting  $x=1-r$ , we obtain,

$$\begin{aligned} E(m) &= 1 + \sum_{m=1}^{n-1} F(m) \\ &= 1 + \sum_{m=1}^{n-1} \int_0^1 P(S_k \geq S_T | m, r) f(r) dr \\ &= 1 + \sum_{m=1}^{n-1} \int_0^1 \left( 1 - \frac{\alpha(1-r)}{2\beta} \right)^{m-1} \cdot m \cdot \binom{n}{n-m} \cdot (1-r)^{m-1} \cdot r^{n-m} dr \\ &= 1 + \sum_{m=1}^{n-1} \int_0^1 m \cdot \binom{n}{n-m} \cdot \left( 1 - \frac{\alpha x}{2\beta} \right)^{m-1} x^{m-1} \cdot (1-x)^{n-m} dx \\ &= 1 + \int_0^1 \left[ \sum_{m=1}^{n-1} m \cdot \binom{n}{n-m} \cdot \left( x - \frac{\alpha x^2}{2\beta} \right)^{m-1} (1-x)^{n-m} \right] dx \\ &= 1 + n \cdot \int_0^1 \left[ \sum_{m=1}^{n-1} \binom{n-1}{n-m} \cdot \left( x - \frac{\alpha x^2}{2\beta} \right)^{m-1} (1-x)^{n-m} \right] dx \\ &= 1 + n \cdot \int_0^1 \left( 1 - \frac{\alpha x^2}{2\beta} \right)^{n-1} dx \end{aligned} \quad (\text{A13})$$

Thus Formula 4.2 is proved.  $\blacksquare$

It may not be easy to compute the above integral analytically. Alternatively, we adopt numeric methods to compute the value. In a little more detail, we divide the  $[0, 1]$  space to  $M$  small pieces ( $M=100n$  here) and compute the total area of the pieces.