Implementations of Partial Document Ranking Using Inverted Files

Wai Yee Peter Wong
Dik Lun Lee
Department of Computer and Information Science, Ohio State University,
2036 Neil Ave, Columbus, Ohio 43210, U.S.A.

May 1992

Abstract - Most commercial text retrieval systems employ inverted files to improve retrieval speed. This paper concerns with the implementations of document ranking based on inverted files. Three heuristic methods for implementing the $tf \times idf$ weighting strategy, where tf stands for term frequency and *idf* stands for inverse document frequency, are studied. The basic idea of the heuristic methods is to process the query terms in an order so that as many top documents as possible can be identified without processing all of the query terms. The first heuristic was proposed by Smeaton and van Rijsbergen (Smeaton & Rijsbergen, 1981), and it serves as the basis for comparison with the other two heuristic methods proposed in this paper. These three heuristics are evaluated and compared by experimental runs based on the number of disk accesses required for *partial document ranking*, in which the returned documents contain some, but not necessarily all, of the requested number of top documents. The results show that the proposed heuristic methods perform better than that proposed by Smeaton and van Rijsbergen in terms of *retrieval accuracy*, which is used to indicate the percentage of top documents obtained after a number of disk accesses. For *total document ranking*, in which all of the requested number of top documents are guaranteed to be returned, no optimization techniques studied so far can lead to substantial performance gain. To realize the advantage of the proposed heuristics, two methods for estimating the retrieval accuracy are studied. Their accuracies and processing costs are compared. All the experimental runs are based on four test collections made available with the SMART system.

1 Introduction

Boolean search strategies are used in most commercial text retrieval systems, but their drawbacks are well-known (Cooper, 1988; Radecki, 1988; Salton, Fox & Wu, 1983; Salton & McGill, 1983). For example, formulation of useful boolean queries is not easy to learn, and users often have difficulty in controlling the output size. In order to improve retrieval effectiveness, vector processing systems employing similarity measures have been suggested and studied extensively (Buckley & Lewit, 1985; Croft & Savino, 1988; Salton, 1989; Salton & McGill, 1983). In a vector processing system, both query and document terms can be weighted to distinguish terms which are more important for retrieval purposes from those which are less important. Let V be the size of the vocabulary for the document collection, a document D_i is represented as a V-dimensional vector $< w_{i,1}, w_{i,2}, ..., w_{i,V} >$, where $w_{i,j}$ represents the weight of term j in document i. Likewise, a query Q can be represented as a V-dimensional vector $< q_1, q_2, \ldots, q_V >$, where q_i specifies the weight of term i in the query. The similarity between a query and the documents can be computed in order to rank the retrieved documents in decreasing order of the query-document similarity. Many similarity measures have been studied (Salton, 1989). For instance, the *cosine coefficient* measures the similarity between the query and document vectors based on the cosine of the angle θ between them in the V-dimensional space:

$$similarity(D_i, Q) = cos(\theta) = \frac{\sum_{j=1}^{V} w_{i,j} \cdot q_j}{\sqrt{\sum_{j=1}^{V} w_{i,j}^2 \cdot \sum_{j=1}^{V} q_j^2}.}$$

One of the well-known weighting strategies uses so-called $tf \times idf$ weights, in which $w_{i,j} = tf_{D_i}(t_j) \cdot idf(t_j)$, for $1 \leq j \leq V$. The term $tf_{D_i}(t)$ represents the term frequency of term t in document D_i (i.e., the number of occurrences of term t in document D_i). The function $idf(t)$ is called the inverse document frequency of term t and is set to $\log_2(N/df(t))$, where N is the number of documents in a collection and $df(t)$ is the document frequency of term t (i.e., the number of documents in which term t is contained) (Croft & Savino, 1988; Salton, 1986). Thus, a term has a high weight in a document if it occurs frequently in the document but infrequently in the rest of the collection.

The vector processing system allows a query to be expressed as a natural language text describing the user's information need. Thus, the description can be treated as a short document so that q_i's can be expressed in $tf \times idf$ weights as well. Since most query terms are likely to appear only once in a short passage, tf's can be assumed to be 1 in the query and the weights of the query terms are sometimes represented by the idf's of the terms. To further reduce the computational cost, q_i can often be simplified into a binary value with "1" signifying the presence of term i in the query and "0" its absence (Stanfill & Kahle, 1986).

Conceptually, the $tf \times idf$ ranking strategy is very simple; however, it has been shown to give good retrieval effectiveness. Implementations of document ranking are studied extensively (Croft & Savino, 1988; Lucarella, 1988; Mohan & Willett, 1985; Murtagh, 1982; Perry & Willett, 1983; Salton, 1968; Shasha & Wang, 1990; Stanfill & Kahle, 1986; Stanfill, Thau & Waltz, 1989; Weiss, 1981; Wong & Lee, 1990); much work is based on inverted files (Buckley & Lewit, 1985; Stanfill, Thau & Waltz, 1989). An inverted file consists of two components, namely the index file and the postings file. Each item in the index file corresponds to a document term in the collection, and it is associated with a postings list in the postings file, which is usually stored on the disk. Each posting records the document which contains the term and some other information such as the corresponding term frequency, depending on the retrieval environement. An inverted file is shown in Figure 1, in which the document frequency of each term is stored in the index file, which can be used to compute the idf value. A query will be presented as a list of terms with associated weights. The postings lists corresponding to the query terms are retrieved and, from the postings lists, the document scores can be computed as shown in the pseudo-code below.

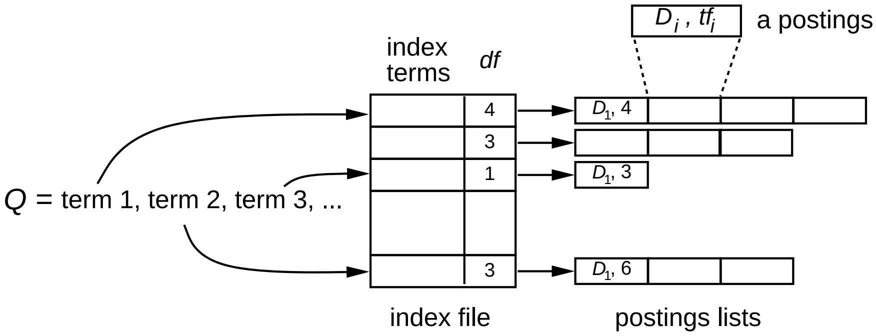

Figure 1: Inverted file supporting vector processing.

```
initialize all document scores to zero
for all q in Q do
    retrieve the postings list for term q
    for each posting < D_i, tf_i,q > in the postings list do
        compute score(D_i) = score(D_i) + tf_i,q · idf_i
    end {for}
end {for}
sort document scores and return top documents
```

The algorithm is rather straightforward since it essentially sums up the weights of the terms specified in the query for each document, taking advantage of the availiability of an inverted index. It should be noted that this algorithm is simplified by assuming binary query weights and no normalization, but it can be easily extended to remove the simplifications.

Many optimization techniques on inverted file systems have been developed to reduce the I/O cost (Buckley & Lewit, 1985; Perry & Willett, 1983; Smeaton & Rijsbergen, 1981). All of these methods process query terms one by one and accumulate partial scores for the documents, rather than compute the final score of a document completely before proceeding to the next document. These methods are motivated by the fact that the number of query terms can be very large when thesaurus and relevance feedback are used to expand the original query (Stanfill & Kahle, 1986). For instance, a query of 10 terms could be expanded to more than 100 terms, when synonyms and related terms (i.e., narrower terms and broader terms) are included. Thus, the corresponding processing cost becomes significant and needs to be reduced. The basic idea behind the optimization techniques is to process the query terms in an order so that the requested number of top documents[1] can be identified without processing all of the query terms. Most of the previous methods aim at *total document ranking* (or *total ranking*), in which all of the requested number of top documents are guaranteed to be returned. In this paper, we focus on *partial document ranking* (or *partial ranking*) (Buckley & Lewit, 1985; Perry & Willett, 1983; Weiss, 1981), in which the retrieved documents contain some, but not necessarily all, of the requested number of top documents. The importance of partial ranking is two-fold. First, when total ranking is implemented, none of the heuristic methods studied so far can yield any significant performance gain (Buckley & Lewit, 1985; Perry & Willett, 1983). Second, document retrieval itself is inherently imprecise. That is, even when total ranking is implemented, it is known that not every top document returned is actually relevant from the user's perspective. Effectiveness should not suffer greatly if the system can return a major portion of the top documents (Buckley & Lewit, 1985), since other techniques

[1] The top documents with respect to the query are those which have the highest scores according to some similarity measure. However, they are not guaranteed to be relevant from the user's perspective.

such as relevance feedback can be used to further improve the degree of relevance. Partial ranking is a profitable approach if it can yield substantial savings in processing costs.

The organization of this paper is as follows. In Section 2, three heuristic methods, called the L method, W method and SW method, are described, and the notion of *retrieval accuracy* is introduced. Their performance in implementing the $tf \times idf$ weighting strategy are evaluated and compared based upon experimental runs on the four test collections made available with the SMART system. In Section 3, two heuristic methods, called the *document movement method* (or simply Dm method) and the *linear regression method* (or simply Lr method), are proposed to predict the number of top documents obtained at different points of the retrieval process for the W method and SW method. The accuracies and processing costs of the Dm method and Lr method are compared. The last section summarizes the merits of the proposed methods.

2 Heuristic methods

As shown by previous studies (Buckley & Lewit, 1985; Perry & Willett, 1983) and further supported by our study, total ranking cannot be achieved until almost all query terms are processed. The goal of partial ranking is to maximize the number of top documents obtained after processing only a subset of the query terms. In other words, the goal of partial ranking technique is to be able to foresee the final document ranking (usually an approximate one) when only a subset of query terms is processed. Clearly, the chance of getting top documents fast depends on how fast document scores are incremented as query terms are processed. This in turn is affected by the processing order of the query terms. In Fig. 1, the *idf*'s of term 1, term 2 and term 3 are 4.6, 4.0 and 6.6 respectively, assuming that $N = 100$. If the query terms are processed in the order as presented in the query, the score of D_1 after processing each query term will be 18.4, 42.4, and 62.2. If the query terms are ordered by increasing document frequencies, the document score will be 19.8, 43.8, and 62.2. With query terms ordered by decreasing term frequencies, the document score will be 24, 42.4, and 62.2. The rationale is that the faster the document scores of top documents increment, the earlier they can be identified. In this particular example, either ordering by document frequencies or by term frequencies is better than the original order.

In this section, we study three methods which process query terms in different orders based upon different criteria. The first method, called L method, was proposed by Smeaton and van Rijsbergen (Smeaton & Rijsbergen, 1981), the second and third methods, called the W and SW methods respectively, are proposed by the authors (Lee & Wong, 1991; Wong & Lee, 1991).

2.1 The L method

The L method, which has been used in the upperbound search algorithm (Buckley & Lewit, 1985; Croft & Savino, 1988; Fukunaga & Narendra, 1975; Mohan & Willett, 1985; Perry & Willett, 1983; Smeaton & Rijsbergen, 1981; Weiss, 1981), processes query terms in descending order of their *idf* values. Since document frequencies correspond to the lengths of the postings lists, in other words, query terms are processed in increasing lengths of the postings lists. This method requires the lengths of the postings lists to be kept in the index file, which can be accessed separately from the postings lists. For simplicity, Q is represented by $(q_1, q_2, \ldots, q_k)$, in which $df(q_i) \leq df(q_j)$ for $i < j$. The upperbound search algorithm works as follows. After a postings list is processed, partial document scores are sorted in descending order and the current top T documents are obtained. The upperbound of the $(T+1)$st document is then computed with the assumption that it contains all the remaining unsearched query terms. The retrieval process stops if the upperbound of the $(T+1)$st document is smaller than the current score of the Tth document. If the search can be stopped

	CACM	CISI	CRAN	MED
Number of documents	3204	1460	1400	1033
Number of distinct terms	17137	7731	3763	6927
Average number of terms / document	37	101	53	52
Maximum document frequency (*df*)	1316	659	860	359
Average *df*	6.87	19.98	19.74	7.69
Standard deviation of *df*	25.42	40.02	54.25	19.74
Maximum tf	59	82	27	25
Maximum *idf*	11.65	10.51	10.45	10.01

Table 1: Characteristics of the document collections.

early, the amount of disk accesses and CPU processing cost can be reduced substantially, since short postings lists are processed first and the unprocessed lists are long.

In order to evaluate the usefulness of the upperbound search algorithm by using the above sorting method, we take checkpoints during the retrieval process. To facilitate discussions, three definitions are given:

1. Q_f: The final set of top documents obtained when all the query terms in query Q are processed. The number of top documents returned is determined by the system or by the user.
2. Q_i: The set of top documents obtained after processing $i\%$ of the total disk accesses in response to query Q.
3. Ra_i *(Retrieval accuracy)*: $(\mid Q_i \cap Q_f \mid / \mid Q_f \mid) \cdot 100\%$.

The values of *Ra*'s indicate the percentages of top documents obtained at different points of the retrieval. Moreover, a sequence of *Ra*'s can show how fast the top documents in Q_f are revealed. For instances, if $Ra_{10} = 0.5$, $Ra_{20} = 0.7$ and $Ra_{30} = 0.85$, then it indicates that the top documents are revealed fast. If $Ra_{10} = 0.1$, $Ra_{20} = 0.15$ and $Ra_{30} = 0.20$, then it indicates that the top documents are revealed rather slowly. For partial ranking, the rate at which *Ra* increases should be the faster the better, since the retrieval process can stop earlier for the same accuracy. We will study the retrieval accuracy of the L method by an experiment carried out on four test collections, which are made available with the SMART system. Some characteristics of the test collections are given in Table 1.

The experiment is performed in the following manner. For each collection, 15 queries are generated for each of the three different query sizes, namely 30, 50 and 100 terms, for a total of 45 queries. For each query size, three groups of queries are generated with 5 queries each. In the first group, each query contains only short postings lists (i.e., only contains query terms with high *idf* values).[2] In the second group, each query contains one-half short postings lists and one-half long postings lists. In the third group, each query contains only long postings lists. Query terms in each query are randomly selected from the vocabulary of the respective test collection. With queries of different sizes and the corresponding postings lists of different lengths, our experiments do not try to favor any particular operational environment.

Ten equally spaced checkpoints are taken and the top 20 documents at each checkpoint are recorded. It means that if a total of P disk pages is required for processing all the query terms, checkpoints are taken after $\lceil P/10 \rceil, \lceil 2P/10 \rceil, \lceil 3P/10 \rceil, \ldots$, and $\lceil 10P/10 \rceil$ disk pages have been processed. Finally, for each collection, we obtain at each checkpoint the retrieval accuracies for all the queries and compute the average of the retrieval

[2] In this paper, a postings list is short if it contains fewer than 50 document postings, long otherwise.

accuracies for queries of the same size. In our study, each document posting requires 6 bytes, 4 bytes for a document identifier and 2 bytes for a term frequency. In most cases, each disk access retrieves 20 document postings, corresponding to 120 bytes. The reason for this rather small page size is to account for the small document database used in the experiment; if the page size is too large (e.g., 1K), most postings list will fit in one page, which is not realistic for large databases.

The results are shown in Figs. 2-5 for CACM, CISI, CRAN and MED, respectively. In each collection, three curves ($\odot$) in three diagrams are shown, for 30-term, 50-term and 100-term queries. Most of the curves are roughly lying at about the 45 degree line. In general, almost all of the query terms must be processed in order to obtain all of the documents in Q_f. This indicates that the stopping criterion specified by the upperbound search algorithm is hard to meet and no significant performance improvement can be achieved by the L method if total ranking is required. Our result is consistent with the study by Buckley and Lewit (Buckley & Lewit, 1985).

In the $tf \times idf$ weighting strategy, document weights are determined by both the tf and *idf* values. For typical document collections, the range of *idf* values is rather small since their values are compressed by the log function. Thus, *idf* is the secondary component for determining the weight of a term, when compared to tf, whose range is larger in general (see Table 1). However, the L method determines the processing order of query terms based on *idf* values alone, without taking tf values into consideration. This explains why the retrieval accuracy of the L method increases rather slowly.

In fact, the upperbound search algorithm is too conservative to yield any significant performance gain. First, it is highly unlikely for the $(T+1)$st document to contain all the remaining unsearched terms. Second, the upperbound method considers the chance of the $(T+1)$st document being promoted to a top document, assuming that the weights of the other documents, including the current top documents, are not changed. This assumption is unrealistic and will be too pessimistic as a stopping criterion.

Two methods have been proposed to improve retrieval efficiency by relaxing the stopping criterion (Buckley & Lewit, 1985; Perry & Willett, 1983; Weiss, 1981). The first one only guarantees a subset of top documents to be returned (i.e., only partial ranking is performed). The upperbound of the $(T+1)$st document is compared with the current score of the Sth document, where $S < T$. With this relaxation, the chance to meet the stopping criterion is higher because of a larger difference in scores (Buckley & Lewit, 1985). The second one applies probability on the upperbound computation (Perry & Willett, 1983; Weiss, 1981). The process may stop early if the probability for the upperbound of the $(T+1)$st document to exceed the current score of the Tth document is small. But we find that the probabilistic stopping criterion is still unlikely to be met because the score difference between the Tth and $(T+1)$st documents is usually very small. Figure 6 shows the document scores of the top 30 documents of one run on MED. There is no significant gap between two consecutive scores. Even if a gap exists, it may not fall between the Tth and $(T+1)$st documents to trigger early stopping. Thus, the $(T+1)$st document has a high probability to become a top document until almost all the query terms have been processed. Moreover, the saving due to this method is still questionable since the cost of computing probabilities with a large number of terms could be very significant. Thus, the upperbound computation not only fails to produce any performance gain but also induces computational overhead.

These problems motivate our study on partial ranking. To overcome the shortcomings of the L method, we propose and investigate two search algorithms in the next two subsections. The idea of the methods is based upon greedy algorithms, and their objective is to improve Ra's, especially at the initial stage of the retrieval process, i.e., to obtain a large portion of top documents fast without a large amount of I/O operations.

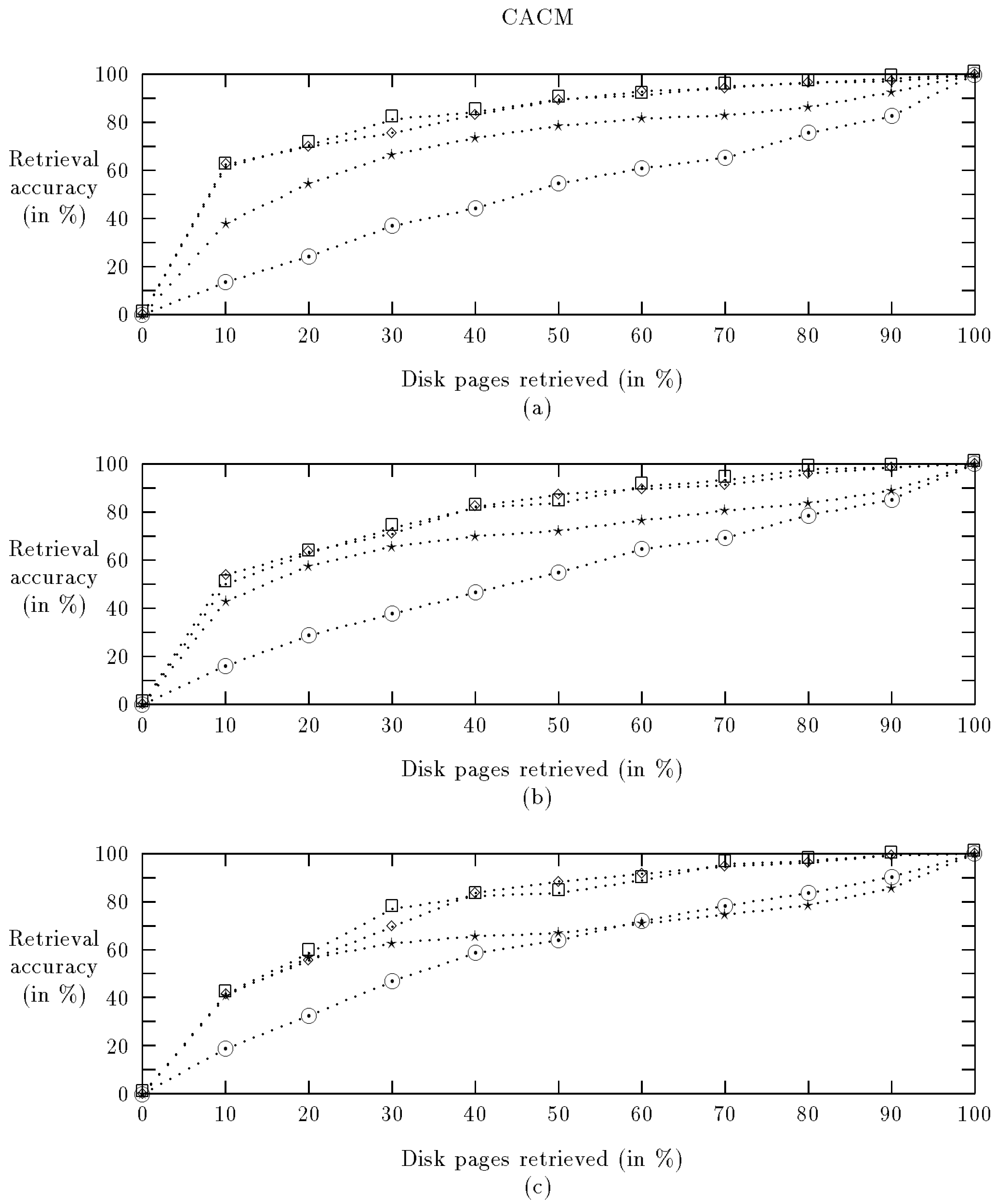

Figure 2: Average retrieval accuracies of the L method ($\odot$), W method ($\star$) and SW method ($\diamond$ [120 bytes/page], $\square$ [240 bytes/page]) for CACM. (a) 30 query terms per query. (b) 50 query terms per query. (c) 100 query terms per query. Fifteen queries are tested for each query size.

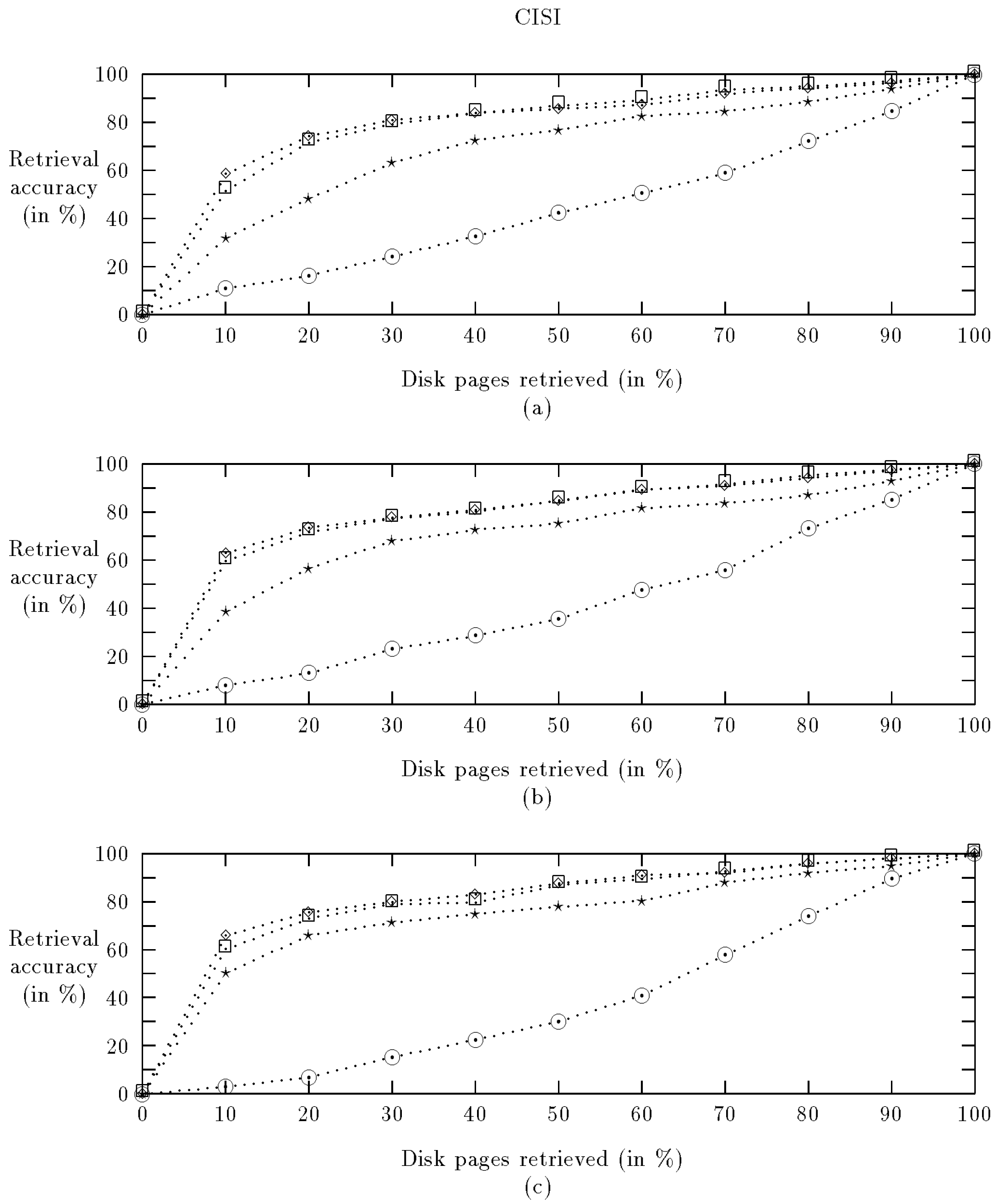

Figure 3: Average retrieval accuracies of the L method ($\odot$), W method ($\star$) and SW method ($\diamond$ [120 bytes/page], $\Box$ [240 bytes/page]) for CISI. (a) 30 query terms per query. (b) 50 query terms per query. (c) 100 query terms per query. Fifteen queries are tested for each query size.

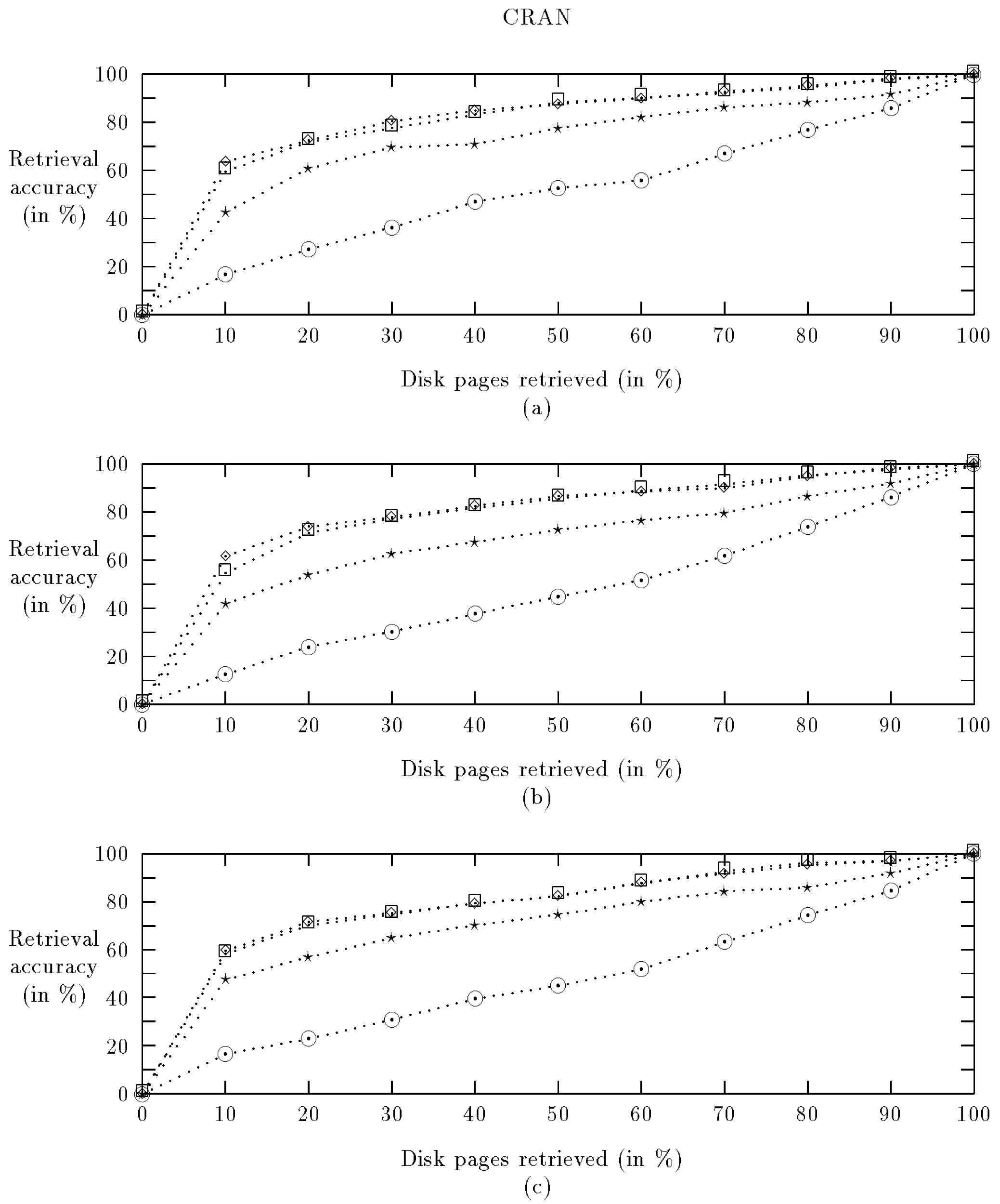

Figure 4: Average retrieval accuracies of the L method ($\odot$), W method ($\star$) and SW method ($\diamond$ [120 bytes/page], $\square$ [240 bytes/page]) for CRAN. (a) 30 query terms per query. (b) 50 query terms per query. (c) 100 query terms per query. Fifteen queries are tested for each query size.

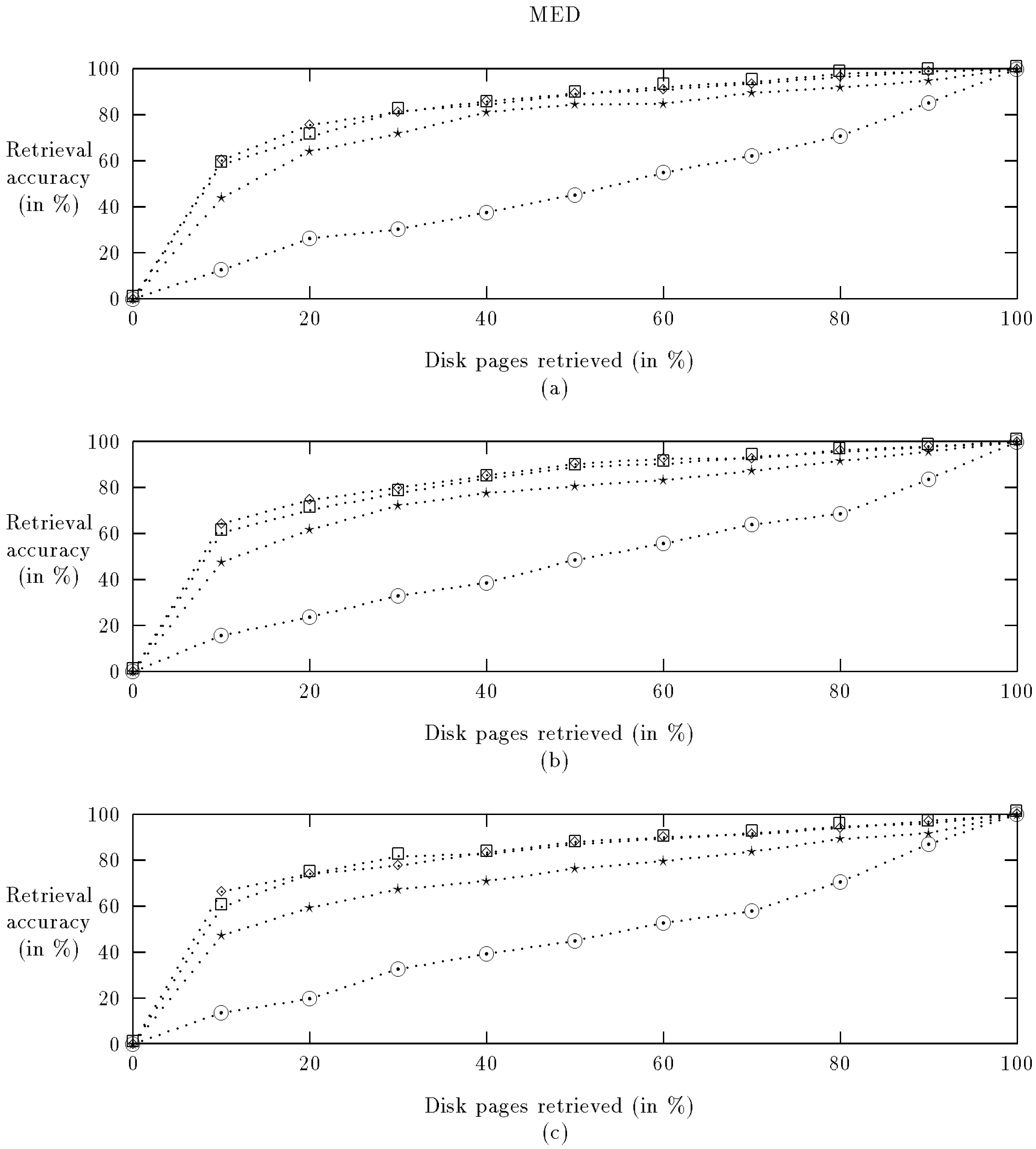

Figure 5: Average retrieval accuracies of the L method ($\odot$), W method ($\star$) and SW method ($\diamond$ [120 bytes/page], $\square$ [240 bytes/page]) for MED. (a) 30 query terms per query. (b) 50 query terms per query. (c) 100 query terms per query. Fifteen queries are tested for each query size.

	id	score		id	score		id	score
1	41	63.9	11	597	37.9	21	133	35.7
2	620	60.2	12	388	37.6	22	88	35.6
3	581	52.4	13	208	37.6	23	721	35.1
4	460	49.6	14	1025	37.6	24	197	34.9
5	390	48.2	15	902	37.4	25	594	34.8
6	77	45.1	16	240	37.0	26	603	34.7
7	697	40.9	17	137	36.7	27	573	33.7
8	505	40.1	18	272	36.6	28	608	33.2
9	344	40.1	19	457	36.5	29	837	33.2
10	681	38.2	20	463	35.8	30	120	33.2

Figure 6: Top 30 document scores of one run on MED.

2.2 The W method

The process of document ranking is to obtain and accumulate term weights of the documents from the postings lists, and to sort the final document scores in descending order. The W method takes two parameters into consideration, namely the maximum tf in a postings list (denoted by tf_{max}) and the length of that postings list. The query terms are then processed in descending order of the $tf_{max} \times idf$ values. In this case, the maximum tf value of each postings list is stored in the index file. Since two bytes are sufficient to store a term frequency, the additional storage overhead will not be significant. A typical entry in the inverted file is as follows:

term t_i	df	tf_{max}	$\longrightarrow$	d_j, tf_x	d_k, tf_y	$\cdots$

Since this method processes postings lists which have a high potential of generating large increments to the document scores, partial scores of top documents will be accumulated faster than the L method without using a large amount of I/O. Consequently, the top documents in Q_f will be revealed earlier in the retrieval process.

An experiment similar to that used for the L method was carried out to find out the retrieval accuracy of the W method. The results are plotted as ($\star$) curves in Figs. 2-5. Like the L method, most of the query terms need to be processed in order to obtain all of the documents in Q_f if total ranking is required, so the W method is still unable to obtain significant performance gain. However, the retrieval accuracies of the W method are consistently higher than those of the L method in all the test collections, except for the 100-term queries in CACM. It means that if partial ranking is allowed, the W method can lead to a substantial improvement in terms of the number of disk accesses, for a retrieval accuracy less than 100%. For instance, for the 100-term queries in CISI, after processing 30% of the disk accesses, the average Ra for the L method is less than 10%, while it is about 70% for the W method.

As seen in Figs. 2-5, the increments of Ra's in the W method are fast for the first 40% of disk accesses; however, the increments slow down as Ra approaches 100%. That is, the disk accesses after the first 40% increase the retrieval accuracy very slowly; and the cost-effectiveness of this portion of I/O operations is low. The major advantage of the W method is the ability to obtain a large number of top documents in a small amount of disk accesses, especially at the initial stage of the retrieval process, with a small additional storage overhead in the index file. However, this sorting method is still not the fastest one to obtain top documents for a given amount of disk accesses. In the following subsection, a more complex method is studied.

2.3 The *SW* method

The *SW* method involves a more complex ordering method than the first two methods. In the *W* method, postings lists are processed by decreasing order of $tf_{max} \times idf$ values. When a postings list of high $tf_{max} \times idf$ is retrieved, many disk pages with low term weights are retrieved and processed at the same time, but they contribute little to finding the top documents. Thus, processing the query Q list by list is not the best way to achieve high retrieval accuracy. To further utilize the idea of greediness, the *SW* search method is investigated.

In the *SW* method, the postings of each postings list are sorted by decreasing tf values first. In other words, for a given term, documents with high tf values are put at the beginning of the list, and those with low tf's are put at the end. Moreover, a postings list is no longer viewed as a single item, but rather as a sequence of individual disk pages. This organization allows disk pages of high term weights to be processed before those of low term weights. For instance, the first page of term t_i may be processed first, and then the first page of term t_j, and then the first page of t_k, and then the second page of term t_i, and so on. The maximum tf of each page is stored in the index file so that disk pages with high tf values can be determined from the index file. A typical entry in the inverted file is shown below:

term t_i	df	$tf_{1,max}$	$tf_{2,max}$	$\cdots$	$\longrightarrow$	d_j, tf_x	d_k, tf_y	$\cdots$

The processing and storage overheads of the *SW* will be higher than the first two methods. However, since most postings lists are short according to the Zipf's law (Zipf, 1949) and therefore will occupy only a small number of disk pages, the overheads in keeping the tf values and maintaining the order of the disk pages are insignificant. This is especially true for environments where updates are done in batch and are infrequent compared to retrieval.

Disk pages are processed in an order defined by three parameters, namely, the maximum tf of the page, the length of the postings list and the number of document identifiers in the page:

$$tf_{max} \times idf \times f(I),$$

where tf_{max} is the maximum tf contained in the disk page and I is the number of document identifiers in the page. The function $f(I)$ is included in the formula to account for the number of document identifiers in a disk page, which affects the contribution of a disk page to finding the top documents, especially when the weights of the top documents are determined by many terms. For example, a page having one identifier with a weight of 31 may not have a higher potential to accumulate weights than a page having 20 identifiers with the maximum weight of 29. First of all, the difference between the maximum weights of these two pages is small. Moreover, the former only increments the weight of one document, while the latter increments the weights of 20 documents. In this case, it is reasonable to process the disk page with the maximum weight of 29 first. The function $f(I)$ is introduced to account for the degree of fullness of disk pages in determining the order of processing. A number of functions have been investigated; in this study, we use $f(I) = I^e$, where $0 < e < 1$. The effect of I is restricted by e so that the maximum weights of disk pages are still the primary determining factor. Since this method requires the document identifiers of each postings list to be sorted by decreasing tf value, it is referred to as the *SW* method.

We study the *SW* method of different page sizes: 120 bytes and 240 bytes, corresponding to 20 postings and 40 postings per disk page. For the page size of 120 bytes, e is set to 0.5. In other words, we take a square root of I in the formula. As the page size is increased to 240 bytes (i.e., the maximum value of I is increased to 40), with the maximum tf in each postings list unchanged, we accordingly reduce e to 0.4 to restrict the significance of I to the processing order. The optimal value of e for a certain page size requires further investigation. Once again, an experiment similar to that used for the L method was carried out to test the retrieval accuracy of the *SW* method. The results are plotted as ($\diamond$) and ($\square$) curves for page sizes of 120 and 240 bytes in Figs. 2-5. We find that the *SW* method is better than the *W* method and *L* method in terms of retrieval accuracy in all the test collections for all query sizes, demonstrating the superior performance of this sorting method.

In Fig. 4, we find that if 80% of top documents in CRAN are required, about 90% of disk accesses are needed for the *L* method, 60% for the *W* method and 40% for the *SW* method. In Figs. 2-5, the average retrieval accuracies of the *SW* method are consistently better than those of the *W* method by about 5-20%. By observing the retrieval accuracies at 10-30% of disk accesses for the *SW* method, there are still plenty of room for improvement, which is left for future investigation. In the following section, two estimation methods for the *W* method and *SW* method are proposed to estimate *Ra*'s at different points of the retrieval process. Note that the retrieval accuracies in the previous experiments are obtained assuming that Q_f is known in advance, which is not the case in reality.

3 Estimations of retrieval accuracies for the *W* method and *SW* method

In order to realize the advantage of the *W* method and *SW* method, the system must be able to estimate *Ra*'s at different points of the retrieval process so that the system can stop the retrieval process when a desired retrieval accuracy has been reached. The terminating condition could be a threshold determined by the system or specified by the user. If the system fails to estimate the retrieval accuracy, we may have searched more postings lists than necessary, incurring unnecessary disk accesses, or stopped too soon, resulting in a lower accuracy than the user's specification. Estimations of retrieval accuracies can be done in a number of ways. For instance, the values of Ra_i's can be determined by carefully calibrating the retrieval system. This method is simple and produce good estimations for a stable database, but may not work well when the parameters of the database change frequently. Alternatively, an analytical model can be developed based upon the distribution functions of the tf values, *df* values, and the search strategy. An accurate analytical model should be able to give better prediction on Ra_i's under different system parameters. Unfortunately, it is difficult to derive. Instead, we study in this section two heuristic methods to estimate *Ra*'s at the checkpoints. It should be noted that they are not used to further improve the capability of obtaining better retrieval accuracy, but rather they are used to estimate the retrieval accuracy during a retrival process. The first method is called the document movement method (or the *Dm* method). It uses a heuristic to estimate whether a document is likely to be a top document based upon document movements during the retrieval process. The second one is called the linear regression method (or the *Lr* method). In the *Lr* method, an indicator is first developed, and then a regression line is constructed to relate the retrieval accuracy and the indicator by using linear regression techniques. We use these two estimation methods to estimate the retrieval accuracies of both the *W* method and *SW* method. Their accuracies and processing costs are compared. Since there are many similarities between the *W* method and *SW* method, the explanations are mainly based on the *W* method, followed by the features corresponding to the *SW* method in parentheses.

3.1 Estimations by the document movement method

As the document weights are sorted during the ranking process, the document identifiers with their weights change their positions in the document-weight array, called *doc_wt* hereafter. To study the behavior of document movements, the top T positions of the array are called the *Candidate Region*, and the documents in this region are called *candidate documents*. Fig. 7 illustrates the document movements of the 10 documents with highest scores. In this example, there are four query terms and assume only the top 5 documents are

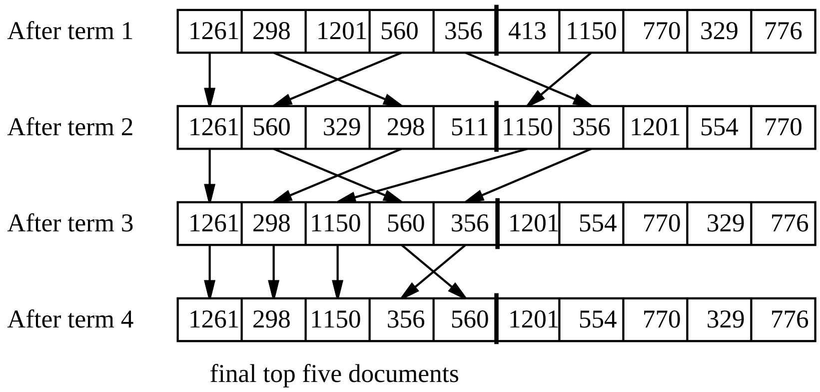

Figure 7: The movement of documents in a ranking process.

returned to the user (i.e., $T = 5$). The arrows indicate the movements of 5 documents which are eventually in Q_f. Documents 1261, 298 and 560 stay in the Candidate Region during the whole retrieval process. Document 356 stays in the Candidate Region after term 1 is processed, but moves out of the Region after term 2 is retrieved. Eventually it gets back to the Candidate Region. Document 1150 was outside the top 5 after term 1 is processed, but it gets into the Candidate Region after term 2 is processed. Even though the document movements seem to be complex and random, with the use of the W method and SW method, some documents remain in the high ranks consistently, while some move up and/or down in the array.

To study the temporal behavior inside the Candidate Region, Fig. 8 records the document movement based on a 200-term query on the MED collection, with the use of the W method. During the whole retrieval process, a total of 47 documents have entered the Candidate Region. To simplify explanations, the documents are assigned pseudo identifiers from 1 to 47 and sorted so that top documents are shown on the right of the figure. In this experiment, the top 20 documents are collected and examined after every five query terms have been processed. As can be observed from Fig. 8, among the top 20 documents obtained at the point when the first 5 postings lists have been processed, only 10 remain after the next five query terms have been processed (documents 1 to 10 shown near the lower left corner of the diagram have been eliminated from and never appeared again in the Candidate Region). We find that some documents stay in the Candidate Region for a longer period of time and some are expelled very soon. To show this behavior pictorially, a continuous vertical line is drawn upwards if a document remains in the Candidate Region at consecutive checkpoints; the continuous vertical line is terminated if the document leaves the Region. The continuous lines at the right part of the graph indicate that many documents stay in the Candidate Region in a stable manner. For instance, if we stop after 100 query terms had been processed, we would have retrieved documents 26-27 and 30-47. That is, 18 out of the final top 20 documents would have been identified, and documents 26 and 27 would be retrieved instead of 28 and 29. Based on the somewhat regular behaviors of the document movements, heuristics can be developed. The following two observations establish the guidelines for our heuristic.

First, those documents which have relatively high scores at the beginning of the retrieval and get into the top t ranks of the Candidate Region, where $t < T$, tend to stay in the Candidate Region for the rest of the processing (the top t ranks of the Candidate Region is called the *Stable Region* and is represented by SR). Therefore, there is a high probability for those documents in SR to be retained in Q_f.

Second, those documents which are eventually in Q_f tend to stay in the Candidate Region for a longer period of time, while those which are not in Q_f tend to move in and out of the Candidate Region more frequently. Thus, the duration for which the documents stay in the Candidate Region can be used to

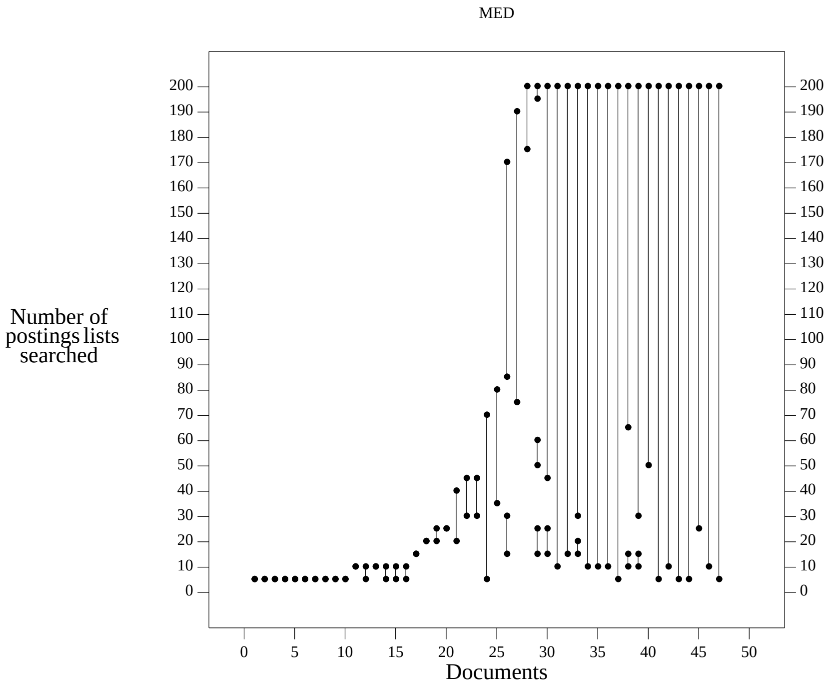

Figure 8: The temporal behavior in the Candidate Region.

predict documents in Q_f. Since postings lists (disk pages) are sorted by decreasing $tf_{max} \times idf$ for the W method ($tf_{max} \times idf \times f(I)$ for the SW method), the influence of the remaining postings lists (disk pages) on the final ranking will decrease as more lists (pages) are processed. Thus, the duration requirement for potential documents to be in Q_f should decrease as the search proceeds. To support this feature, we add to each document a counter which will be increased by one whenever the document moves into the Candidate Region. The document-weight array *doc_wt* is sorted after one disk page of postings is processed. A candidate document is considered as a top document if it stays in the Candidate Region for a certain period of time.

There are 10 checkpoints taken for the whole retrieval process. At each checkpoint, the number of top documents is estimated based upon the following requirements. In our study, we set SR to be 40% of the top T documents, with T equal to 20. The choice of the SR value is experimental. Documents in this sub-region are counted as top documents independently of how long they stay in it. For the first 4 checkpoints (i.e., after 10%, 20%, 30% and 40% of disk accesses), we require that candidate documents to be counted as top documents stay in the Candidate Region 75% of the time. This means that for 100 disk accesses, after 30% of disk accesses are processed, the counted documents must stay in the Candidate Region for at least 22 times. This requirement is decreased as more lists are processed to reflect the decreasing influence of smaller term weights on the final ranking, towards the end of the retrieval process. The requirement is decreased by 5% for every 10% of disk accesses. This means that for 100 disk accesses, at 50% of the disk accesses, the counted documents must stay in the Candidate Region for at least 35 of the times.

The discrepancies for the four test collections based on this estimation method are shown in Figs. 9 and 10. In each diagram, the horizontal axis corresponds to the percentages of the total disk accesses; the vertical axis corresponds to the absolute discrepancy between the actual Ra and the estimated Ra. We use the height of the box to represent the maximum discrepancy at a particular checkpoint, and the horizontal line inside the box to represent the average discrepancy. For each collection, 45 queries are processed. In

Fig. 11, the estimation discrepancies of the W method and SW method based on the Dm method are shown. The results will be discussed after the next method is described.

3.2 Estimations by the linear regression method

The major reason that the W method (SW method) performs better than the L method is that it processes postings lists (disk pages) with heavy weights earlier than those with light weights. Let the total weight of all the postings lists (disk pages) corresponding to a query be w. At a particular checkpoint, if the weights processed thus far is w', it is reasonable to expect that the larger the weight ratio $r = w'/w$, the more it is true that the top documents have been obtained. Initially, $w' = 0$, then $r = 0$ and the corresponding $Ra = 0\%$. When all postings lists (disk pages) are processed, $w' = w$, then $r = 1$ and the corresponding $Ra = 100\%$. The weight ratio r can be expected to be more or less proportional to Ra.

However, the weight ratio r alone cannot provide accurate estimations. The problem can be shown by the different distributions of the partial scores among the candidate documents, with a number of unprocessed postings lists (disk pages). For a given value of w', if most of the weight is concentrated on a few documents, and they are much heavier than the remaining postings lists (disk pages), then they have a high chance to be in Q_f. On the other hand, if the weight is evenly distributed among the documents, and they are not much heavier than the remaining postings lists (disk pages), then they have a low chance to be in Q_f. We find that the differences between the partial scores of the candidate documents and the maximum weights of the unprocessed postings lists (disk pages) are also an important indicator for Ra's. If h candidate documents have weights higher than the maximum weight of the next heaviest postings list (disk page) (i.e., they are heavier than all the remaining postings lists (disk pages)) and h is very close to T, then it is likely that a large portion of the candidate documents will be in Q_f. However, if h is small, then it is likely that very few candidate documents will be in Q_f. Thus, the value of h is proportional to the value of Ra, and $s = h/T$ is called the safe ratio of candidate documents in the Candidate Region.

The weight ratio r and the safe ratio s are two indicators that can estimate the number of top documents obtained at a particular checkpoint. We expect that if $r + s$ is small at a checkpoint, Ra is small; if it is large, Ra is large. However, like the Dm method, the query size can affect Ra at the same time. If there is a small number of postings lists to be processed, the chance for a document to further gain weight is small and the sum of r and s reflects the value of Ra to a greater extent. The situation becomes complex when a large number of postings lists remain to be processed, because there are high chances for a document to gain weight from different postings lists (disk pages). Thus, the size of the query must be taken into consideration as well.

To formulate the indicator, a few more definitions are given below:

1. $w = \sum_{j=1}^{P} max\_wt[j]$, where P is the total number of disk pages required to process all the query terms and $max\_wt[j]$ is the maximum weight of the jth page. It should be noted that the maximum term weight of a page, rather than the sum of term weights in a page, is used in our computation.

2. $w_i = \sum_{j=1}^{\lceil iP/10 \rceil} max\_wt[j]$. The value depends on which sorting method is used, namely the W method or the SW method.

3. $r_i = w_i/w$, the weight ratio.

4. $s_i = h_i/T$, the safe ratio, where h_i is the number of candidate documents whose current weights are larger than the maximum weight of the remaining postings lists (disk pages), after $i\%$ of the disk pages are processed.

Figure 9: The estimation discrepancies of the W method by the Dm method (light gray boxes) and the Lr method (white boxes). Forty-five queries are processed.

Figure 10: The estimation discrepancies of the SW method by the Dm method (dark gray boxes) and the Lr method (gray boxes). Forty-five queries are processed.

Figure 11: The estimation discrepancies of the W method (light gray boxes) and SW method (dark gray boxes) by the Dm method. Forty-five queries are processed.

Figure 12: The estimation discrepancies of the W method (white boxes) and SW method (gray boxes) by the Lr method. Forty-five queries are processed.

After $i\%$ of the disk accesses have been processed and the array *doc_wt* has been sorted, the indicator Ind_i for the W method (SW method) is defined as follows:

$$Ind_i = (r_i + s_i) - comb\_factor(k, i),$$

for $0 \le i \le 10$.

The value of $comb\_factor(k, i)$ is the amount that is to be subtracted from $r + s$ to account for the effect of the combination of term weights from different postings lists to the final ranking. If the query size k is large, a large amount should be subtracted. On the contrary, if the query size is small, a small amount should be subtracted. However, this amount of subtraction is reduced to reflect the decreasing influence of smaller term weights to the final ranking towards the end of retrieval process. Therefore, we use $comb\_factor(k, i) = ck \times (1 - i/10)$, where c is a constant. Since the maximum value of $r + s$ is 2 and the values of $r + s$ are generally below 0.5 at the first few checkpoints, we set $c = 1/500$ to control the amount of subtraction; for instance, the maximum subtraction is 0.2 at the first checkpoint for 100-term queries. At the point when all query terms have been processed, nothing is subtracted, $r + s = 2$ and $Ra = 100\%$. However, it seems to be reasonable to have a bound for subtractions; we require that Ind_i is non-negative. In order to compute the above indicator, in addition to the maximum tf stored in the index file, the sum of the maximum weights of the disk pages in a list, denoted by sum_{max_wt}, is also stored in the index file and is used to compute w above.

To examine the relationship between Ind's and Ra's, the following experiment was conducted. For each collection, 45 queries are run. We take checkpoints at every 10% of disk accesses as before and the Ind's are computed at the same time. In Table 2, the (Ind, Ra) pairs for two queries in CACM by the W method are shown. These (Ind, Ra) pairs can be viewed as points in a two-dimensional plane, with the x-axis for the Ind and y-axis for the Ra. In Figs. 13 and 14, the points corresponding to the W method and SW method are shown in two diagrams for each collection. Since there are 45 queries for each collection, and 11 points for each query (including the 0% and 100%), 495 points in total are plotted in each diagram. The $\odot$'s and $\diamond$'s in Fig. 13 correspond to sample queries 1 and 2 in Table 2. Based upon the distribution of points for each collection, we find that in general the larger the Ind, the larger the Ra. Statistically, we can establish a linear relationship between Ind's and Ra's; regression lines can be drawn by using the linear regression technique for each collection. The regression lines and the corresponding correlation coefficients for CACM, CISI, CRAN and MED are shown in Figs. 13 and 14, respectively. Since the correlation coefficients are about 0.8, 0.9, 0.89, and 0.89 for the W method (0.95, 0.95, 0.97 and 0.95 for the SW method) for CACM, CISI, CRAN and MED, we can conclude that there is a strong correlation between Ind and Ra, especially for the SW method. Once the regression line is drawn, we can estimate the value of Ra when Ind is computed. The discrepancies of this method are shown in Fig. 9 and 10. The estimation discrepancies of the W method and SW method based on the Lr method are shown in Fig. 12.

3.3 The comparison between these two estimation methods

A good estimator should have the following two properties: its estimations are close to the actual values and it is easy to implement. Based upon these criteria, the two estimation methods are compared.

Let us first consider the estimations for the W method. By comparing the heights of the light gray boxes with those of the white boxes in Fig. 9, except for 3 cases in CACM after the 60% point and one case in CRAN at the 50% point, the Lr method performs better than the Dm method in terms of maximum discrepancies. As far as the average discrepancies are concerned, the Lr method also performs better than the Dm method in most cases, except for 4 cases in CACM and one case in CRAN. The average discrepancies in either estimation method are mostly in the 10-20% range. In general, both the maximum and average discrepancies decrease

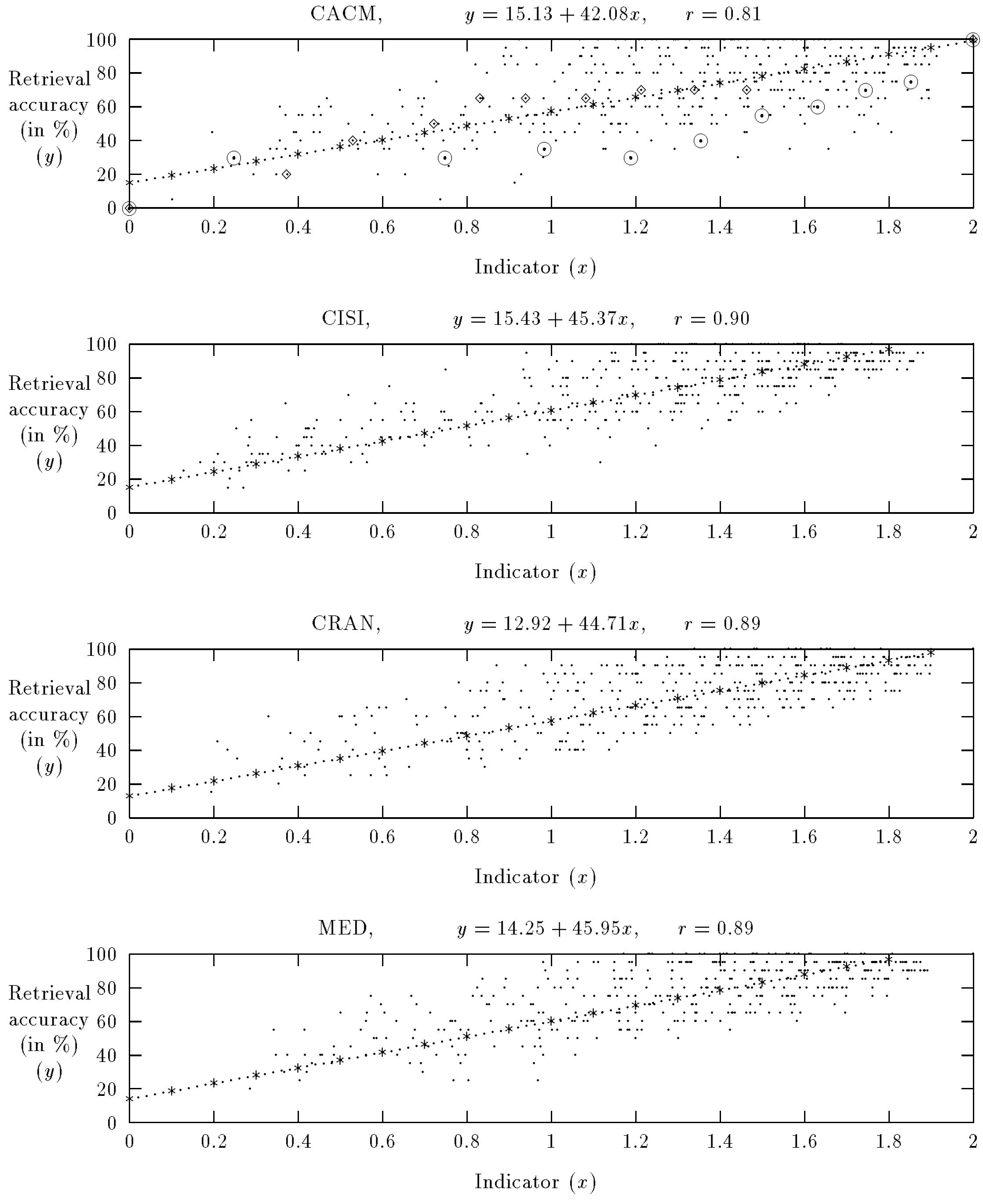

Figure 13: The regression lines for the test collections based on the W method. The horizontal axis corresponds to the indicator; the vertical axis corresponds to the retrieval accuracy. The regression line and the correlation coefficient (r) are given on top of each diagram.

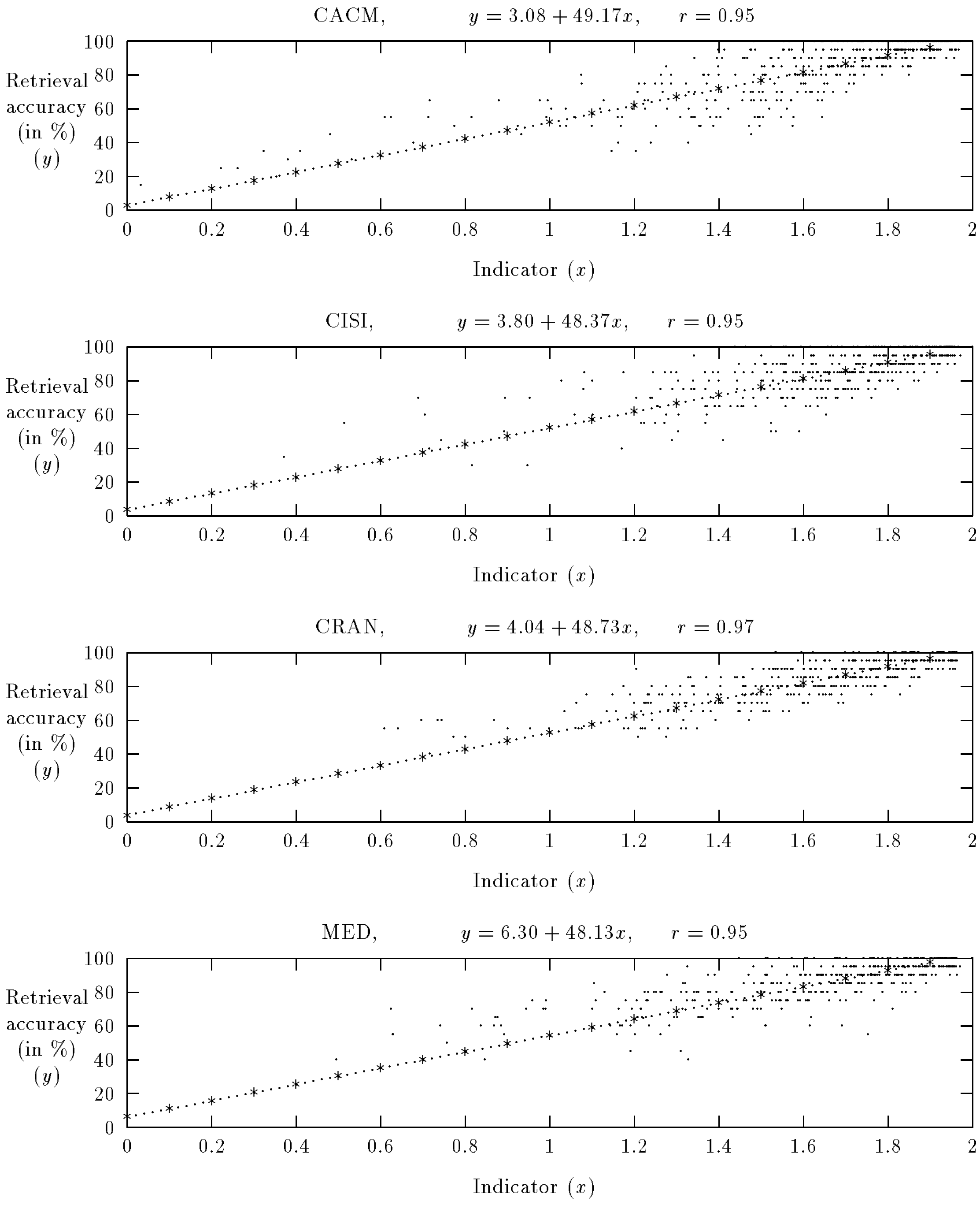

Figure 14: The regression lines for the test collections based on the *SW* method. The horizontal axis corresponds to the indicator; the vertical axis corresponds to the retrieval accuracy. The regression line and the correlation coefficient (r) are given on top of each diagram.

	Sample Query 1		Sample Query 2	
Checkpoints	*Ind*	Actual *Ra* (in %)	*Ind*	Actual *Ra*(in %)
0	0.00	0.00	0.00	0.00
1	0.25	30.00	0.37	20.00
2	0.75	30.00	0.53	40.00
3	0.98	35.00	0.72	50.00
4	1.19	30.00	0.83	65.00
5	1.35	40.00	0.94	65.00
6	1.50	55.00	1.08	65.00
7	1.63	60.00	1.21	70.00
8	1.75	70.00	1.34	70.00
9	1.85	75.00	1.46	70.00
10	2.00	100.00	2.00	100.00

Table 2: The (*Ind*, *Ra*) pairs for two sample queries in CACM by the *W* method.

as the search proceeds. Now, consider the estimations for the *SW* method. By comparing the heights of the dark gray boxes with those of the gray boxes in Fig. 10, the *Lr* method consistently performs better than the *Dm* method at all points in terms of maximum and average discrepancies. The average discrepancies by the *Dm* method are mostly in the 10-20% range, while they are most in the 5-10% range by the *Lr* method. Based upon these comparisons, the estimation ability of the *Lr* method is better than that of the *Dm* method.

To find out which sorting algorithm is better estimated by the estimation methods, the maximum and average discrepancies of the *W* method and *SW* method for each estimation heuristic are compared in Figs. 11 and 12, respectively. For the *Dm* method, except for 1 case in CACM, 1 case in CISI and 1 case in MED, the maximum discrepancies for the *SW* method are smaller. Except for 4 cases in CACM, the average discrepancies for the *SW* method are smaller as well. As far as the *Lr* method is concerned, the estimations for the *SW* method are better than those of the *W* method for almost all of the cases, in terms of maximum and average discrepancies. Moreover, the discrepancies of the *SW* method are much smaller than those of the *W* method. Based upon these results, the estimation methods can approximate the *SW* method with higher accuracies. For practical purposes, both estimation methods achieve satisfactory results on the average.

Let us now turn to the efficiency of the methods. The algorithm of the *Dm* method seems to be easy to implement. It involves the counting of the durations of documents in the Candidate Region and the sorting of the document-weight array after each disk page is processed. Based upon the description above, the *Lr* method seems to be more complex, because it involves the construction of an indicator and the techniques of linear regression to generate a regression line for each document base. However, in operating environments, the *Lr* method performs better than the *Dm* method. The process to generate a regression line is done once for each document collection (or after the document collection has been extensively updated) and thus it is not costly. To process a query, the document-weight array are sorted only 10 times at 10 checkpoints, and the estimation of *Ra*'s can be done rather inexpensively by mapping the value of *Ind* on the regression line. However, the *Dm* method has to sort the document-weight array after a disk page is processed. If there are 300 disk pages retrieved, it will be sorted 300 times, compared to 10 times required by the *Lr* method. One immediate suggestion for the *Dm* method is to sort *doc_wt* less frequently. However, the discrepancies of estimations could get worse when the behavior of document movements in the Candidate Region is not monitored closely.

According to the average discrepancies and processing costs of these two estimation methods, they are both acceptable in practical environments. Based upon the above comparisons, the *Lr* method is more desirable in general, but the *Dm* method is more suitable for document bases which are updated frequently. It should be emphasized that these two estimation methods are not used to improve the retrieval accuracies, they are used to estimate retrieval accuracies.

4 Conclusion

In this paper, we first describe three simple heuristic methods for implementing the $tf \times idf$ weighting strategy, namely the *L* method proposed by Smeaton and van Rijsbergen (Smeaton & Rijsbergen, 1981), the *W* method and the *SW* method proposed by the authors (Lee & Wong, 1991; Wong & Lee, 1991). The basic idea of the optimization methods is that the majority of the top documents can be identified from partial document scores without processing all the query terms.

We introduce the notion of retrieval accuracy, which is used to indicate how fast the top documents are retrieved by different retrieval algorithms. The experimental results show that the *SW* method performs better than the *W* method, which in turn performs better than the *L* method, in terms of the number of disk accesses for partial document ranking. As seen in Figs. 2-5, the increments of *Ra*'s by the *W* method and *SW* method are fast for the first 40% of disk accesses; the average retrieval accuracies are in the range of 70-80%. However, the increments of *Ra*'s slow down as *Ra* approaches 100%; the cost effectiveness of this portion of I/O operations is low.

To realize the advantage of implementing partial document ranking, the estimation of the retrieval accuracies at different points of the retrieval is needed. We proposed and studied two heuristic methods. The first one is called the document movement method. It is based upon the document movements in the document-weight array. The second one is called the linear regression method. It develops a linear relationship between an indicator and the retrieval accuracy so that given an indicator value, the retrieval accuracy can be predicted. According to the amount of discrepancy and the processing costs, the linear regression method performs better than the document movement method. Between the *W* method and *SW* method, both estimation methods do a better job in the *SW* method. The reason that the linear regression method performs better is that it takes different characteristics of a document base into account, while the approach by the *Dm* method is rather simplistic. Even though document bases roughly follow the Zipf's law, their characteristics actually differ from one document base to another. A method taking document characteristics into account, such as the linear regression method, can help narrow down the discrepancy range.

It is true that the *SW* method takes more time to do updates and more space to store the maximum tf's in the index file, but the system performance will not be significantly degraded in an environment where updates are few, or when updates are done in batch. Moreover, the additional storage overhead in the index file introduced by the *SW* method should not be significant since most of the postings lists are short according to the Zipf's law. Based upon the retrieval accuracies and discrepancies of the estimations, the *SW* method seems to be a better candidate for partial document ranking using inverted files.

The major contribution of this paper is to propose and evaluate two search methods so that a large number of top documents can be retrieved with only a small number of disk accesses. Further, two estimation methods have been proposed and evaluated so that the retrieval process can be stopped when the retrieval accuracy reaches the desirable level. We are investigating the mathematical model of partial ranking so as to establish a theoretical background for the heuristics. We are also using meaningful queries to measure the effect of partial ranking on the actual recall and precision of the queries.

References

C. Buckley, & A. F. Lewit (June 5-7, 1985). Optimization of inverted vector searches. Proceedings of the 8th International ACM SIGIR Conference, Montreal, Canada, 97–110.

W. S. Cooper (1988). Getting beyond Boole. *Information Processing and Management*, *24*(3), 243–248.

W. B. Croft, & P. Savino (January 1988). Implementing ranking strategies using text signatures. *ACM Trans. Office Info. Syst.*, *6*(1), 42–62.

K. Fukunaga, & P. M. Narendra (July 1975). A branch and bound algorithms for computing k-nearest neighbors. *IEEE Trans. on Computers*, *C-24*(7), 750–753.

D. L. Lee, & W. Y. P. Wong (May 27-29, 1991). Partial document ranking by heuristic methods. Proceedings of The International Conference on Computing and Information, Ottawa, Canada, 231–239.

D. Lucarella (1988). A document retrieval system based on nearest neighbour searching. *Journal of Info. Science*, *14*, 25–33.

K. C. Mohan, & P. Willett (1985). Nearest neighbor searching in serial files using text signatures. *Journal of Info. Science*, *11*, 31–39.

F. Murtagh (1982). A very fast, exact nearest neighbour algorithm for use in information retrieval. *Information Technology: Research and Development*, *1*, 275–283.

S. A. Perry, & P. Willett (1983). A review of the use of inverted files for best match searching in information retrieval systems. *Journal of Info. Science*, *6*, 59–66.

T. Radecki (1988). Trends in research on information retrieval-the potential for improvements in conventional boolean retrieval systems. *Information Processing and Management*, *24*(3), 219–227.

G. Salton (1968). *Automatic Information Organization and Retrieval.* New York, McGraw-Hill Book Co..

G. Salton (1989). *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Reading, MA, Addison-Wesley.

G. Salton (July 1986). Another look at automatic text-retrieval systems. *Commun. ACM*, *29*(7), 648–656.

G. Salton, E. A. Fox and H. Wu (December 1983). Extended boolean information retrieval. *Commun. ACM*, *26*(12), 1022–1036.

G. Salton, & M. J. McGill (1983). *Introduction to Modern Information Retrieval.* New York, McGraw-Hill Book Co..

D. Shasha, & T. Wang (January 1990). New techniques for best-match retrieval. *ACM Trans. Office Info. Syst.*, *8*(2), 140–158.

A. F. Smeaton, & C. J. van Rijsbergen (May 31-June 2, 1981). The nearest neighbour problem in information retrieval. An algorithm using upperbounds. Proceedings of the 4th International ACM SIGIR Conference, Oakland, California, 83–87.

C. Stanfill, & B. Kahle (December 1986). Parallel free-text search on the Connection Machine system. *Commun. ACM*, *29*(12), 1229–1239.

C. Stanfill, R. Thau and D. Waltz (June 24-28, 1989). A parallel indexed algorithm for information retrieval. Proceedings of the ACM SIGIR Conference, Cambridge, MA, 88–97.

S. F. Weiss (1981). A probabilistic algorithm for nearest neighbour searching, in *Information Retrieval Research*, R. N. Oddy, S. E. Robertson, C. J. van Rijsbergen and P. W. Williams, Eds. London, UK: Butterworths, 325–333.

W. Y. P. Wong, & D. L. Lee (1990). Signature file methods for implementing a ranking strategy. *Information Processing and Management*, *26*(5), 641–653.

W. Y. P. Wong, & D. L. Lee (April 2-4, 1991). A heuristic method for document ranking. Proceedings of The Second International Symposium on Database Systems for Advanced Applications, Tokyo, Japan.

G. K. Zipf (1949). *Human Behavior and the Principle of Least Effort.* Reading, MA, Addison Wesley Publishing.