1. Exact and Approximate Retrieval Status Values

$$\begin{aligned} \text{“}exact\text{”} : RSV(q,d_j) &:= \frac{1}{|\vec{d_j}|} * \sum_{\varphi_i \in q : \varphi_i \in d_j} a_{ij} * b_i \\ \text{“}approx\text{”} : RSV^0(q,d_j) &:= \frac{1}{|\vec{d_j}|} * \sum_{\varphi_i \in q : \sigma(d_j)[h(\varphi_i)]=1} a^0_{ij} * b_i \end{aligned}$$

2. Document Length

$$|\vec{d_j}| := \sqrt{\sum_{\varphi_i \in d_j} {a_{ij}}^2}$$

3. Weights of Query Features

$$\begin{aligned} \text{“}ntn\text{”} : b_i &:= ff(\varphi_i, q) * nidf(\varphi_i) \\ \text{“}ltn\text{”} : b_i &:= (1 + log(ff(\varphi_i, q))) * nidf(\varphi_i) \end{aligned}$$

4. Exact and Approximate Weights of Document Features

$$\begin{aligned} \text{“}ntc\text{”} : a_{ij} &:= \frac{ff(\varphi_i, d_j)}{\max\{ff(\varphi_h, d_j) \mid \varphi_h \in d_j\}} * nidf(\varphi_i) \\ a^0_{ij} &:= 1 * nidf(\varphi_i) \\ \text{“}lnc\text{”} : a_{ij} &:= \frac{(1 + log(ff(\varphi_i, d_j)))}{(1 + log(\max\{ff(\varphi_h, d_j) \mid \varphi_h \in d_j\}))} \\ a^0_{ij} &:= 1 \\ \text{“}ltc\text{”} : a_{ij} &:= \frac{(1 + log(ff(\varphi_i, d_j)))}{(1 + log(\max\{ff(\varphi_h, d_j) \mid \varphi_h \in d_j\}))} * nidf(\varphi_i) \\ a^0_{ij} &:= 1 * nidf(\varphi_i) \end{aligned}$$

5. Normalized Inverse Document Frequency

$$nidf(\varphi_i) := 1 - \frac{\log(1 + df(\varphi_i))}{\log(1 + n)}$$

Table 1: Classification of weighting schemes.

itors, *ACM SIGIR Conference on R&D in Information Retrieval*, pages 1–10, 1992.

[5] D. Harman, E. Fox, R. A. Baeza-Yates, and W. Lee. Inverted Files. In W. B. Frakes and R. A. Baeza-Yates, editors, *Information Retrieval, Data Structures & Algorithms*, pages 28–43. Prentice-Hall, Englewood Cliffs, NJ, 1992.

[6] M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.

[7] J. A. Rice. *Mathematical Statistics and Data Analysis*. Wadsworth & Brooks, Pacific Grove, CA, 1988.

[8] G. Salton and C. Buckley. Parallel Text Search Methods. *Communications of the ACM*, 31(2):202–215, 1988.

[9] G. Salton and C. Buckley. Improving Retrieval Performance by Relevance Feedback. *Journal of the ASIS*, 41(4):288–297, 1990.

[10] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[11] P. Schäuble. SPIDER: A Multiuser Information Retrieval System for Semistructured and Dynamic Data. In *ACM SIGIR Conference on R&D in Information Retrieval*, pages 318–327, 1993.

[12] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, second edition, 1979.

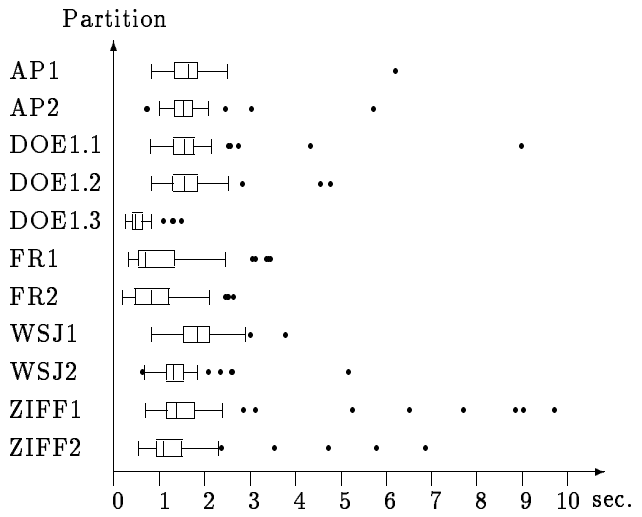


Figure 4: Response times of the first ranked document per run (adhoc queries versus the listed partitions) for the fastest method (M8 nltc.ntc.df15).

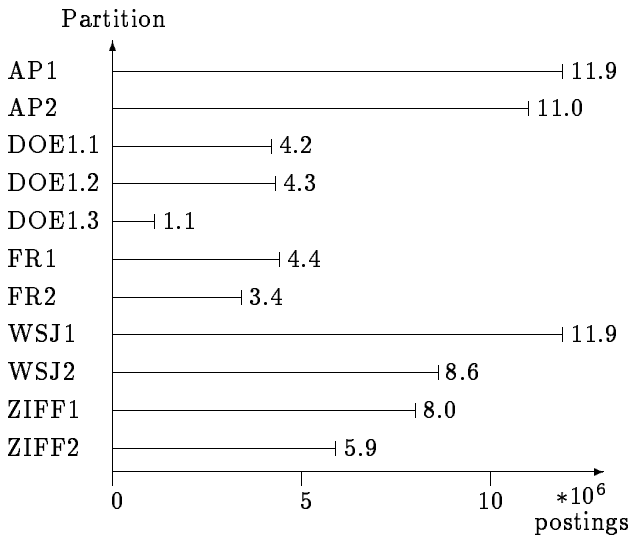


Figure 5: Number of postings per partition.

weighting because of the scaling with the $nidf(\varphi_i)$. On the other hand, the approximation error for the "ntc" weighting is very large compared to both the "ltc" and the "lnc" weighting because of the normalized linear weighting instead of the normalized logarithmic weighting.

The box plots [7, pp. 336] presented in Figure 4 show the distribution of the response times required to determine the top ranked document. A box plot visualizes the median (the line within the box), half of all samples (the box), and outliers (the dots) of a sample collection. In our case we have 50 samples: the response times of the 50 adhoc queries run against a partition. For most queries the top ranked document was retrieved in less than two seconds. The few outliers were all produced by the same queries (#103, #136, #138, #144). In general, the response times become shorter if a partition contains less documents and if the document descriptions consist of less postings on an average (see Figure 5).

# 5 Conclusions

In our approach we stressed the update efficiency. We have shown that the retrieval effectiveness does not have to be sacrificed to achieve a high update efficiency when coping with highly dynamic document collections. Our approach could probably be further improved by finding a weighting scheme that, on the one hand, achieves a very good retrieval effectiveness and that, on the other hand, can be approximated by frequency independent weights with only little variation from the exact weights. The retrieval efficiency could be improved by better partitioning the document collections according to the lengths of the documents. Our approach seems to be very amenable to parallel processing. We may think of several configurations (partitioning the query, partitioning the document collection, etc.). At this moment, it is not clear which configuration is appropriate for which requirements. Furthermore, we do not know yet how dynamic the document collection must be such that our access structure outperforms inverted files.

# References

[1] C. Buckley, G. Salton, and J. Allan. Automatic Retrieval With Locality Information Using SMART. In *TREC-1 Proceedings*, pages 59–72, 1992.

[2] W. B. Croft and P. Savino. Implementing Ranking Strategies Using Text Signatures. *ACM Transactions on Information Systems*, 6(1):42–62, 1988.

[3] U. Glavitsch and P. Schäuble. A System for Retrieving Speech Documents. In N. Belkin, P. Ingwersen, and A. M. Pejtersen, editors, *ACM SIGIR Conference on R&D in Information Retrieval*, pages 168–176, 1992.

[4] D. Harman. Relevance Feedback Revisited. In N. Belkin, P. Ingwersen, and A. M. Pejtersen, ed-

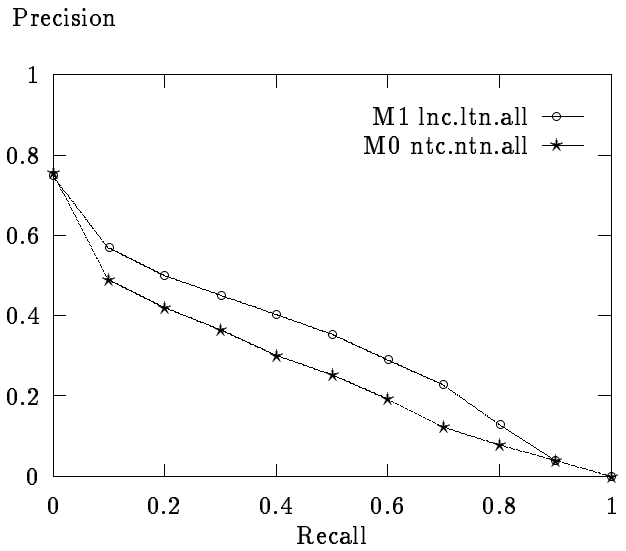


Figure 2: Precision recall graphs of the most effective method (M1) and of the least effective method (M0).

effective than the "ntn" weighting (10-15%). Restricting the vocabulary results in a 2-7% lower precision. We can summarize the experiences as follows:

- The $nidf(\varphi_i)$'s in the document feature weights have a bad influence on the retrieval effectiveness. It could be that for long documents the estimation of the lengths $|\vec{d_j}|$ is inappropriate when the $nidf(\varphi_i)$'s are taken into account.

- Logarithmic feature weighting is more appropriate for the long TREC documents and queries than linear feature weighting. Logarithmic feature weighting avoids an overweighting of features occurring very frequently within a document.

- Restricting the indexing vocabulary by ommitting features with a high document frequency $df(\varphi_i)$ has a noticeable influence on the average precision.

In what follows, we discuss the *influences of the different parameters on the response time* (as shown in Figure 3). Restricting the vocabulary accelerates the query evaluation (by 9-14%) for the reasons described in Section 2. The "ntn" weighting of the query features is also 9-14% faster than the "ltn" weighting. For document feature weighting, the "lnc" weighting is 5-10% slower than the "ltc" weighting. The "ntc" weighting of is even slower. These results can be explained in terms of the approximation error.

- It is obvious that for the "ltc" weighting the approximation error $(a_{ij}^0 - a_{ij}) * b_i$ is smaller than for the "lnc"

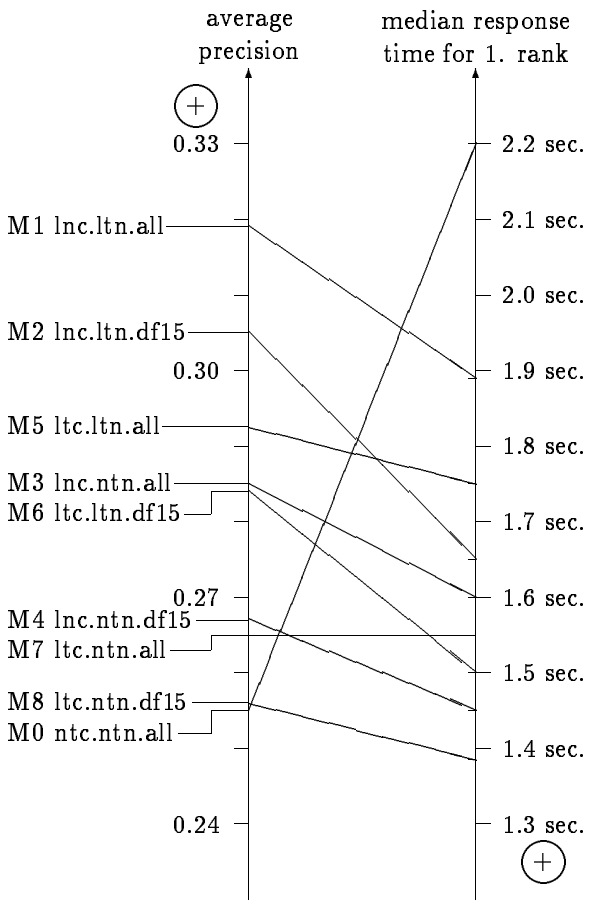


Figure 3: Average precisions and response times of the first ranked document.

# 4 Experiments

In this section, we present the evaluation of the method described above and we compare it to methods to other weighting schemes. We focus on the efficiency of modifying documents and on the correlation between the retrieval efficiency and the retrieval effectiveness. We will also see what is the influence of the vocabulary restriction on the retrieval effectiveness and on the retrieval efficiency. For the final evaluations we concentrated on the adhoc queries.

Before discussing the results, we define what we mean by a *partition*, a *run* and an *experiment*. The document collection has been split up into several *partitions*, each consisting of at most 100,000 documents. Thus, the large collections DOE1 (Department of Energy, Disk 1) and ZIFF3 (Ziff-Davids Publishing, Disk 3) were divided into three and two partitions respectively. A *run* consists of the evaluation of 50 queries (either the set of routing queries or the set of ad hoc queries) against all documents of one partition. For each query, the 1000 top ranked documents have been retrieved. An *experiment* consists of several runs and the merging of the lists of ranked documents for each query. For TREC-2, the two sets of experiments "Topics 51-100 versus Disk 3" and "Topics 101-150 versus Disks 1 and 2" have been evaluated.

All efficiency evaluations are based on CPU time rather than on real time in order to eliminate side effects from other jobs running on the same machine. In these experiments, we used a SUN SPARCserver MP690 with 128 MBytes RAM.

We derived the document descriptions directly from the CD's. The indexing process included the elimination of stop words (van Rijsbergen's stop list [12, pp. 18]) and Porter's word reduction algorithm [6]. The normalized inverse document frequencies have been derived from the documents of disks 1 and 2 only. Uncompressing and indexing a single document needs around 100 msec on an average depending on the length of the document. The computation of the inverse document frequencies from the descriptions took about 1.5 hours of CPU time.

The average time for inserting a document description into the access structure is on a scale of 10 msec - again depending on the number of features $\varphi_i$ per document. Inserting a document description into an inverted file would need more time because the postings had to be inserted into the different lists associated to each feature.

The restriction of the vocabulary was accomplished by omitting features occurring in more than 15% of all documents (from the disks 1 and 2), i.e. in more than 111'337 documents. We have chosen 15% of the collection although also a stronger limit of 10% should not affect the retrieval effectiveness [1]. In our experiments we compare the 15% limit ("df15") to a non restricted vocabulary ("all").

We now have three parameters which can be combined to specify eight different retrieval methods. Each method can be identified by a string built from the labels for the document feature weighting, the query feature weighting and the vocabulary restriction:

$$\langle doc_feat_weight \rangle . \langle query_feat_weight \rangle . \langle vocab \rangle$$

In what follows, we present the results of the following nine methods:

M0 ntc.ntn.all

M1 lnc.ltn.all
M2 lnc.ltn.df15
M3 lnc.ntn.all
M4 lnc.ntn.df15

M5 ltc.ltn.all
M6 ltc.ltn.df15
M7 ltc.ntn.all
M8 ltc.ntn.df15

First, we compare the retrieval effectiveness of our method (M1) described in Section 2 to the standard $tf * idf$ method (M0) by means of the precision-recall graph in Figure 2. As expected, the method M1 is more effective than M0 and achieves a retrieval effectiveness among the best methods presented at TREC-2. In order to find out the reason for this difference in the retrieval effectiveness we must have a closer look at the influences of each parameter (document and query feature weighting).

In Figure 3, the 11-pts average precisions of each method (M0 to M8) are plotted on the left axis, and they are connected to the median response times (for the top ranked document) plotted on the right axis. The most obvious conclusion from this graph is the following: the higher the precision, the slower the response, and vice versa. The method M0 performs clearly worse than the methods M1 to M8 in respect to both retrieval effectiveness and retrieval efficiency.

We concentrate on the response times of the top ranked document because the response times of all further ranked documents are of secondary interest, since a user is supposed to read the top ranked document before looking at the other documents and the retrieval system can retrieve further documents while the user is reading the top ranked document.

We can also see from Figure 3, what are the *influences of the different parameters on the average precision*. Regarding the weighting of the document features, the "lnc" weighting achieves a 4-10% higher precision than the "ltc" weighting. The "ntc" looses 5% of precision compared to "ltc". In the case of query feature weighting, again the logarithmic "ltn" weighting is more

exact retrieval status values are computed as necessary. Assume that $d_7$ is the first (i.e. $d_7$ has the highest $RSV^0$), $d_{13}$ is the second, and $d_5$ is the third document as shown in Figure 1. In this case, $RSV(q, d_7)$ is computed first and next, $RSV(q, d_{13})$ is computed. According to Figure 1, $RSV(q, d_{13})$ is the highest exact retrieval status value of those two exact retrieval status values that are known at this moment. Furthermore, all unknown exact retrieval status values are less than or equal to $RSV^0(q, d_5)$ because of (3). Since $RSV(q, d_{13}) > RSV^0(q, d_5)$, we can conclude that $d_{13}$ has the highest exact retrieval status value of *all* documents. In this example, only two exact retrieval status values had to be computed to determine the top ranked document. In practice, more exact retrieval status values have to be computed.

The evaluation algorithm is based on an *access structure* that specifies the following functions.

$$\sigma : d_j \mapsto \sigma(d_j) \tag{4}$$

$$\pi : d_j \mapsto \{(\varphi_i, ff(\varphi_i, d_j)) : \varphi_i \in d_j\} \tag{5}$$

$$nidf : \varphi_i \mapsto nidf(\varphi_i) \tag{6}$$

$$size : d_j \mapsto |\vec{d_j}| \tag{7}$$

The functions $\sigma$ and $\pi$ determine the signatures and the non-inverted document descriptions respectively. The functions *nidf* and *size* provide scaling and normalization factors. The function values $\sigma(d_j)$ and $\pi(d_j)$ are determined completely by the document $d_j$ itself. The function values $nidf(\varphi_i)$ and $size(d_j)$, however, depend on the entire document collection. Fortunately, the variation of $nidf(\varphi_i)$ is small if the data collection is sufficiently large. Thus, $nidf(\varphi_i)$ has to be recomputed only if the domain of the data collection is shifting and $size(d_j)$ has to be recomputed if either *nidf* was recalculated or $d_j$ was modified. In the next section, we will see how this basic query evaluation scheme can be refined.

# 3 Refinements

The basic evaluation algorithm described in Section 2 achieves an excellent update efficiency because the description of a single document is stored in a compact form which can be updated efficiently. However, the retrieval effectiveness as well as the retrieval efficiency need further refinements in the case of the TREC experiment where extremely large queries are matched against a large document collection containing some large documents.

To *improve the retrieval effectiveness*, we adapted the basic evaluation algorithm to the best global weighting scheme achieved by the SMART system at TREC-1 [1]. The query features are "ltn" weighted, i.e. the query weights depend on the logarithm of the feature frequencies and on the inverse document frequencies, and they are not cosine normalized. The "lnc" weighted document features depend on the logarithm of the feature frequencies. They are cosine normalized, but they do not have a document frequency component. The detailed definitions of the feature weights are given in Table 1.

In order to *achieve a good retrieval efficiency* for the TREC collection we have reduced the indexing vocabulary in such a way that on one hand the retrieval efficiency is improved and on the other hand, the retrieval effectiveness is not deteriorated very much. Our indexing vocabulary was not only reduced by applying a stop word list and a word reduction algorithm, but also by eliminating further indexing terms. From another project [3] and from the experiences of the SMART system at TREC-1 [1] we knew that the removal of the indexing features having a very high document frequency does not deteriorate the retrieval effectiveness very much. The reasons why this restriction has a small effect on the retrieval effectiveness is the following. The elimination of the high document frequency features changes the retrieval status values very little because these features have a small inverse document frequency and hence, they contribute little to the retrieval status values. Such an elimination of the high document frequency features is similar to the application of a collection dependent stop word list in addition to a general stop word list. Thus, this restriction has only a small effect on the retrieval status values.

In what follows, we focus on the retrieval efficiency and why it is improved by this restriction. With a reduced indexing vocabulary,

1. fewer documents have a positive approximate retrieval status value,

2. fewer features $\varphi_i$ contribute an error $(a_{ij}^0 - a_{ij}) * b_i$ to the approximate retrieval status values, and

3. fewer false matches (caused by collisions) occur when computing the approximate retrieval status values.

Both scanning the signatures and sorting the approximate retrieval status values become faster when fewer approximate retrieval status values are positive. Furthermore, the approximate retrieval status values become tighter upper bounds when fewer false matches occur and when the sum of the errors $(a_{ij}^0 - a_{ij}) * b_i$ is reduced. Having tighter upper bounds means that fewer exact retrieval status values have to be computed until the top ranked document is determined. The retrieval effectiveness and the retrieval efficiency achieved by means of the reduced indexing vocabulary are presented in the next section.

imate retrieval status values provide fairly tight *upper bounds* for the exact retrieval status values. We will see how we can take advantage of these upper bounds such that only a few exact retrieval status values have to be computed.

The retrieval method determining the approximate retrieval status values and the retrieval method determining the exact retrieval status values are given by the functions $RSV^0$ and $RSV$ respectively. Let

$$\Phi \quad := \quad \{\varphi_0, \ldots, \varphi_{m-1}\}$$

be the indexing vocabulary (e.g. a set of terms) and let

$$D \quad := \quad \{d_0, \ldots, d_{n-1}\}$$

be the current document collection. We assume that the signatures $\sigma(d_j)$ consist of $w$ bits where the bit at position $p$ has the value $\sigma(d_j)[p]$.

$$\sigma(d_j) \quad = \quad (\sigma(d_j)[0], \ldots, \sigma(d_j)[w-1])$$

Every indexing feature $\varphi_i$ is assigned a bit position $p = h(\varphi_i)$ by means of a hash function $h(\varphi)$.

$$h : \Phi \rightarrow \{0, \ldots, w-1\}$$

The function $h$ specifies a signature $\sigma(d_j)$ for every document $d_j$ by setting the bit at position $p$ iff $d_j$ contains a feature $\varphi_i$ which is hashed to this position.

$$\sigma(d_j)[p] \quad := \quad \begin{cases} 1 & \text{if } \exists \varphi_i \in d_j : h(\varphi_i) = p \\ 0 & \text{otherwise} \end{cases}$$

We now define the approximate retrieval status value $RSV^0(q, d_j)$ and the exact retrieval status value $RSV(q, d_j)$ as follows.

$$RSV^0(q, d_j) := \frac{1}{|\vec{d_j}|} * \sum_{\varphi_i \in q : \sigma(d_j)[h(\varphi_i)]=1} a^0_{ij} * b_i \qquad (1)$$

$$RSV(q, d_j) := \frac{1}{|\vec{d_j}|} * \sum_{\varphi_i \in q : \varphi_i \in d_j} a_{ij} * b_i \qquad (2)$$

where $a^0_{ij}$ denote the approximate weights of document features, $a_{ij}$ denote the exact document weights, and $b_i$ denote the query weights. The size $|\vec{d_j}|$ of the description vector $\vec{d_j}$ is defined as usual (see Table 1).

Our basic approach consists of a $tf * idf$ weighting scheme, i.e. the query features are "ntn" weighted whereas the document features are "ntc" weighted (see Table 1).

The query weights $b_i$ depend on the feature frequencies $ff(\varphi_i, q)$ and on the normalized inverse document frequencies $nidf(\varphi_i)$. The exact document weights $a_{ij}$ depend on the feature frequencies $ff(\varphi_i, d_j)$ and on the $nidf(\varphi_i)$. In addition, they $(a_{ij})$ are cosine normalized, which is expressed by the division by $|\vec{d_j}|$ in (1) and (2). The $nidf(\varphi_i)$'s are determined by the document frequencies $df(\varphi_i)$. The normalizations of the feature frequency component and the normalizations of the document frequency component do not affect the retrieval status values because they cancel out when using the cosine measure.

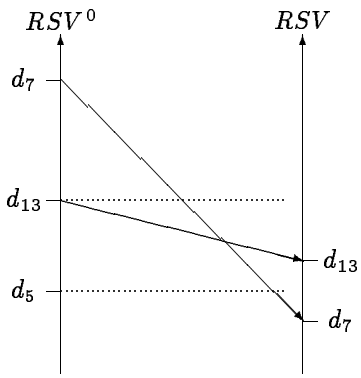


Figure 1: Computation of the exact retrieval status values in decreasing order of the approximate retrieval status values.

Because of these normalizations, the approximate document weights $a^0_{ij}$ are always upper bounds of the exact document weights $a_{ij}$, but they do not depend on the feature frequencies $ff(\varphi_i, d_j)$.

$$0 \leq a_{ij} \leq a^0_{ij}$$
$$0 \leq b_i$$

It is easy to show that $RSV(q, d_j) = 0$ if $\sigma(q) \wedge \sigma(d_j) = 0$. Furthermore, from the fact that $\varphi_i \in d_j$ implies $\sigma(d_j)[h(\varphi_i)] = 1$ and from $a_{ij} \leq a^0_{ij}$ follows

$$RSV(q, d_j) \quad \leq \quad RSV^0(q, d_j) \qquad (3)$$

In what follows, we describe the evaluation algorithm by means of an example. Let $q$ be the user's query. In a first step, bitwise AND-operations are performed with the query signature $\sigma(q)$ and the signature $\sigma(d_j)$ of every document $d_j \in D$. If $\sigma(q) \wedge \sigma(d_j)$ is non-zero then the approximate retrieval status value $RSV^0(q, d_j)$ is computed. The first step is finished by ordering the documents in decreasing order of the approximate retrieval status values. Remember that $\sigma(q) \wedge \sigma(d_j) = 0$ implies that $RSV(q, d_j) = 0$ and hence, $d_j$ can be ignored in the sequel.

In the second step, the exact retrieval status values are computed in the order that was previously determined by means of the approximate retrieval status values. As shown in the following example, only as many

# Effective and Efficient Retrieval from Large and Dynamic Document Collections

Daniel Knaus, Peter Schäuble
(knaus|schauble)@inf.ethz.ch
Swiss Federal Institute of Technology (ETH)
CH-8092 Zürich, Switzerland

## Abstract

A new retrieval method together with a new access structure is presented that is aimed at a high update efficiency, a high retrieval efficiency and a high retrieval effectiveness. The access structure consists of signatures and non-inverted descriptions. This access structure can be updated efficiently because the description of a single document is stored in a compact form. The signatures are used to compute approximate retrieval status values first, and the non-inverted descriptions are then used to determine the final list of documents ranked by the exact retrieval status values. Our basic approach based on the standard $tf * idf$ weighting scheme has been improved in in both retrieval effectiveness and retrieval efficiency. On an average, the time for retrieving the top ranked document is clearly below two seconds while the document collection can be updated in 10 msec. (inserting, deleting, or modifying a document description).

## 1 Introduction

Information retrieval systems are more and more used to retrieve information from *large and dynamic* data collections, e.g. in office automation or in newscasting companies. Since in such environments, the data collections may have to be updated many times every second, the update efficiency is an important evaluation criterion that should be taken into account in addition to the traditional evaluation criteria (retrieval effectiveness, retrieval efficiency, etc.) [10, pp. 161], [12, pp. 144].

Before introducing our approach, we show that the update efficiency is conflicting with the retrieval effectiveness and with the retrieval efficiency. In the case of static data collections, the retrieval effectiveness criterion is usually met by means of *weighted retrieval* including relevance feedback [4], [9] whereas the retrieval efficiency criterion is met by precomputing the document descriptions and by organizing these descriptions as an inverted file [5]. In the case of dynamic data collections, however, a transaction updating the inverted file may block other retrieval and update transactions for an unacceptable long time because adding the description of a new document to an inverted file is time consuming, particularly if the document is long. A possible remedy is to use signatures instead of an inverted file [2]. Signatures can be updated efficiently; however, they do not allow document feature weighting and hence, the retrieval effectiveness of the signature based method is inferior to a fully weighted retrieval method [8]. Thus, it is difficult to achieve simultaneously high retrieval effectiveness, high retrieval efficiency, and high update efficiency.

In our approach, we focus on the retrieval from large and dynamic data collections where we face the problem of achieving simultaneously high retrieval effectiveness, high retrieval efficiency, and high update efficiency. Addressing this problem is justified by the need for appropriate retrieval capabilities in dynamic environments such as in office automation or in newscasting companies. Within the SPIDER project [11] we have developed a retrieval method and an access structure which supports fully weighted retrieval and which achieves short response times even when the collection is updated several times every second. In Section 2, we describe our basic approach, in Section 3, we focus on refinements of this basic approach, in Section 4, we present the results, and in Section 5, we draw some conclusions.

## 2 SPIDER's Query Evaluation Algorithm

In this section, we present a retrieval method together with a new access structure facilitating both fast weighted retrieval and efficient updates of the access structure. A preliminary version was presented in [11]. Our access structure consists of *signatures and non-inverted descriptions* of the documents. The signatures are used to compute approximate retrieval status values $RSV^0(q, d_j)$ first and the non-inverted document descriptions are then used to determine the exact retrieval status values $RSV(q, d_j)$. The trick is that the approx-