Document Ranking on Weight-Partitioned Signature Files

Dik Lun Lee and Liming Ren
The Ohio State University

November, 1994

Abstract

A signature file organization, called the weight-partitioned signature file, for supporting document ranking is proposed. It employs multiple signature files, each of which corresponds to one term frequency, to represent terms with different term frequencies. Words with the same term frequency in a document are grouped together and hashed into the signature file corresponding to that term frequency. This eliminates the need to explicitly record the term frequency for each word.

We investigate the effect of false drops on retrieval effectiveness if they are not eliminated in the search process. We have shown that false drops introduce insignificant degradation on precision and recall when the false drop probability is below a certain threshold. This is an important result since false drop elimination could become the bottleneck in systems using fast signature file search techniques.

We perform an analytical study on the performance of the weight-partitioned signature file under different search strategies and configurations. An optimal formula is obtained to determine for a fixed total storage overhead the storage to be allocated to each partition in order to minimize the effect of false drops on document ranks. Experiments were performed using a document collection to support the analytical results.

Categories and Subject Descriptors: H.2.2 **[Database management]:** Physical Design - *access methods*; H.3.3 **[Information Storage and Retrieval]:** Information Search and Retrieval - *retrieval models*; H.3.6 **[Information Storage and Retrieval]:** Library Automation

General Terms: Design, Experimentation, Performance

Additional Keywords and Phrases: Document retrieval, information retrieval, text retrieval, superimposed coding, signature file, access method.

1 Introduction

The signature file approach has been investigated quite extensively in the last decade. It has been applied to a large variety of applications, including textual databases such as news databases [Stanfill and Kahle 1986], multi-attribute retrieval in relational databases [Aho and Ullman 1979;

Part of the research was done while the first author was on sabbatical leave to Online Computer Library Center (OCLC), Dublin, Ohio, from Sept 1992 to May 1993.
Authors' address: Department of Computer and Information Science, The Ohio State University, 2036, Neil Ave., Columbus, Ohio 43210-1277, USA; email: {dlee; ren}@cis.ohio-state.edu; Lee is currently on leave to Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong; email: dlee@cs.ust.hk.

text block	⋯ text ⋯⋯ database ⋯

word signatures:	
text	001 000 110 010
database	000 010 101 001
block signature ($\vee$)	001 010 111 011

Queries	Query Signatures	Results
1) retrieval	010 001 000 011	$\leftarrow$ no match
2) database	000 010 101 001	$\leftarrow$ match
3) information	001 000 111 000	$\leftarrow$ false drop

Figure 1: Signature generation and comparison based on superimposed coding.

Lee and Leng 1990], multimedia office filing [Christodoulakis et al. 1986], and chemical databases for DNA matching [Lipman and Pearson 1985].

Superimposed coding is the most common way for generating signatures from a text. In superimposed coding, a text is divided into text blocks containing the same number of unique, nontrivial words. Each word in a text block is hashed into a word signature. A block signature is generated by superimposing all word signatures generated from the block. In a query, the query terms are hashed and superimposed into a query signature in a similar way. Then the query signature is matched against each signature in the signature file. Figure 1 is an example showing the generation of the block signature from a text block.

The signature file is a filtering mechanism which will eliminate most, but not all, of the text blocks which will not match the query. The first case shown in Fig. 1 illustrates this point. The query signature doesn't "match" with the text signature in that some of the bits in the text signature are zero while the corresponding bits in the query signature are set to one. If the query term "retrieval" is indeed in the text, the query signature would be one of the word signatures forming the text signature; thus, every bit set in the query signature will be set in the text signature. The second case shows a match, where for each bit in the query signature set to one, the corresponding bit in the block signature is also set to one. The third case is a false drop. False drops are text blocks which the signature file identifies as containing the query terms (i.e., a match) but indeed they don't. They can be eliminated by further comparing the query terms with the text blocks.

Compared to inverted files, the signature file approach has two major advantages:

- Its storage overhead can be controlled easily and in general is very low. Therefore, if only a limited space for indexes is available, the signature length can be set so that the whole signature file will fit into the given space.

- Its structure is very simple. Thus, it has low processing overhead on insertion. Since signatures are conceptually organized as a sequential file, it is easy to partition a signature file for parallel processing [Stanfill and Kahle 1986].

The drawbacks of the signature file approach are:

- Its search speed is low compared to inverted files, since the signature file has to be searched exhaustively if signatures are organized in a single sequential file.

- A signature encodes the presence of terms in a text. Therefore, it can easily support the Boolean retrieval model. However, in order to support document ranking, term frequencies must be stored; this is difficult to do on signatures, which are simple bit vectors.

- Signature files introduce false drops which are expensive to eliminate.

Recent research on signature files has addressed the improvement of search speed, as exemplified by the partitioned signature file approach [Lee and Leng 1989; Lee and Leng 1990; Zezula, Rabitti and Tiberio 1991]. This paper addresses the last two drawbacks of signature files listed above. We first present a technique, called the weight-partitioned signature file, for implementing document ranking with signature files. The technique we propose does not represent term frequency values explicitly. Furthermore, it also avoids exhaustive search on the signature file. An analytical study is performed on various performance issues of the weight-partitioned signature file. We also present an extensive evaluation on the degradation of retrieval effectiveness if false drops are not eliminated from the signature file.

Section 2 discusses the problem of false drops in signature files and the motivation of this study. Section 3 defines the term weights and the similarity formula under the vector-space model. The design of the signature file for supporting document ranking is outlined in Section 4. Section 5 presents an analytical study and the experimental results on the performance of the weight-partitioned signature file under different search strategies and storage configurations. Section 6 presents the conclusions and outlines some problems for future research.

Problems of False Drops

False drop is an inherent property of signature file. The false drop probability, defined as the ratio of false drops to the number of unqualified signatures, is dependent on the size of the signatures and thus on the storage overhead. False drop probability can be reduced arbitrarily by using a longer signature length while keeping the number of distinct keywords in a text block the same. However, to eliminate false drops completely, it is necessary to retrieve the text and perform pattern matching between the query term(s) and the text after matching the signature file. This process imposes a great penalty on the speed of signature file. The cost of false drop elimination is particularly acute for large collections since the number of false drops is proportional to the number of signatures and thus to the size of the collection. Furthermore, when queries are not highly selective, a large number of matches will be returned and each of the matches has to be verified with pattern matching. The high cost raises a serious concern that it will become the bottleneck in retrieval, especially in systems employing fast signature search techniques [Lee, Kim and Patel 1994]. In other words, the search speed will be slow no matter how fast the signature file itself can be searched.

This paper investigates the feasibility of retaining the false drops generated by the signature file. For Boolean retrieval, it is problematic since users are accustomed to the semantics of Boolean retrieval that each document retrieved will satisfy the Boolean condition that the user specifies. Compared to the Boolean model, retrieval in statistical models is less predictable in that documents are retrieved based on their relative ranking rather than their absolute scores. Therefore, given a query, it is difficult to tell why a document is retrieved without examining the other documents. This makes the effect of false drops less observable to the user even if the document ranks are slightly perturbed by the false drops. This paper is based upon the conjecture that since document ranking is not a precise process (i.e., only some of the retrieved documents are relevant to the users), the addition of some "noise" or perturbation, due to false drops, to the result document set should have a small impact on retrieval effectiveness. If the conjecture is true, then the signature file approach will become a very attractive technique for implementing document ranking.

2 Term Weights and Similarity Measure

Term weights are determined by the $tf \times idf$ strategy [Salton and Buckley 1988]. (Table 1 summaries the important symbols in this paper.) Precisely, the weight of term j in document i is defined as:

$$w_{i,j} = tf_{i,j} \times idf_j,$$

where $tf_{i,j}$ is the frequency of occurrence of term j in document i and idf_j is the inverse document frequency of term j in the entire collection. The similarity between a document and a query is based on the cosine measure:

$$\Theta(D_i, Q) = \frac{\sum_{j=1}^{V} w_{Q,j} \times w_{i,j}}{\sqrt{\sum_{j=1}^{V} w_{Q,j}^2 \sum_{j=1}^{V} w_{i,j}^2}}, \tag{1}$$

where V is the vocabulary size, $w_{Q,j}$ is the weight of term j in the query, defined in the same way as document term weights.

It is clear from equation (1) that the factor $\sum_{j=1}^{V} w_{Q,j}^2$ in the denominator is a constant for each document, so it only affects the absolute scores but not the ranks of the documents. Therefore, this factor is not used in our study. The factor $\sum_{j=1}^{V} w_{i,j}^2$ is very expensive to compute, because the weight of every term in a document has to be computed, not just for the terms specified in the query. Note that this factor cannot be pre-computed for each document because the weight depends on idf which changes as documents are inserted into and deleted from a collection. Therefore, we approximate the normalization factor with the square root of the number of terms in a document (i.e., the term weights are set to one when the normalization factor is computed). The effect of the approximation has been shown to be negligible compared to the exact normalization factor [Lee and Chuang 1994]. In summary, the similarity is computed as:

$$\Theta(D_i, Q) = \frac{\sum_{j=1}^{V} w_{Q,j} \times w_{i,j}}{\sqrt{d_i}}, \tag{2}$$

where d_i is the number of terms in document D_i.

3 Weight-partitioned Signature File

One problem of signature files is the difficulty in encoding term frequency information. Croft and Savino evaluated several approaches which range from not representing term frequencies at all to approximating term frequencies with the number of text blocks containing the term [Croft and Savino 1988]. None of the approaches allow exact representation of term frequencies. Wong and Lee proposed a method to represent the exact values of term frequencies by using a separate signature file for terms with the same term frequency. It is done by first grouping terms with the same term frequency in a document. Then, signatures are generated from each group and stored in the signature file corresponding to that term frequency [Wong and Lee 1990]. In this paper, we follow a similar approach. The main difference is that our approach uses superimposed coding whereas

C	Total storage constraint.
D	A document.
D_i	i^{th} document in the database.
d	Number of terms in document D.
d_i	Number of terms in document D_i.
I_i	Average number of terms in a document with term frequency i.
$I_i(D)$	Number of terms in D with term frequency i.
idf_j	Inverse document frequency of term j in the database.
l_i	Total number of signatures in SF_i.
$l_i(D)$	Number of signatures in SF_i for document D.
m	Signature length.
m_i	Signature length in SF_i.
$\vec{m}$	Abbreviation for $m_1, \cdots, m_n$.
N	Total number of documents.
n	Number of tf groups.
$p(m, w, s)$	False drop probability given m, w and s.
p_i	False drop probability of a signature of file tf_i.
Q	Query.
q_i	False drop probability of a signature against file tf_i.
s	Number of words hashed into a signature.
s_i	Number of words hashed into a signature in SF_i.
$\vec{s}$	Abbreviation for $s_1, \cdots, s_n$.
SF_i	Signature file corresponding to term frequency i.
$\Theta(D_i, Q)$	Similarity between document D_i and query Q.
$tf_{i,j}$	Term frequency of term j in document i.
V	Size of the document vocabulary.
w	Weight of a word signature (number of bits set to 1).
w_i	Weight of a word signature in tf_i (number of bits set to 1).
$\vec{w}$	Abbreviation for $w_1, \cdots, w_n$.
$w_{i,j}$	Weight of term j in document i.
$w_{Q,j}$	Weight of term j in query Q.
W	General term ranking weight.

Table 1: Major Symbols.

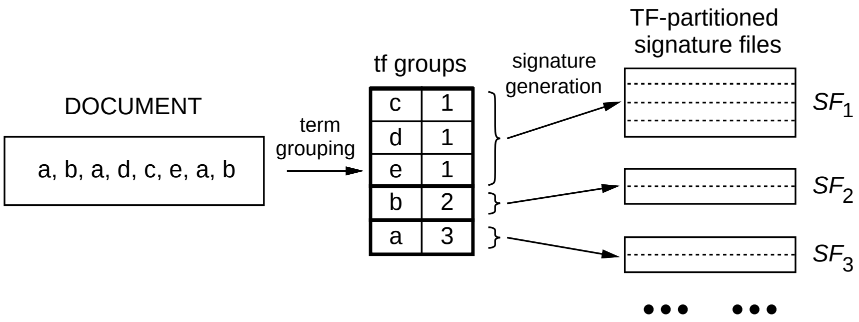

Figure 2: An example of weight-partitioned signature files.

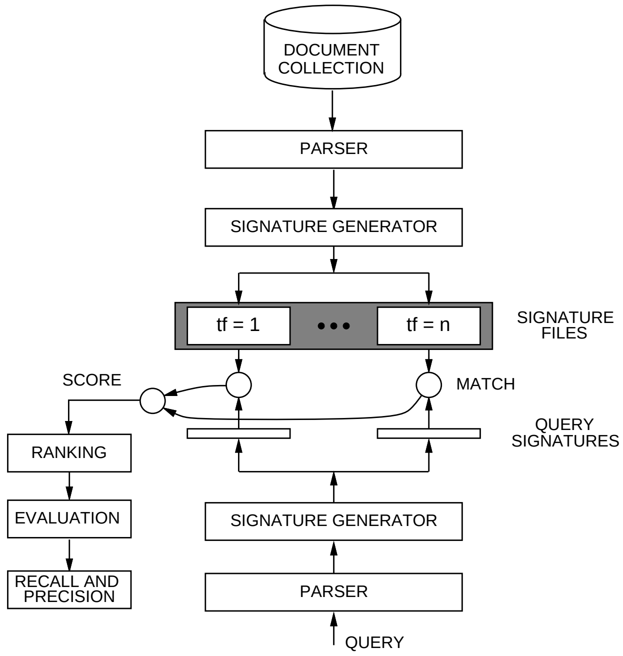

Figure 3: Retrieval system based on weight-partitioned signature files.

Wong and Lee uses an encoding scheme which, unlike superimposed codes, does not introduce any false drops.

Figure 2 is an example of the signature generation process and Figure 3 illustrates the general design of the retrieval system. A document first goes through a preprocessing stage where common words are removed and keywords stemmed. Then, the keywords of a document are grouped according to term frequency. Signatures are generated from keywords in each group as in a traditional signature file, but signatures generated from different groups are stored in different signature files. The group corresponding to term frequency i is denoted by tf_i and the corresponding signature file is denoted by SF_i. During a search, the query signature is compared to each of the SF_is. If a match is found in SF_i, the term frequency of the term producing the match has a term frequency equal to i, which can be used in computing the document score.

The algorithms for generating and searching the signature files are given in Algorithms 1 and 2, where n and N denote, respectively, the number of tf-groups and the number of documents in the collection. They are shown at the conceptual level for the purpose of illustration only, and are not necessarily implemented as described in real implementations.

Algorithm 1: signature file generation

sort keywords and tabulate term frequencies of keywords;
for $i = 1$ to n do
    for every s keywords in tf-group i do
        generate signature and insert into SF_i;
    end {for}
end {for}

Algorithm 2: signature file search

1. for all term j in Q do
2.   obtain *idf* of term j;
3.   generate signature for term j;
4. end {for}
5. for $i = 1$ to N do
6.   for $j = 1$ to n do
7.   for all term k in Q do
8.   if term k's signature matches D_i's signature in the jth partition then
9.   increment D_i's score by i by $j \times idf_k \times w_{Q,k}$
10.   break;
11.   end {if}
12.   end {for}
13.   end {for}
14. end {for}

In the weight-partitioned signature file, multiple occurrences of the same keyword in a document will generate only one word signature and thus are represented in the signature file only once. This is different from the traditional method where a keyword will be represented as many times as the number of text blocks containing it (although multiple occurrences of a keyword within the same text block will be collapsed). This collapsing of multiple occurrences into a single representation in the signature file significantly reduces the storage overhead for long documents; on the other hand, the distribution of a document's signatures into several signature files causes additional overhead. Furthermore, this method does not always search all of the signatures in the signature file; whenever a match is found, the search can proceed to the next document since a query signature will find *at most* one match (ignoring false drops) within a document. This explains the break statement in step 10 of Algorithm 2.

Since the signature file is partitioned, the partitions can be searched in different orders and may have different signature lengths. The effects of different choices are explained in the following two subsections.

3.1 Search Strategies

A search can start from the signature file with highest term frequency and move towards the signature file with the lowest term frequency, or vice versa. We refer to these two approaches, respectively, as HL (high to low) and LH (low to high) methods. There are two contradicting factors affecting the merits of these two search orders, as explained in Figure 4. Suppose signatures

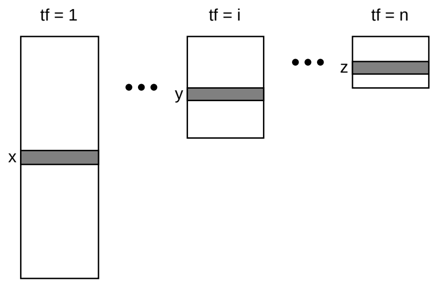

Figure 4: Effects of search strategies.

x and z are false drops matching the query term being searched. If the search starts from the highest term frequency, z will be identified as a match. In this case, the effect of the false drop on the document score will be very high since z has a high term frequency. On the contrary, if the search starts from the lowest term frequency, the false drop x will have a small effect on the document score since the term frequency is low. This suggests that LH is better than HL. However, there is a contradicting factor due to the fact that the number of signatures at high term frequencies is much less than that at low term frequencies, owing to the Zipfian distribution of terms among term frequencies. If we further assume that the probability that a true match occurs at SF_i is the same for all i, then, on the average, the chance of encountering a false drop before a true match is found is smaller in the HL approach than in the LH approach. In Fig. 4, if y is a true match, then there will be much fewer signatures to the "right" than to the "left" of y. Consequently, the probability that the search will arrive at y without hitting any false drops is higher if the search starts from the "right" (i.e., high term frequency). Further, the search speed is improved since once a match is found, the search can proceed immediately to the next query term. This factor is magnified if the user has a tendency to specify query terms which have high term frequencies (i.e., true matches tend to skew towards high term frequencies). This suggests that the search should start from the highest term frequency. The performance of these two search strategies will be studied in Section 4.

3.2 Optimal Signature File Configuration

When the signatures are partitioned among several files, it is possible to use different signature configurations for different partitions. Since a false match at high term frequencies affects the document scores more than those at low term frequencies, it is desirable to reduce the false drop probabilities at high term frequencies. On the other hand, if we set lower false drop probabilities for the signatures with low term frequencies, the absolute number of false drops will be large since there are more signatures at the low frequency end. Therefore, there are conflicting factors affecting the determination of the signature lengths for the partitions. For a given storage overhead (i.e., a fixed total size for all the signature partitions), we want to obtain an optimal way of allocating the storage to the partitions so as to minimize the effect of false drops on precision and recall. This is investigated in Section 4.

4 Analytical Results

In superimposed coding, there are four major parameters related by the following formula [Roberts 1979]:

$$p = (1 - (1 - w/m)^s)^w,$$

where: p: false drop probability,
w: weight of a word signature (number of bits set by a word),
m: signature length,
s: number of words hashed into a block signature.

It has been shown that in order to minimize false drop probability for a given storage overhead, the number of bits set in a block signature must be equal to half of the signature length, and

$$w = (1/\ln 2)^2 s(1 - 2^{-\frac{1}{s}}) \ln(1/p) \quad (3)$$
$$m = (1/\ln 2)^2 s \ln(1/p). \quad (4)$$

Formula (3) can be approximated by:

$$w = (1/\ln 2) \log_e(1/p) \quad (5)$$

when $s >= 5$. These formulas are used later in this paper to determine the values of m, s and w when two of these parameters are given.

To improve clarity, we sometimes use $p(m, w, s)$ instead of p to indicate the relationship between p, w, m and s. With this notation, the false drop probability for signatures in partition SF_i is given by $p(m_i, w_i, s_i)$, or p_i.

In the experiment, p is a variable which is varied for each experiment run. The signature length, m, is a design parameter which is fixed for each test collection. Choosing a length which fits into the native word length of the machine is most efficient, but the native word length is short and thus will limit the achievable minimum false drop probability. Further, a low false drop probability with a short signature length entails an extremely small number of words to be hashed in a signature, resulting in a large number of signatures and disk accesses (if buffering is not used). On the other hand, a long signature length is less efficient to manipulate and will generate under-utilized signatures for short documents.

4.1 HL versus LH search strategies

In this subsection, we analyze the performance of the HL and LH search strategies. The performance of these two strategies depend on the distributions of false and true drops. The distribution of false drops further depends on the signature lengths of the partitions. In order to compare these two methods, we assume in this subsection that the distributions and the signature lengths are uniform.

In the search algorithm, a signature file SF_i is treated as a single unit. A match with any signature in SF_i results in tf equal to i. We need to find the probability of false match when we compare a random signature with all of the signatures in SF_i. We denote the false drop probability against SF_i as q_i to distinguish it from the usual false drop probability, which is the probability for two signatures to match combinatorially.

Lemma 1 *The false drop probability against tf_i is given by:*

$$q_i = 1 - (1 - p(m_i, w_i, s_i))^{l_i} \approx l_i p(m_i, w_i, s_i).$$

where l_i is the number of signatures in SF_i.

Proof: This is clear. □

We may use different m_i, w_i and s_i for different tf-groups. As long as

$$1 \geq p_1 \geq \cdots \geq p_n \geq 0,$$

we have

$$0 \leq 1 - p_1 \leq \cdots \leq 1 - p_n \leq 1.$$

From Zipf's Law, we know $l_1 \geq l_2 \cdots \geq l_{n-1}$ and $l_1 > l_n$. This implies

$$(1 - p_1)^{l_1} \leq \cdots \leq (1 - p_{n-1}))^{l_{n-1}} \quad \text{and} \quad (1 - p_1)^{l_1} \leq (1 - p_n)^{l_n},$$

or, equivalently, $q_1 \geq \cdots \geq q_{n-1}$ and $q_1 \geq q_n$. These relations among q_i's imply that false drop probabilities are higher at the low tf end than at the high tf end. This turns out to be very important in our analysis. One thing we want to point out is that $p_1 \geq \cdots \geq p_n$ does not necessarily mean $m_1 \geq \cdots \geq m_n$ (neither does the other direction hold) since there are other parameters such as s_i's and w_i's involved.

Lemma 2 *Given a random word signature, the expected tf at which a match is found can be obtained by $S_n^{(n)}$ for HL and $T_1^{(n)}$ for LH, where the definitions of $S_i^{(n)}$ and $T_i^{(n)}$ are given as follows:*
(1) For HL:

$$\begin{aligned} S_n^{(i)} &= q_i i + (1 - q_i) S_{i-1}^n \quad \textit{for } i = n, \cdots, 2 \\ S_1^{(n)} &= q_1. \end{aligned}$$

(2) For LH:

$$\begin{aligned} T_i^{(n)} &= i q_1 + (1 - q_i) T_{i+1}^{(n)} \quad \textit{for } i = 1, \cdots, n-1 \\ T_n^{(n)} &= n q_n. \end{aligned}$$

Further, $S_n^{(n)} \geq T_1^{(n)}$.

Proof: In the HL method, the probability that there is a match in tf_n is q_n. If there is a match in tf_n, the expected tf value is n. Otherwise, we have to check the next lower tf group. Therefore, we get the following recurrence relation:

$$S_n^{(n)} = q_n n + (1 - q_n) S_{n-1}^{(n)}.$$

For tf_1, there is no lower tf group, the expected tf value is given by q_1. $T_n^{(n)}$ for the LH method can be derived in a similar way. That $S_n^{(n)} \geq T_1^{(n)}$ is clear from the definitions of $S_n^{(n)}$ and $T_1^{(n)}$. □

Lemma 3 *Suppose a word appears in a document with $tf = j$, then HL yields the following expected tf:*

$$j + ((n-j)q_n + (n-j-1)q_{n-1} + \cdots + q_{j+1})$$

and LH yields:

$$j - ((j-1)q_1 + (j-2)q_2 + \cdots + q_{j-1},$$

when high order terms such as $q_i q_j$ and $q_i q_j q_k$ are dropped.

Proof: We will use the formulas in Lemma 2. The key in this Lemma is that search will stop at tf_k, where $k \geq j$ for HL and $k \leq j$ for LH.

(1) For HL:

$$\begin{aligned} S_n^{(i)} &= q_i i + (1-q_i) S_{i-1}^n \quad \text{for } i = n, \cdots, 2 \\ S_1^{(n)} &= q_1. \end{aligned}$$

By eliminating $S_i^{(n)}$ for $i = n-1, \cdots, j$ from the above equations, we have

$$\begin{aligned} S_n^{(n)} &= q_n n + (n-1)q_{n-1}(1-q_n) + \cdots + (j+1)q_{j+1}(1-q_n)\cdots(1-q_{j+2}) \\ &\quad + j(1-q_n)(1-q_{n-1})\cdots(1-q_{j+1}). \end{aligned}$$

Multiplying out everything and dropping small high order terms, we get

$$\begin{aligned} S_n^{(n)} &\approx nq_n + \cdots + (j+1)q_{j+1} + j - j(q_n + \cdots + q_{j+1}) \\ &= j + ((n-j)q_n + (n-j-1)q_{n-1} + \cdots + q_{j+1}). \end{aligned}$$

(2) For LH: As in HL, we have

$$T_1^{(n)} \approx j - ((j-1)q_1 + (j-2)q_2 + \cdots + q_{j-1}).$$

Since false drops exist, it is clear that in the above two expressions $S_n^{(n)} \geq j$ is an over-estimation and that $T_1^{(n)} \leq j$ is an under-estimation. $\Box$

Since we know the exact tf value is j, we can check the errors introduced by these two methods:

$$\begin{aligned} |S_n^{(n)} - j| &\approx (n-j)q_n + (n-j-1)q_{n-1} + \cdots + q_{j+1}. \\ |T_1^{(n)} - j| &\approx (j-1)q_1 + (j-2)q_2 + \cdots + q_{j-1}. \end{aligned}$$

It is easy to see that $|S_n^{(n)} - j| \leq |T_1^{(n)} - j|$ provided $n - j \leq j - 1$. Therefore, $S_n^{(n)}$ is a better approximation of the true tf value if $j \geq (n+1)/2$.

The number $(n+1)/2$ is a very conservative estimation. In fact, since q_1 (the false drop probability for SF_1) is typically much larger than q_n (the false drop probability for SF_n), the tf value above which the HL approach yields better results than the LH method is much smaller than $(n+1)/2$ owing to the Zipfian distribution of terms among term frequencies. In the following discussion, $I_i(D)$ is used to denote the number of terms in D with term frequency i, and $l_i(D)$ the number of signatures in SF_i for document D. Given a document D and the parameters m_i, s_i, and w_i for each tf_i, we have the relation $l_i(D) \approx I_i(D)/s_i$. The Zipf Law can be mathematically approximated as [Salton 1989]:

$$I_i(D) \;=\; \frac{2}{i(i+1)} I_1(D) \quad \text{for } i \geq 1. \tag{6}$$

From formula (6), we can get

$$I_{>=n}(D) \;\approx\; \frac{2}{n} I_1(D). \tag{7}$$

For simplicity, we will use I_n to denote $I_{>=n}$.

Lemma 4 *Suppose Zipf's Law holds for our document collection, $n = 30$ and*

$$\frac{p_1}{s_1} \geq \cdots \geq \frac{p_n}{s_n} \tag{8}$$

then for any term occurring in the document with $tf \geq 4$, $S_n^{(n)}$ is a better approximation of the tf value than $T_1^{(n)}$.

Proof: Without loss of generality, we assume that the average number of terms in tf_1 for our document collection is I_1 and that $tf = j$. Then using (6) and (7), we have

$$\begin{aligned}
|S_n^{(n)} - j| &\approx (n-j) l_n p_n + (n-j-1) l_{n-1} p_{n-1} + \cdots + l_{j+1} p_{j+1} \\
&\approx \frac{2(n-j) I_1}{n} \frac{p_n}{s_n} + \cdots + \frac{2 I_1}{(j+1)(j+2)} \frac{p_{j+1}}{s_{j+1}} \\
&\leq I_1 \frac{p_j}{s_j} \left(\frac{2(n-j)}{n} + \frac{2(n-j-1)}{(n-1)n} + \cdots + \frac{2}{(j+1)(j+2)} \right) \\
|T_1^{(n)} - j| &\approx (j-1) q_1 + (j-2) q_2 + \cdots + q_{j-1} \\
&\geq I_1 \frac{p_j}{s_j} \left((j-1) + \frac{2(j-2)}{2 \times 3} + \cdots + \frac{2}{(j-1)j} \right)
\end{aligned}$$

Note that $I_1 p_j / s_j$ is a common factor in both formulas. A simple computation will show that $|S_n^{(n)} - j| \leq |T_1^{(n)} - j|$ for any $j \geq 4$. $\Box$

The condition (8) is not restrictive. Clearly, it is true when the signature lengths are uniform. We will show later that this condition also holds for our optimal signature configurations. From the proof, we can see that $tf \geq 4$ is still a conservative result. In our experiment with the TREC subset, HL is better than LH for $j >= 2$ using the configuration $n = 30$, $m = 29$, $s = 2$ and $w = 10$. Lemma 3 can be used to find the exact j when the statistical data about the document collection is available.

Lemma 3 considers the situation where the query term indeed exists in the documents. We now consider the case when there is no such word in the document (i.e., the match is a false drop).

Lemma 5 *In the case of false drop, the expected tf value given by $S_n^{(n)}$ and $T_1^{(n)}$ all have the form*

$$\sum_{i=1}^{n} i q_i + \textit{ high order terms.}$$

Proof: Eliminating all $S_i^{(n)}$ for $i = n-1, \cdots, 1$ from part (1) of Lemma 2, we get

$$S_n^{(n)} = \sum_{i=1}^{n} i q_i + \text{ high order terms } .$$

Similarly, we can obtain the expression for $T_1^{(n)}$. □

Theorem 1 *If the false drop probabilities p_i's and terms per signature s_i's satisfy*

$$\frac{p_1}{s_1} \geq \cdots \geq \frac{p_n}{s_n},$$

the HL search strategy is superior than the LH method.

Proof: From Lemma 4, we can see that for true matches, HL produces tf values which are closer to the true values than LH does on the average.

On the other hand, Lemma 5 indicates that the expected errors created by these two methods on a false match all have the form: $\sum_{i=1}^{n} i q_i +$ high order terms. The difference between the errors introduced by the two methods is negligible considering that the normal value of q_i is typically very small. Combining these two facts, we can claim that HL is better than LH. □

We will see in the following subsection that in the optimal signature configuration, the signature lengths of the partitions are increasing and the false drop probabilities are decreasing from low to high term frequencies. Under this condition, it is clear that HL is even more desirable than LH.

4.2 Optimal Signature Configuration

We have implicitly assumed so far that every signature partition uses the same signature length m, the same w and the same s. However, for the weight-partitioned signature file, different partitions may be assigned different signature lengths, different signature weights, etc. This implies that for a given total storage overhead, we can control the amount of storage allocated to each partition by changing the configuration of that partition. Thus, for a fixed total storage overhead, there is an optimal way of assigning signature lengths (or storage) to each partition to minimize the impact of false drops on precision and recall. This section derives the optimal signature configurations.

Since different SF_i's may have different signature configurations and hence different false drop probabilities, the weighted average of the false drop probabilities of all SF_i's is used to characterize the false drop probability of the whole signature file. The effective false drop probability can be written as

$$\begin{aligned} p(\vec{m}, \vec{w}, \vec{s}) &= \sum_{i=1}^{n} \frac{l_i(D)}{\sum_{j=1}^{n} l_j(D)} p(m_i, w_i, s_i) \\ &= \sum_{i=1}^{n} \frac{I_i(D)}{\sum_{j=1}^{n} I_j(D)} p(m_i, w_i, s_i). \end{aligned} \tag{9}$$

where $\vec{m}$ denotes $m_1, m_2, \cdots, m_n$. In this formula, $l_i(D)$ and $I_i(D)$, for $i = 1, \cdots, n$, are document dependent variables. We will use Zipf's law to eliminate them. Applying (6) and (7) to (9), we get a formula for the document independent effective false drop probability:

$$p = \sum_{i=1}^{n-1} \frac{1}{i(i+1)} p(m_i, w_i, s_i) + \frac{1}{n} p(m_n, w_i, s_i). \tag{10}$$

When $m_1 = \cdots = m_n$, $s_1 = \cdots = s_n$ and $w_1 = \cdots = w_n$ (i.e., all SF_i's use the same signature configuration), we can easily check that

$$p(\vec{m}, \vec{w}, \vec{s}) = p(m_1, w_1, s_1) = \cdots = p(m_n, w_n, s_n).$$

In other words, the uniform signature structure is just a special case of the non-uniform signature structure. We now present a result about the relation between the uniform signature length and the non-uniform signature length approaches.

Lemma 6 *For a signature file with parameters m, s, and w, reducing s by a factor of λ has the same effect on false drop probability as increasing m by a factor of λ, where $\lambda \geq 1$. Furthermore, increasing or decreasing m and s by the same factor does not change the false drop probability.*

Proof: We will use the false drop probability formula:

$$p(m, w, s) = (1 - (1 - w/m)^s)^k.$$

If fact, for small w and s (relative to m), we have

$$\begin{aligned}(1 - w/m)^s &= \left[(1 - \frac{1}{\frac{m}{w}})^{\frac{m}{w}}\right]^{\frac{ws}{m}} \\ &\approx e^{-\frac{ws}{m}}.\end{aligned}$$

Thus, we have $p(m, w, s) \approx (1 - e^{-\frac{ws}{m}})^w$. Now increasing m by a factor of λ, we get

$$p(\lambda m, w, s) \approx (1 - e^{-\frac{ws}{\lambda m}})^w.$$

Reducing s by a factor of λ, we get

$$p(m, w, s/\lambda) \approx (1 - e^{-\frac{w(s/\lambda)}{m}})^s = (1 - e^{-\frac{ws}{\lambda m}})^w.$$

We can reach the second conclusion in a similar way. □

To obtain formulas to determine the optimal signature configurations for the partitions, we first obtain a formula for the general case, where each partition is associated with a general ranking weight W.[1] Then, we can apply the formula to the case where tf is used as the ranking weight (we called this the TF-partitioned signature file). The theorem presented below determines the optimal signature configurations for the signature partitions to minimize the effect of false drops on retrieval effectiveness. The condition under which the theorem applies is similar to that for the false drop probability formula [Roberts 1979] (i.e., when w_i, s_i, and l_i are small relative to m_i, which is true for most applications).

It is well known that for the signature file approach, the space needed is proportional to $\ln(1/p)$. Given the space constraint, we will first compute the optimal false drop probabilities for all SF_i's. From the false drop probabilities, we can compute the other parameters such as signature lengths, bits per word, etc.

[1] In order to distinguish this weight, which is used to determine the importance of a match in the vector space model, from the signature weight, which is the number of ones in a signature, the former is hereafter called *ranking weight*.

We assume that document terms are partitioned into different groups, say, G_1, G_2, $\cdots$, G_n and each term in G_i carries a weight W_i. Without loss of generality, we can assume that $W_1 \leq \cdots \leq W_n$. The corresponding signature files are denoted as SF_1, SF_2, $\cdots$, SF_n as before. Under these assumptions, the storage overhead for a document is $\sum_{i=1}^{n} l_i m_i$, where l_i is the number of signatures in SF_i and m_i is the length of signatures in SF_i.

Lemma 7 *For false matches, the expected weight given by $S_n^{(n)}$ and $T_1^{(n)}$ all have the form*

$$\sum_{i=1}^{n} W_i q_i + \textit{ high order terms.}$$

Proof: Similar to Lemma 5. □

Lemma 8 *Suppose the storage constraint is $\sum_{i=1}^{n} l_i m_i = C$. In order to minimize the effect of false drops, the relations between different false drop probabilities p_i are:*

$$\begin{aligned} \frac{W_i}{s_i} p(m_i, w_i, s_i) &= \frac{W_j}{s_j} p(m_j, w_j, s_j) \quad \textit{for } 1 \leq i \leq j \leq n \\ p(m_n, w_n, s_n) &= exp\left(- \frac{C(\ln 2)^2 + \sum_{k=1}^{n} s_k l_k \ln(\frac{W_n s_k}{W_k s_n})}{\sum_{k=1}^{n} l_k s_k} \right) \end{aligned}$$

Proof: We know from the previous result that the expected tf value is

$$\sum_{i=1}^{n} W_i q_i$$

for a false drop. We will minimize the function $\sum_{i=1}^{k} W_i q_i$ under the storage constraint $\sum_{i=1}^{n} l_i m_i = C$.

The existence of a minimum value is clear since the constraint defines a compact set in space R^n and the function is continuous. It is also not difficult to see that the minimum value cannot be achieved on boundaries. First, from Lemma 1, we have

$$\sum_{i=1}^{n} W_i q_i \approx \sum_{i=1}^{n} W_i l_i p_i.$$

There are too many variables to handle. We first eliminate all m_i's. From formula (4), we have

$$m_i = (1/\ln 2)^2 s_i \log_e(1/p_i).$$

The constraint can be rewritten as

$$C = \sum_{i=1}^{n} l_i m_i = \sum_{i=1}^{n} (1/\ln 2)^2 s_i l_i \log_e(1/p_i).$$

The Lagrange's method can be used for obtaining the minimum. Let

$$F(\lambda, \vec{m}, \vec{w}, \vec{s}) = \lambda \left(\sum_{i=1}^{n} (1/\ln 2)^2 s_i l_i \log_e(1/p_i) - C \right) + \sum_{i=1}^{n} W_i l_i p_i.$$

The partial derivatives with respect to p_i's and λ are

$$\begin{aligned}\frac{\partial F}{\partial p_i} &= \lambda\frac{l_i s_i}{p_i} + W_i l_i \quad i = 1, \cdots, n. \\ \frac{\partial F}{\partial \lambda} &= \sum_{i=1}^{n}(1/\ln 2)^2 s_i l_i \log_e(1/p_i) - C.\end{aligned}$$

Now set all the partial derivatives to zero and cancel l_i from the i^{th} equation for $i = 1, \cdots, n$, we have

$$\frac{W_i}{s_i}p_i + \lambda = 0 \quad i = 1, \cdots, n. \tag{11}$$

$$\sum_{i=1}^{n}(1/\ln 2)^2 s_i l_i \log_e(p_i) + C = 0 \tag{12}$$

Eliminating λ from the i^{th} and j^{th} equations, we get

$$\frac{W_i}{s_i}p(m_i, w_i, s_i) = \frac{W_j}{s_j}p(m_j, w_j, s_j)$$

for $1 \leq i \leq j \leq n$. Setting $j = n$, we get

$$p_i = \frac{W_n s_i}{W_i s_n}p_n.$$

Substituting these equations into equation (12), we have

$$\sum_{i=1}^{n} s_i l_i \ln\frac{W_n s_i}{W_i s_n} + \left(\sum_{i=1}^{n} s_i l_i\right)\ln p_n + (\ln 2)^2 C = 0.$$

Solving for p_n, we get

$$p_n = exp\left(-\frac{C(\ln 2)^2 + \sum_{k=1}^{n} s_k l_k \ln(\frac{W_n s_k}{W_k s_n})}{\sum_{k=1}^{n} l_k s_k}\right).$$

$\Box$.

There are several things worth pointing out. First, since $W_1 \leq \cdots \leq W_n$, we have

$$\frac{p_1}{s_1} \geq \cdots \geq \frac{p_n}{s_n}.$$

These relations will ensure that HL is better than LH from Lemma 4. Secondly, it is a little surprising that the signature file size l_i disappeared in

$$\frac{W_i}{s_i}p(m_i, w_i, s_i) = \frac{W_j}{s_j}p(m_j, w_j, s_j).$$

This can be explained intuitively by the fact that there are two conflicting factors: large l_1 means more false drops, thus requiring more storage to be assigned to tf_1; on the other hand, with a fixed space, we cannot assign too much storage to tf_1. The optimal method we are presenting will give us an optimal way of allocating storage among all tf-groups.

Theorem 2 *Under the same assumption as in the previous Lemma, to minimize the effect of false drops, the signature length m_i, bits per word w_i and words per signature s_i must satisfy:*

$$m_i = \frac{1}{\ln^2 2} s_i \left(\ln(\frac{W_i s_n}{W_n s_i}) + \frac{C(\ln 2)^2 + \sum_{k=1}^{n} s_k l_k \ln(\frac{W_n s_k}{W_k s_n})}{\sum_{k=1}^{n} l_k s_k} \right) \tag{13}$$

$$w_i = \frac{1}{\ln 2} s_i (1 - 2^{-\frac{1}{s_i}}) \left(\ln(\frac{W_i s_n}{W_n s_i}) + \frac{C(\ln 2)^2 + \sum_{k=1}^{n} s_k l_k \ln(\frac{W_n s_k}{W_k s_n})}{\sum_{k=1}^{n} l_k s_k} \right), \tag{14}$$

where $i = 1, \cdots, n$.

Proof: From Lemma 8, we know for $1 \le i \le j \le n$:

$$\frac{W_i}{s_i} p_i = \frac{W_j}{s_j} p_j$$

$$p_n = exp \left(- \frac{C(\ln 2)^2 + \sum_{k=1}^{n} s_k l_k \ln(\frac{W_n s_k}{W_k s_n})}{\sum_{k=1}^{n} l_k s_k} \right).$$

If we let $j = n$, we get $p_i = \frac{W_n s_i}{W_i s_n} p_n$. This implies that

$$p_i = \frac{W_n s_i}{W_i s_n} exp \left(- \frac{C(\ln 2)^2 + \sum_{k=1}^{n} s_i l_i \ln(\frac{W_n s_k}{W_k s_n})}{\sum_{k=1}^{n} l_i s_i} \right).$$

From formulas (3), (4) and (15), we arrive at

$$\begin{aligned} m_i &= \frac{1}{\ln^2 2} s_i \ln \frac{1}{p_i} \\ &= \frac{1}{\ln^2 2} s_i \left(\ln(\frac{W_i s_n}{W_n s_i}) + \frac{C(\ln 2)^2 + \sum_{k=1}^{n} s_k l_k \ln(\frac{W_n s_k}{W_k s_n})}{\sum_{k=1}^{n} l_k s_k} \right) \\ w_i &= \frac{1}{\ln 2} s_i (1 - 2^{-\frac{1}{s_i}}) \ln \frac{1}{p_i} \\ &= \frac{1}{\ln 2} s_i (1 - 2^{-\frac{1}{s_i}}) \left(\ln(\frac{W_i s_n}{W_n s_i}) + \frac{C(\ln 2)^2 + \sum_{k=1}^{n} s_k l_k \ln(\frac{W_n s_k}{W_k s_n})}{\sum_{k=1}^{n} l_k s_k} \right) \end{aligned}$$

with some simple algebraic operations. □

In order to use Theorem 2, we need to decide C, s_i, and l_i. Note that only C and s_i's are design parameters, since $l_i = \lceil I_i/s_i \rceil$, which is determined by the document collection. C can be determined from the storage overhead desired. When C is fixed, the other parameters, namely, s_i's, need to be determined. The selection of s_i's needs some consideration. Suppose in a document collection, each document has on the average I_i terms with term frequency i. Then, a factor of I_1 should be used as s_1 to make sure we hash approximately the same number of words into a signature. For example, the TREC subset has on the average 105 terms in I_1. Then, $s_1 = 5$ is a good choice since each signature in SF_1 will contain 5 terms. On the other hand, $s_1 = 8$ will be a bad choice since one of the signatures will contain only one word. After we select $s_1 = s$,

$s_i = \lceil \min(s, I_i) \rceil$ is the logical choice for SF_i, where $i = 2, \cdots, n$. Finally, since we have $s_i l_i = \lceil I_i \rceil$, the optimal formulas (13) and (14) can be simplified as

$$m_i = \frac{1}{\ln^2 2} s_i \left(\ln(\frac{W_i s_n}{W_n s_i}) + \frac{C(\ln 2)^2 + \sum_{k=1}^{n} \lceil I_k \rceil \ln(\frac{W_n s_k}{W_k s_n})}{\sum_{k=1}^{n} \lceil I_k \rceil} \right) \tag{15}$$

$$w_i = \frac{1}{\ln 2} s_i (1 - 2^{-\frac{1}{s_i}}) \left(\ln(\frac{W_i s_n}{W_n s_i}) + \frac{C(\ln 2)^2 + \sum_{k=1}^{n} \lceil I_k \rceil \ln(\frac{W_n s_k}{W_k s_n})}{\sum_{k=1}^{n} \lceil I_k \rceil} \right). \tag{16}$$

Although it is natural to use different signature lengths in the weight-partitioned signature file method, it is difficult in practice (e.g., in terms of programming effort) to use different signature lengths for different SF_i's because the signature lengths must be memorized for signature generation. A better alternative is to use the same signature length for all signature partitions but different s_i values. The next Corollary gives an optimal signature file configuration in which all signatures have the same length.

Corollary 1 *Suppose the storage constraint is C, then the following signature file configuration will minimize the effect of false drops.*

$$\begin{aligned} m &= \min\{\bar{m}_1, \cdots, \bar{m}_n\} \\ s_i &= \left\lceil \frac{m}{\bar{m}_i} * \bar{s}_i \right\rceil \qquad \textit{for } i = 1, \cdots, n \\ w_i &= \frac{1}{\ln 2} s_i (1 - 2^{-1/s_i}) \left(\ln(\frac{W_i s_n}{W_n s_i}) + \frac{C(\ln 2)^2 + \sum_{k=1}^{n} s_k l_k \ln(\frac{W_n s_k}{W_k s_n})}{\sum_{k=1}^{n} l_k s_k} \right) \qquad \textit{for } i = 1, \cdots, n \end{aligned}$$

where $l_i = \lceil I_i / s_i \rceil$ and w_i, $\bar{m}_i$ and $\bar{s}_i$ satisfy Theorem 2.

Proof: We know from Lemma 6 that increasing or decreasing m and s by the same factor simultaneously will not change the false drop probability. We first using Theorem 2 to compute all $\bar{m}_i$'s and $\bar{s}_i$'s. Then, we select the shortest signature length (or whichever we prefer) as our new uniform signature length m and obtain s_i by scaling down $\bar{s}_i$ by the factor $m/\bar{m}_i$. The weight of each term, w_i, is the same as in Theorem 2. □

We can see that Theorem 2 and Corollary 1 both depend on $I_1, \cdots, I_n$, the sizes of the term frequency groups. These values can be determined by taking the averages of the term frequency group sizes over the document collection. Alternatively, we can use the Zipf's Law to get an approximation of $I_2, \cdots, I_n$ from I_1. From (6) and (7), we get

$$s_i = \lceil \min(s, I_i) \rceil = \left\lceil \min(s, \frac{I_1}{i(i+1)}) \right\rceil.$$

Then, we can use Theorem 2 to determine m_i's and w_i's and generate the signature files.

The general result can now be applied to the TF-partitioned signature file using the term frequencies as the ranking weights. That is, $W_i = i$ in eqns. 13–16. This amounts to assigning weights to a match in proportion to the term frequencies. Since false drops occurring at high term frequencies will have a large negative impact on the retrieval effectiveness, we want to allocate more storage to high term frequency partitions by increasing their weights.

4.3 Effect of false drops on document ranks

Another application of Lemma 3 is on false drop eliminations. False drop elimination is expensive to perform. When false drop elimination is not performed, we want to verify analytically that reducing the number of false drops beyond a certain threshold yields no gain in precision and recall. Furthermore, if p denotes the exact value of a parameter, p' denotes the approximate value of the parameter derived from the weight-partitioned signature file (i.e., with false drops). For example, $tf'(t)$ denotes the tf value of term t obtained in the search process and $tf(t)$ for the true tf value of t in the document; $\Theta'(D,Q)$ denotes the similarity obtained from the weight-partitioned signature file and $\Theta(D,Q)$ to denote the actual similarity if no false drop exist. $\Theta'(D,Q)$ is also used to denote the expected value of $\Theta(D,Q)$ for simplicity. The expected $\Theta'(D,Q)$ is a theoretical representation of $\Theta'(D,Q)$ on the average.

Lemma 9 *Given a query Q, the difference between the true similarity and the expected similarity of a document D obtained using the weight-partitioned signature file is a linear combination of q_i's.*

Proof: We divide the query terms into two groups $Q = Q_1 \bigcup Q_2$, where $Q_1 = D \bigcap Q$ and $Q_2 = Q \setminus D$. The true similarity between document D and query Q is given by

$$\Theta(D,Q) = \frac{\sum_{t \in Q_1} tf(t) \times idf(t)}{\sqrt{d}},$$

and the expected similarity obtained by the weight-partitioned signature file method is:

$$\Theta'(D,Q) = \frac{\sum_{t \in Q_1} tf'(t) \times idf(t) + \sum_{t \in Q_2} tf'(t) \times idf(t)}{\sqrt{d}}.$$

Due to errors caused by false drops, the tf value obtained by the HL method may be larger than the true tf value, although the difference is typically small. The formula for the expected tf values of the terms in Q_1 is given by Lemma 3. That is,

$$tf'(t) - tf(t) \approx ((n - tf(t))q_n + (n - tf(t) - 1)q_{n-1} + \cdots + q_{tf(t)+1}).$$

And the expected tf-value for elements of Q_2 is given by

$$tf'(t) = \sum_{i=1}^{n} i q_i.$$

The right hand side of these two expectations are linear combinations of q_i's for a fixed query Q and document D. From the above formulas, we have

$$\begin{aligned}
& \Theta'(D,Q) - \Theta(D,Q) \\
= & \frac{\sum_{t \in Q_1}(tf'(t) - tf(t)) \times idf(t)}{\sqrt{d}} + \frac{\sum_{t \in Q_2}(tf'(t)) \times idf(t)}{\sqrt{d}} \\
= & \sum_{t \in Q_1} \frac{idf(t)}{\sqrt{d}}(tf'(t) - tf(t)) + \sum_{t \in Q_2} \frac{idf(t)}{\sqrt{d}}(tf'(t)),
\end{aligned}$$

which is clearly a linear combination of q_i's since $tf'(t) - tf(t)$ for $t \in Q_1$, and $tf'(t)$ for $t \in Q_2$ are for every term t in Q. $\Box$

From Lemma 9, we can see that the expected $\Theta'(D,Q)$ approaches $\Theta(D,Q)$ as q_i's approach zero. For query Q, the vector-space model assigns a score $\Theta(D,Q)$ to each document. For simplicity, let's assume that no two documents receive the same score. In this way, a complete ranking of all the documents in $\mathcal{D}$ is produced. Under this assumption, the following theorem can be obtained.

Theorem 3 *For a document collection $\mathcal{D}$ and a set of queries $\mathcal{Q}$, there is a constant ϵ such that when the false drop probabilities for all SF_i's are smaller than ϵ, the expected precision and recall resulted from searching the weight-partitioned signature file for $Q \in \mathcal{Q}$ will be the same as that without false drops. In other words, reducing the false drop probabilities beyond this ϵ yields no gain in precision and recall.*

Proof: For each query $Q \in \mathcal{Q}$, the vector space model assigns a unique score $\Theta(D,Q)$ to each document D. Sorting all documents by decreasing order of their scores, we obtain a complete ranking. Let's use δ_Q to denote the minimum difference between two adjacent documents. In other words,

$$\delta_Q = \min\{\Theta(D_i,Q) - \Theta(D_{i+1},Q) \quad | \quad i \geq 1\}.$$

Finally, let $\delta = \min\{\delta_Q \mid Q \in \mathcal{Q}\}$. Since no two documents receive the same score, δ is a positive number. From Lemma 9, for each document D and query Q, we have

$$\lim_{q_i \to 0} (\Theta'(D,Q) - \Theta(D,Q)) = 0.$$

Thus, there must be a positive number ϵ such that when all of the false drop probabilities are less than ϵ,

$$0 \leq \Theta'(D,Q) - \Theta(D,Q) < \delta.$$

This means that the expected ranking order given by $\Theta'(D,Q)$ will be identical to the ranking order given by $\Theta(D,Q)$. Reducing false drop probabilities beyond this ϵ will not alter the precision and recall values. $\Box$

It is not always the case that no two documents receive the same score, but we can modify Theorem 3 to imply there is a number ϵ such that if all false drop probabilities are less than ϵ,

$$\Theta(D_1,Q) > \Theta(D_2,Q) \Longrightarrow \Theta'(D_1,Q) > \Theta'(D_2,Q).$$

Theorem 3 does not give any method to compute the threshold ϵ. The main difficulty is we don't have a prior collection of queries. Even with this knowledge, obtaining this number is not simple. Constant update of the collection makes it even more complicated to predict. Sampling or experimentation may be a practical method to obtain this threshold ϵ.

5 Experimental Results

The criteria for retrieval effectiveness are precision and recall. The goal of the experiments is:

1. to observe the extent to which false drops can be tolerated without causing unacceptable degradation to recall and precision,

2. to confirm the analytical results on the effectiveness of the LH and HL search strategies,

3. to observe the performance of various signature length assignment methods.

Experiments were performed with a document subset extracted from the TREC collection [Harman 1993b]. The full TREC collection supplied by NIST consists of about 2 Gbytes worth of articles. Since the full collection exceeded the resource of our computing environment, a subset, hereafter called the TREC subset, of 10,000 Wall Street Journal articles in 1987 with a raw size of about 27 Mbytes were used.[2]

In the experiments, a set of 25 standard queries were used; each of the queries came with a standard relevance judgment. The precision and recall values are obtained as follows. First, the standard queries are run on the collections. The ranked outputs of the queries are then compared to their respective list of relevant documents. Finally, the precision and recall values are calculated and averaged over all of the queries. In the computation, the recall values are rounded to the nearest 0.05 increment and interpolation is used to obtain the precision values for all 21 recall points (from 0 to 1, at 0.05 increments). Note that owing to interpolation, the precision at the zero recall point is actually the precision of the recall point which is nearest to zero. The 21 pairs of values can then be plotted in a precision and recall graph. This procedure is commonly used in evaluating text retrieval methods [Harman 1993a; Salton and McGill 1983]. For clarity and conciseness reasons, we further average the 21 precision values to obtain the *average precision* for each set of parameters used in the experiments. In order to observe the effects of false drops on precision and recall and storage overhead, we use a very wide range of false drop probability in the experiments. In practice, the false drop probability is typically below 1%.

Three different signature length assignment methods are tested in the experiments. The first two are derived from the optimal formulas (Eqn. 13) and (Eqn. 14) using (1) term frequencies as ranking weights of the signature partitions (denoted as TF in the graphs) and (2) uniform weights (denoted as U). The third method assigns signature lengths based upon the following table:

term frequency	multiplying factor
1	1
2	2
3–29	4
30	8

This method is based on the intuition that longer signature lengths should be used for higher term frequencies to reduce false drops occurred at high TF partitions. However, the multiplying factors are arbitrarily chosen. The inclusion of this method provides a data point with which the other two methods can compare. This method is denoted as EXP in the graphs.

Note that in all experiments, the maximum term frequency is set to 30. For large collections (used in experiments described later), there may be many terms with term frequencies greater than 30 and thus are mapped to $tf = 30$. This may result in a larger number of terms for $tf = 30$ (or whatever ceiling was chosen).

The storage overhead is used as the main controlling parameter in the experiments. Storage overhead is measured by the size of the signature file divided by the total size of the documents *after stemming and common word removal*. This measures the true storage overhead attributable to the signature file mechanism (as opposed to the space reduction achieved by stemming and common word removal). When the raw collection size is used, the storage overhead is much lower. For instance, in the TREC subset, the storage overhead calculated using the raw collection size is typically only half of the storage overhead calculated using our method.

[2]The subset was extracted at Online Computer Library Center (OCLC), Dublin, Ohio.

For a fixed storage overhead, the signature lengths and the other related parameters are computed based on the three methods described above.

Performance of HL and LH strategies

Figure 5 compares the average precisions of the HL and LH search strategies using the TREC subset. It is clear that the HL method is consistently better than the LH method, as predicted by our analysis. The difference is large for high false drop probabilities (i.e., low storage overheads) but narrows down as the false drop probability decreases. In general, the gap between the HL and LH methods increases when the number of false drops increases. This explains the large gap in the EXP method, because the EXP method produces the largest number of false drops given a fixed storage overhead. Since the HL method is much better than the LH method, we will only show the performance of the HL method in subsequent discussion.

Comparison of signature length assignment methods

The performance of the three signature file generation methods for the HL search strategies is shown in Figure 6. We can see from the figures that the EXP method is consistently the worst in all the test runs, whereas the uniform and optimal methods give much better performance. Further, the optimal method is the best everywhere. In fact, the LH approach in the optimal method is as good as the HL method in the uniform method. This confirms our previous discussion that the optimal method utilizes the space resource more effectively.

Effect of false drops on retrieval effectiveness

We can see from the graphs that although the average precision in general improves as false drop probability decreases, there is no observable improvement on retrieval effectiveness beyond a certain threshold. For the TREC subset, the average precision reaches the perfect level (when there is no false drops) when the storage overhead is about 37%. According to Figure 7, this corresponds to roughly 0.01% false drop probability and about 150 false drops generated from a query.

Number of false drops and false drop probability

Figure 7 shows the average number of false drops produced by a query and the weighted false drop probability. The former is obtained by counting the total number of false drops in each run and then dividing it by the number of queries in each run. The latter is computed from the analytical formula (Eqn. 9). We can see that the analytical and experimental data are very consistent with each other. The uniform and TF methods are almost identical in the number of false drops as well as in false drop probability. The EXP method is much worse than the other two methods.

6 Conclusion and Future Work

This paper presented a general technique for implementing the vector space statistical model with signature files. In this technique, terms in a document are grouped according to their term frequencies and hashed as a group into the signature file with the corresponding term frequency. Thus, when a match is found, the signature file in which the match occurs will indicate the term frequency of the matching term. This avoids storing the term frequency of each term explicitly.

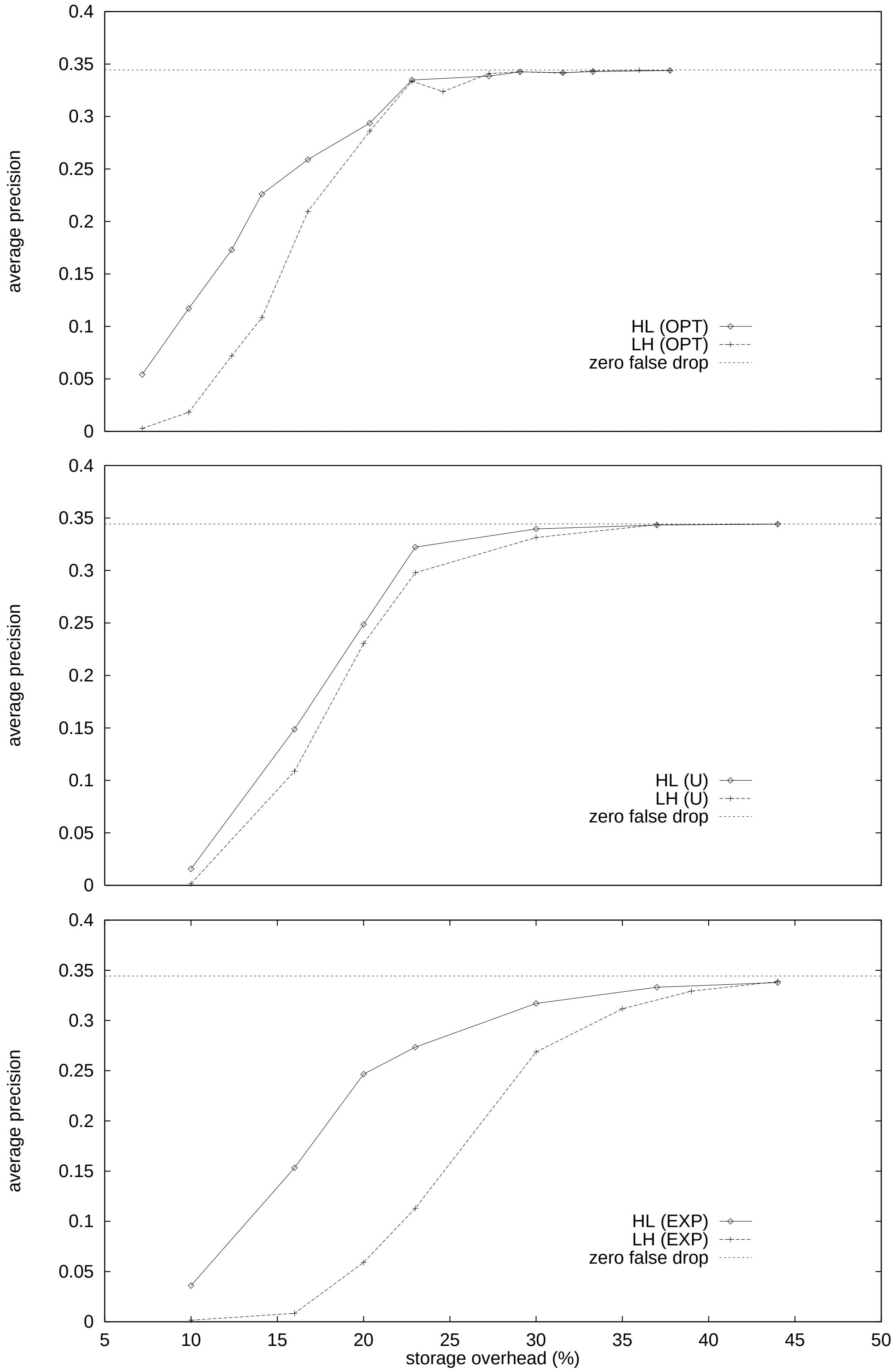

Figure 5: Performance of LH versus HL search strategies.

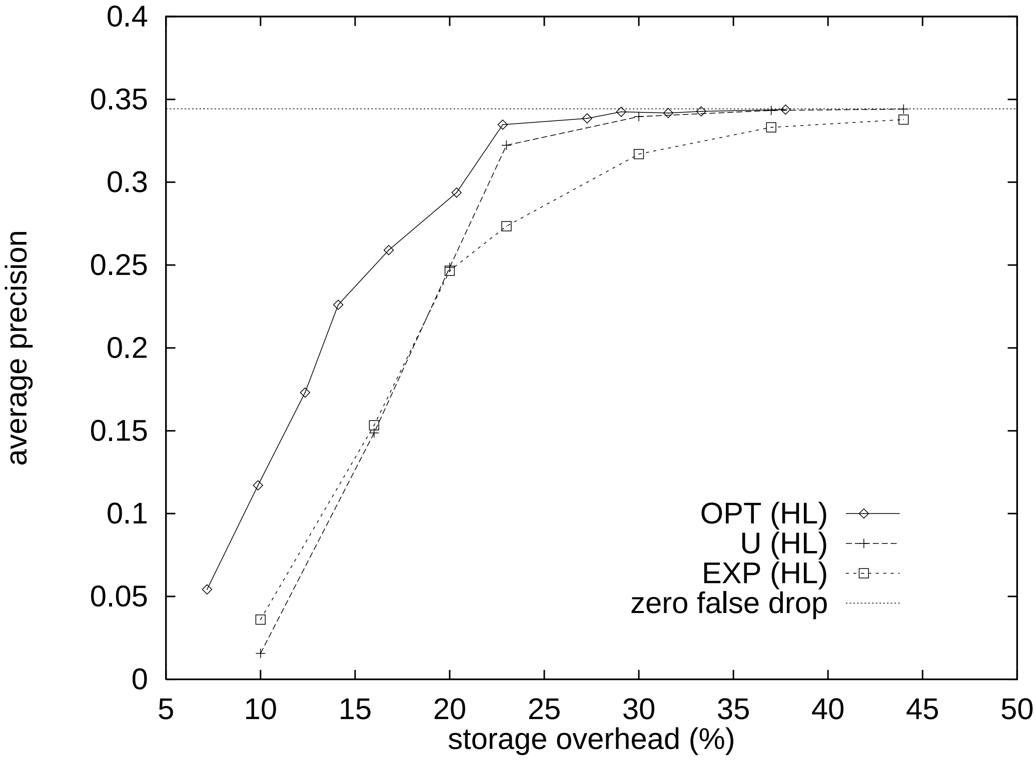

Figure 6: Comparison of signature length assignment methods (HL).

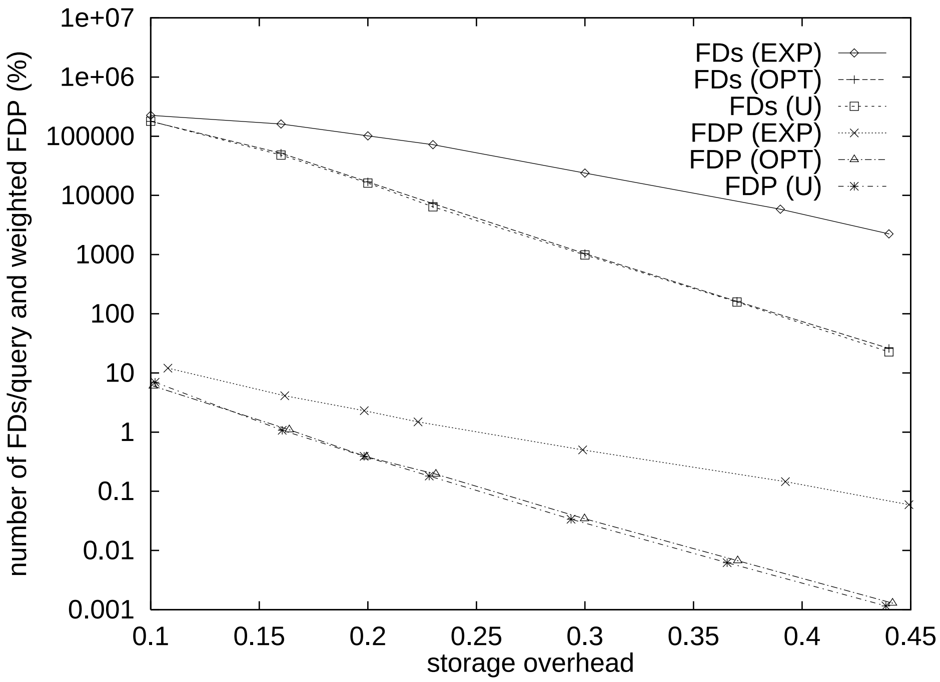

Figure 7: Number of false drops (FDs) and weighted false drop probability (FDP) (HL).

We presented an analytical study of the performance of two search strategies, namely, the LH and HL methods, and obtained an optimal way of assigning signature lengths, and thus storage, to the signature partitions to minimize the effect of false drops on precision and recall. Experiments were performed to support our analytical study.

The work presented in this paper is a significant step toward a viable way of supporting document ranking with signature files. The signature file technique will become much more competitive in speed if it doesn't have to perform false drop elimination. Another significant observation from the experiments is that a storage overhead of around 25% gives a precision which is quite comparable to the ideal case where there is no false drops. There is little gain in retrieval effectiveness if a larger storage overhead is used. However, there are still many interesting problems to be investigated. For instance, we are investigating search heuristics to further improve search performance and reduce the effect of false drops [Wong and Lee 1993]. We are investigating methods which use the signature file to produce a coarse ranking and apply exact ranking (without false drops) only to the top ranked documents, thus obtaining low storage overhead and high precision [Knaus and Schäube 1993].

Acknowledgment

The authors are indebted to the anonymous reviewers for their comments which led to many improvements on this paper.

References

Aho, A. V. , AND Ullman, J. D. (June 1979). *Optimal partial-match retrieval when fields are independently specified.* ACM Transactions on Database Systems, *4*(2), 168–179.

Christodoulakis, S., Theodoridou, M., Ho, F., Papa, M. and Pathria, A. (Oct. 1986). *Multimedia document presentation, information extraction and document formation in MINOS: A model and a system.* ACM Transactions on Office Information Systems, *4*(4), 345–383.

Croft, W. B. , AND Savino, P. (Jan. 1988). *Implementing ranking strategies using text signatures.* ACM Transactions on Office Information Systems, *6*(1), 42–62.

Harman, D. (Aug. 1993). *Overview of the second text retrieval conference (TREC-2).* Proceedings of the Second Text Retrieval Conference (TREC-2), Gaithersburg MD, 1–20.

——— (June 1993). *Overview of the first Text REtrieval Conference.* Proceedings of the Sixteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, Pittsburgh, PA, 36–47.

Knaus, D. , AND Schäube, P. (Aug. 1993). *Effective and efficient retrieval from large and dynamic document collections.* Proceedings of the Second Text Retrieval Conference (TREC-2), Gaithersburg MD, 163–170.

Lee, D. L. , AND Chuang, H. (1994). *Performance of document ranking and relevance feedback.* Submitted for publication.

Lee, D. L., Kim, Y. M. and Patel, G. (1994). *Efficient signature file methods for text retrieval..* To appear in IEEE Transactions on Data and Knowledge Engineering.

Lee, D. L. , AND Leng, C. -W. (Apr. 1989). *Partitioned signature files: Design issues and performance evaluation.* ACM Transactions on Information Systems, *7*(2), 158–180.

——— (Feb. 1990). *A partitioned signature file structure for multiattribute and text retrieval.* Proceedings of the Sixth International Conference on Data Engineering, Los Angeles, 389–397.

Lipman, D. J. , AND Pearson, W. R. (Mar. 1985). *Rapid and sensitive protein similarity searches.* Science, *227*, 1435–1441.

Roberts, C. S. (Dec. 1979). *Partial-match retrieval via method of superimposed codes.* Proc. IEEE, *67*(12), 1624–1642.

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

Salton, G. , AND Buckley, C. (1988). *Term-weighting approaches in automatic text retrieval.* Information Processing and Management, *24*(5), 513–523.

Salton, G. , AND McGill, M. J. (1983). Introduction to Modern Information Retrieval. New York, N.J., McGraw-Hill, Inc..

Stanfill, C. , AND Kahle, B. (Dec. 1986). *Parallel free-text search on the connection machine system.* Communications of ACM, *29*(12), 1229–1239.

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

——— (Sept. 1993). *Implementations of partial document ranking using inverted files.* Information Processing and Management, *29*(5), 647–669.

Zezula, P., Rabitti, F. and Tiberio, P. (Oct. 1991). *Dynamic partitioning of signature files.* ACM Transactions on Information Systems, *9*(4), 336–339.

Contents

List of Figures

List of Tables