Signature Files: An Access Method for Documents and Its Analytical Performance Evaluation

CHRIS FALOUTSOS and STAVROS CHRISTODOULAKIS
University of Toronto

The signature-file access method for text retrieval is studied. According to this method, documents are stored sequentially in the "text file." Abstractions ("signatures") of the documents are stored in the "signature file." The latter serves as a filter on retrieval: It helps in discarding a large number of nonqualifying documents. In this paper two methods for creating signatures are studied analytically, one based on word signatures and the other on superimposed coding. Closed-form formulas are derived for the false-drop probability of the two methods, factors that affect it are studied, and performance comparisons of the two methods based on these formulas are provided.

Categories and Subject Descriptors: H.2.2 [**Database Management**]: Physical Design—*access methods*; H.3.6 [**Information Storage and Retrieval**]: Library Automation; H.4.1 [**Information Systems Applications**]: Office Automation; I.7.0 [**Text Processing**]: Text Editing

General Terms: Design, Performance

Additional Keywords and Phrases: Document retrieval, information retrieval, office automation, signature file, superimposed coding, text retrieval

1. INTRODUCTION

Traditional database management systems (DBMSs) are designed for formatted records. Recently there seem to be many attempts to extend these systems so that they will be able to handle unformatted free text [8, 9, 14, 22, 33]. The major application of such extended systems is office automation. Many types of messages circulate in an office: correspondence, memos, reports, etc. These messages consist not only of attributes (e.g., sender, date) but also of text. In an automated office there should exist a system that allows electronic storage and retrieval of these messages.

Another important application of a text-retrieval method is the computerized library. The problem of handling queries on the contents has attracted a lot of research interest in the past few decades [30, 35]. As a result of the above research activity many text-retrieval methods have been proposed in the literature.

Authors' address: Computer Systems Research Institute, Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A4, Canada.

In this paper we concentrate on text-retrieval methods and especially on the signature-file approach, which seems to be most suitable for the office environment. Signature files can also be applied for attribute retrieval, and this makes the method suitable for documents. The structure of this paper is as follows: In Section 2 we provide an overview of the text-retrieval methods that have been proposed in the literature up to now. In Section 3 we describe the two methods to be compared and provide a mathematical formulation of the problem. We decide to use the false drop probability as the comparison criterion. In Section 4 we provide exact and approximate formulas for the false drop probability for the word-signature method. In Section 5 we do the same for the superimposed coding method. In Section 6 we perform the comparison of these methods, on the basis of the formulas of Sections 4 and 5. In Section 7 we provide a summary of the work and research directions.

2. OVERVIEW OF TEXT RETRIEVAL METHODS

In this paper, we ignore the existence of attributes in a document. We focus our attention on text-retrieval methods only. Many such methods have been proposed in the literature; however, they seem to form the following large classes:

1. *Full text scanning.* Given a search pattern, the whole database is scanned until the qualifying documents are discovered and returned to the user. The method requires no space overhead and minimal effort on insertions and updates. The main disadvantage is the retrieval speed. Despite the existence of some fast string searching algorithms [1, 4, 19], scanning of a large database may take too much time [15].
2. *Inversion.* This method uses an index. An entry of this index consists of a word (or stem or concept) along with a list of pointers. These pointers point to documents that contain this word. Many commerical systems have adopted this approach: STAIRS [17], MEDLARS, ORBIT, LEXIS [30], etc. The main advantage of the method seems to be its retrieval speed. However, it may require large storage overhead for the index; 50–300 percent of the initial file size, according to [13]. Moreover, insertions of new documents require expensive updates of the index.
3. *Signature files.* The documents are stored sequentially in the "text file." Their abstractions are stored sequentially in the "signature file." When a query arrives, the signature file is scanned sequentially, and a large number of nonqualifying documents are discarded. The rest are either checked (so that the "false drops" are discarded) or they are returned to the user as they are. A document is called a "false drop" if it does not actually qualify in a query, although its signature indicates it does. The method is faster than full text scanning but is expected to be slower than inversion [26]. It requires much smaller space overhead than inversion ($\approx$10 percent [8]), and it can handle insertions easily.
4. *Clustering.* In this method, similar documents are grouped together to form clusters. Usually, they are also physically contiguous in storage. Clustering is the dominating access method in the literature of library science [29, 30, 35]. It seems difficult to compare this method with the previous ones. The reason is that the emphasis in clustering is on the "relevance" queries, for example, "give me the

documents that are relevant to 'information retrieval' (even if a document does not contain the above two words)." Papers on the performance analysis of clustering are mainly concerned with *recall* and *precision* and seem to ignore the space overhead, the retrieval speed, and the efficiency upon insertions. Recall is the proportion of retrieved relevant documents over the total number of relevant documents. Precision is the proportion of retrieved relevant documents to the total number of retrieved documents. Our opinion is that the space overhead is rather small ($O(\log n)$, where n is the number of documents). Similarly, the retrieval time seems to be proportional to $\log n$ as well. However, it seems that clustering cannot handle insertions easily: Van Rijsbergen [35, pp. 58–59] observes that "sound" clustering methods usually need $O(n)$ time to perform an insertion, whereas "iterative" clustering methods may need reorganization, which takes $O(n \log n)$ time.

5. *Multiattribute hashing.* According to Knuth [18, pp. 562], Gustafson proposed a multiattribute hashing scheme based on superimposed coding in 1969. His method is applicable to bibliographic databases. All the keywords of a title are hashed and yield a signature for the title. A sophisticated one-to-one function transforms this signature into an address of the hash table. The interesting property of this method is that the number of buckets to be searched decreases exponentially with the number of search terms in the (conjunctive) query. However, no commercial system has applied this method, to the best of our knowledge.

This brief discussion on text retrieval methods reveals that none of them is clearly superior over the others. Therefore, we have to consider the operational characteristics of the specific environment, in order to choose the best access method. For the office environment the main features are

—Large databases: 1 Gbyte [3], or 65 Gbytes [16].
—Large insertion rates, but few deletions and updates [16].

Under the above considerations, the signature file method seems to be a reasonable choice: Tsichritzis and Christodoulakis [33] analyze the advantages of the method in more detail. In addition, it should be mentioned that the method seems to integrate well with attribute-retrieval methods: Signature files have been recently used for attributes [25, 27]. Tsichritzis et al. [34] describe a prototype, multimedia office filing system that applies the signature file approach both for attributes and text. Christodoulakis [7] applies this approach to image captions in order to handle queries on images.

Signatures, usually based on superimposed coding, have been used many times in the past in a variety of applications: to speed up the substring testing for text editors [12], to support differential files in database applications [31], to compress a dictionary for spelling programs [21], etc.

3. COMPARISON OF WORD SIGNATURES VERSUS SUPERIMPOSED CODING—PROBLEM FORMULATION

As mentioned before, the signature file approach seems to be a promising method for text retrieval, at least in the office environment. In this section we go into more detail and describe two methods of signature extraction. The first method

Document	free	text	retrieval	methods
	↓	↓	↓	↓
Word signature	0000	0100	0111	1011
Document signature		0000 0100 0111 1011		

Fig. 1. Illustration of the word signature (WS) method. The document consists of four words only. Each word yields a 4-bit word signature ($f = 4$).

Word	Signature
Free	001 000 110 010
Text	000 010 101 001
Block signature	001 010 111 011

Fig. 2. Illustration of the superimposed coding method. It is assumed that each logical block consists of D_{bl} = two words only. The signature size F is 12 bits; $m = 4$ bits per word.

[20, 33] suggests that each word of the document is hashed into a bit pattern of length f. These patterns (*word signatures*) are concatenated to form the document signature (see Figure 1). Searching is performed in the obvious way. For example, on a single word query the signature of the search word is extracted, and all the document signatures are searched. Those that contain the signature of the search word are retrieved. To improve performance, common words (e.g., "the," "a,") may be ignored. In the rest of this paper, we refer to this method as WS.

The second method [8] is based on superimposed coding [23]. It is referred to as SC for the rest of the paper. The method works as follows: Each document is divided into *logical blocks*. A logical block is defined as a piece of text that contains a constant number D_{bl} of distinct, noncommon words. Each such word yields a bit pattern of size F. These bit patterns are ORed together to form the block signature. The concatenation of the block signatures of a document form the document signature. The WS creation is rather sophisticated and needs more details: Each word yields m bit positions (not necessarily distinct) in the range 1–F. The corresponding bits are set to "1", and all the others bits are set to "0". For example, in Figure 2, the word "free" sets to "1" the 3rd, 7th, 8th, and 11th bits ($m = 4$ bits per word). In order to allow searching for parts of words, the following method is suggested: Each word is divided into successive, overlapping triplets (e.g., "fr", "fre", "ree", "ee", for the word "free"). Each such triplet is hashed to a bit position by applying a hashing function on a numerical encoding of the triplet. In the case in which a word has l triplets with $l > m$, the word is allowed to set l (nondistinct) bits. If $l < m$, the additional bits are set using a random number generator, initialized with a numerical encoding of the word.

Searching for a word is handled as follows: The signature of the word is created. Suppose that the signature contains "1" in positions 2, 3, 6, and 9. Each block signature is examined. If the above bit positions (i.e., 2, 3, 6, and 9) of the block signature contain "1", then the block is retrieved. Otherwise it is discarded. More complicated Boolean queries can be handled easily. In fact, conjunctive (AND) queries result in a smaller number of false drops. Even sequencing of words can be handled: It is replaced with conjunction (at the expense of increasing the number of false drops).

3.1 Problem Formulation

The main problem that we have to face in all the signature file methods is the control of false drops. False drops directly affect the number of block accesses (I/O time). Also, they affect the CPU time if we decide to apply full text scanning in order to decide whether a document should be returned to the user. The number of disk accesses depends on other design decisions, too, for example, bit slice storage of the signature file for SC [27, 28]. The I/O time and CPU time depend on the specific machine used, the devices used, the operating system (buffering algorithms), etc. A complete comparison of the two methods would compare the response times for a given space overhead. However, in this paper we tackle the problem of comparing the false drop probabilities only, for two reasons:

(1) The estimation of the number of false drops is the most difficult part of the comparison.
(2) We want to concentrate on the study of the inherent screening capacity of each method, independently of the hardware configuration.

A more detailed analysis would deal with a cost function of the form

$$\text{Cost} = aF + bm + cF_{\text{d}} + dA_{\text{d}},$$

where A_{d} is the selectivity (or actual drop probability), F is the size of a block signature in bits, m is the number of bits that a word sets to "1" in the SC method, F_{d} is the false drop probability, and a, b, c, d are proportionality constants, depending on the file size, the speed of the devices, the blocking factor, etc. Analytical estimation of the above parameters might be too involved. In this paper we concentrate on the following problem:

> Given the same space overhead for the signature file, which method exhibits the smallest false drop probability?

We have to define precisely the meaning of the false-drop probability.

Definition. False drop probability F_{d} is the probability that a block signature seems to qualify, *given that the block does not actually qualify.* Expressed mathematically,

$$F_{\text{d}} = \text{Prob}\{\text{signature qualifies/block does not}\}.$$

The procedure for calculating the F_{d} is as follows:

(a) The user submits a single word query.
(b) From the total number of blocks M in the database we discard the actually qualifying blocks M_{a}.
(c) From the remaining $M - M_{\text{a}}$ nonqualifying blocks we find the number of false drops M_{f}, whose block signature *seems* to qualify.

Table I. Definitions of Symbols Common to Both Methods

Symbol	Definition
F	Signature size in bits
D_{bl}	Distinct, noncommon words per logical block
V	Vocabulary size: total number of distinct *noncommon* words in the database.
M	Number of (logical) blocks in the database
M_a	Number of actually qualifying blocks (actual drops)
M_f	Number of false qualifying blocks (false drops)
V	Vocabulary size
a_j	Number of blocks that the jth word appears in ("appearance" or "occurrence frequency" or "selectivity" of the word)
$\bar{a}$	Average number of appearances per word

In this case, F_d for the specific query is

$$F_d = \frac{M_f}{M - M_a}.$$

Averaging over all possible queries, we have the expected F_d we are looking for.

On the basis of the above discussion, we claim here that we need only study the case of unsuccessful search (i.e., the search word does not exist in any block). The case of successful search can be studied in exactly the same way: After the actual drops (step b) have been removed, we have an unsuccessful search on the remaining blocks. The only differences between a successful and an unsuccessful search are the following:

(1) The number of blocks is reduced to $M - M_a$.
(2) The vocabulary of the $M - M_a$ blocks is $V - 1$.
(3) The distribution of word occurrence is (probably) changed.

In the next two sections we prove that this claim gives approximate, but accurate, results under certain conditions, which usually hold in real environments.

In order to create a common framework for the comparison, we assume that

—In both methods, a document is split into "logical blocks." We calculate the false drop probability of such a logical block under both methods.
—In the WS method the duplicate words are eliminated: If a (noncommon) word appears more than once in a logical block, it yields a signature only the first time it is encountered. This way the comparison is fairer, because SC eliminates duplicates automatically.

The problem is defined as following:

Given that there are D_{bl} distinct noncommon words per logical block, and that the size of the signature of logical block is F,
Find the false drop probabilities for each method for *single word* queries.

3.2 Preliminaries

Before we start the analysis of the two methods, we think it is a good idea to present together all the symbol definitions (Tables I–III), to give a list of the parameter values that are typical in a real environment (Table IV), a list of the

Table II. Definitions of Symbols for the WS Method

Symbol	Definition
f	Size of word signature
$S_{\max}$	Maximum possible number of distinct word signatures
$F_{d,ws}$	False drop probability for the WS method
$F_{d,ws,us}$	False drop probability for the unsuccessful search
σ	Signature of the search word
w	Number of words that hash to σ
p	Probability that a word hashes to σ
a_{j_i}	Appearances of the ith word that hashes to σ
$\vec{A}$	$= (a_{j_1}, a_{j_2}, \ldots, a_{j_w})$
A	Total number of times that the signature σ appears in the signature file $A = a_{j_1} + a_{j_2} + \cdots + a_{j_w}$
$Q_{\vec{A}}$	Expected number of false qualifying block signatures for the specific value of $\vec{A}$
Q	Expected number of false qualifying block signatures (averaged over all possible values of $\vec{A}$)

Table III. List of Symbols for the SC Method

Symbol	Definition
m	Number of bits that each word sets to "1"
$F_{d,sc}$	False drop probability for the SC method
$F_{d,sc,us}$	False drop probability for the unsuccessful search
β	One of the bit positions specified in a query
w	Number of words that set β to "1"
p	Probability that a word set β to "1"
a_{j_i}	Appearances of the ith word that sets β to "1"
$\vec{A}$	$= (a_{j_1}, a_{j_2}, \ldots, a_{j_w})$
A	Total number of times that above w words appear in the database $A = a_{j_1} + a_{j_2} + \cdots a_{j_w}$
$Q_{\vec{A}}$	Expected number of false qualifying block signatures for a single bit query, for the specific value of $\vec{A}$
Q	Expected number of false qualifying block signatures for a single-bit query (averaged over all possible values of $\vec{A}$).

Table IV. Typical Values of the Basic Parameters for Large Files

V	10,000	
F	600	
M	10,000	
D_{bl}	40	
$\bar{a}$	40	$(= MD_{bl}/V)$
m	10	$(= F \ln 2/D_{bl})$
$S_{\max}$	32,768	$(= 2^{**}(F/D_{bl}))$

notational conventions (Table V), and a list of the approximations (Table VI). Also, we mention all the assumptions we are going to use. In Appendix 1 we mention some useful theorems and formulas, which are too complicated mathematically to be presented here.

The values in Table IV are chosen so that they agree with the ones in [8]. From this table we see that the following assumptions hold in typical

Table V. Notational Conventions

$\bar{x} = E[x]$	Mean value of the random variable x
σ_x^2	Variance of the random variable x
comb(n, m)	Combinations of n choose m

Table VI. Approximations

$\left(1 - \frac{1}{x}\right)^y \approx e^{-y/x}$	if $x \gg 1, \quad y \gg 1$	(VI-1)
$e^{-z} \approx 1 - z$	if $z \ll 1$	(VI-2)

environments:

Assumption 1. Large number of possible signatures: $S_{\max} \gg D_{\text{bl}}$.
Assumption 2. Large vocabulary: $V \gg D_{\text{bl}}$.
Assumption 3. Large database: $M \gg 1$.

A more debatable assumption is that the appearances (or selectivities) a_j are small. This will be true if we use a stop list of common words. This is easy to implement and effective: According to Dewey [10] the first 69 most frequent words occur 50 percent of the time, while the first 732 most frequent words occur 75 percent of the time. Since we are going to disregard the common words during the signature extraction, it is reasonable to assume that the remaining words will have small selectivities:

Assumption 4. Small selectivities: $M \gg a_j$.

In the analyses we need an estimate for the variance σ_a^2 of the occurrence frequency distribution a. First, we have to express this distribution mathematically. A realistic, as well as convenient distribution, is the geometric one (shifted by 1 to the right):

$$\text{Prob}\{a = n\} = g^{n-1}(1 - g) \qquad \text{for} \quad n \geq 1$$

with $g = (\bar{a} - 1)/\bar{a}$. The a is a random variable that takes value according to the following experiment: a number j is chosen randomly in the range $1 - V$; a_j is the value of a. The geometric distribution implies that most of the words appear once, fewer words appear twice, etc. According to the statistics by Dewey that were mentioned before, this seems to be the case. The variance can be shown to be $\sigma_a^2 = \bar{a}(\bar{a} - 1)$. Thus, we have

Assumption 5. The occurrence frequencies follow the geometric distribution whose variance is $\sigma_a^2 = \bar{a}(\bar{a} - 1)$.

One more assumption is necessary:

Assumption 6. Randomly selected words appear independently of each other in the blocks of the database.

4. ANALYSIS FOR WORD SIGNATURES (WS)

As mentioned above, we need only study the case of unsuccessful search. Let σ be the signature of the search word. Let

$$P(w, V, S_{\max}) = \text{Prob}\{w \text{ out of } V \text{ words hash to } \sigma, \text{ when there are } S_{\max} \text{ possible signatures}\}.$$

The above distribution is binomial:

$$P(w, V, S_{\max}) = \text{comb}(V, w) p^w (1-p)^{V-w}$$

with

$$p = \frac{1}{S_{\max}}.$$

Let a_j denote the number of blocks that the jth word appears in $(1 \le j \le V)$. We want to estimate the expected number of false drops, that is, nonqualifying blocks, whose block signature contains the search signature σ.

Let us condition on the event that w words (out of V) hash on σ and that these words are $\text{Word}_{j_1}, \text{Word}_{j_2}, \ldots, \text{Word}_{j_w}$. Then σ appears (accidentally) $A = \sum_{i=1}^{w} a_{j_i}$ times in the signature file and the expected number of block signatures that create (false) drops are

$$Q_A = S(A, M, D_{\text{bl}}).$$

Averaging over all possible comb(V, w) selections of w words out of V and over all the possible values of w, we have

$$F_{\text{d,ws,us}} = \frac{1}{M} \sum_{w=0}^{V} P(w, V, S_{\max}) \frac{1}{\text{comb}(V, w)} \sum_{j_1, j_2, \cdots, j_w} S(A, M, D_{\text{bl}}). \tag{1}$$

The above formula is exact, but it is too complicated to give some intuition about the behavior of the method. Fortunately, we can simplify it using the Assumptions 1–6.

Since $\bar{w} = V/S_{\max} \ll V$, we may assume that these w words are selected *with* replacement out of the V ones. In this case, the values a_{j_i} can be considered as independent random variables identically distributed. Applying Theorem 2 (eqs. (A2) and (A3) in Appendix A), we have

$$\bar{A} = \overline{aw}, \qquad \sigma_A^2 = (\bar{a})^2 \sigma_w^2 + \bar{w}\sigma_a^2.$$

Also,

$$Q = E[S(A, M, D_{\text{bl}})],$$

and, applying Theorem 1 (eq. (A1) in Appendix A), we have

$$Q = S(\bar{A}, M, D_{\text{bl}}) + \frac{\sigma_A^2}{2} S''(\bar{A}, M, D_{\text{bl}}) + \cdots,$$

where $S'' = d^2S/(dA)^2$. We show next that the second term is negligible with respect to the first one, if Assumptions 1–6 hold. We do not examine the higher order terms, because they seem to be even smaller than the second term. From eq. (A4) (see Appendix A), we have

$$S''(A, M, D_{\text{bl}}) = -M\left[\frac{1}{M^2} e^{-A/M}\right]$$

and, therefore,

$$Q = M\left[1 - e^{-\bar{A}/M}\left(1 + \frac{\sigma_A^2}{2M^2}\right)\right]. \tag{2}$$

For typical values of the parameters (see Table IV) we have

$$\begin{aligned} \bar{a} &= 40, \\ \sigma_w^2 &= Vp(1-p) = 0.305, \\ \bar{w} &= Vp = 0.305. \end{aligned}$$

Assuming that the distribution of word appearances a_j is geometric (Assumption 5), we have

$$\sigma_a^2 = \bar{a}(\bar{a} - 1) = 1560.$$

In this case, we have

$$\frac{\sigma_A^2}{2M^2} = \frac{40^2 \times 0.305 + 0.305 \times 1560}{2 \times 10{,}000^2} = 4.82 \times 10^{-6} \ll 1.$$

Therefore we can simplify eq. (2):

$$Q = M[1 - e^{-\bar{A}/M}].$$

Since $\bar{A} = \overline{aw} = (MD_{\text{bl}}/V)Vp = MD_{\text{bl}}/S_{\max}$,

$$Q = M[1 - e^{D_{\text{bl}}/S_{\max}}]$$

and, by eq. (1) and Assumption 1,

$$F_{\text{d,ws,us}} = 1 - \left[1 - \frac{1}{S_{\max}}\right]^{D_{\text{bl}}}. \tag{3}$$

Equation (3) can be justified in an interesting, intuitive way. It gives the answer to the question: Given the signature of a nonqualifying block, what is the probability that at least one of its D_{bl} word signatures will (accidentally) match the search signature σ?

4.1 Remarks

The independence assumption (Assumption 6) is perhaps the most interesting to discuss. It claims that the w words that hash to σ are scattered over the M blocks independently of each other. More precisely, if the words Word$_j$ and Word$_k$ both hash to σ and if i is an arbitrary block, then the independence assumption says that

$$\begin{aligned} &\text{Prob[block } i \text{ contains Word}_j/\text{it contains Word}_k] \\ &\quad = \text{Prob[block } i \text{ contains Word}_j]. \end{aligned}$$

We conjecture that if the hashing function is "good," the above condition is true *even if words do not appear independently*!

Let us now analyze the implications of eq. (3). It states that the false drop probability for the case of unsuccessful search

—does not depend on the vocabulary size,
—does not depend on the size of the database,
—does not depend on the occurrence frequencies of words,
—is not affected by word interdependencies.

The above observations fully justify our claim that we do not have to study the case of successful search for single word queries. If we are searching for the jth word, we remove the a_j qualifying blocks and then we perform an *unsuccessful search* on the remaining $M - a_j$ blocks, with a vocabulary of $V - 1$ words and occurrence frequencies of words $a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_V$. However, none of the above changes affects the false drop probability, which still remains

$$F_{\mathrm{d,ws,ss}} = F_{\mathrm{d,ws,us}} = F_{\mathrm{d,ws}} = 1 - \left[1 - \frac{1}{S_{\max}}\right]^{D_{\mathrm{bl}}}.$$

A final observation is that the false drop probability does not depend on the search word. This implies that the distribution of query frequencies

$$R_j = \mathrm{Prob}\{\text{the } j\text{th word appears in a (single word) query}\}$$

does not affect the false drop probability. If

$$R_0 = \mathrm{Prob}\{\text{search word does not exist}\},$$

then we have

$$F_{\mathrm{d,ws,nonunif.queries}} = \sum_{j=0}^{V} R_j F_{\mathrm{d,ws}} = F_{\mathrm{d,ws}}.$$

The final conclusion of this section is that the false drop probability for the WS method is given by

$$F_{\mathrm{d,ws}} = 1 - \left[1 = \frac{1}{S_{\max}}\right]^{D_{\mathrm{bl}}},$$

and this equation holds for arbitrary occurrence and query frequency distributions of words, as long as the selectivities a_j are small ($a_j \ll M$) and the database is reasonably large ($V \gg D_{\mathrm{bl}}$, $M \gg 1$).

5. ANALYSIS FOR SUPERIMPOSED CODING (SC)

As in the previous section, we study only the case of unsuccessful search. At the end of this section, we show that the case of successful search reduces to the case of unsuccessful search.

Here, a single word query specifies m bit positions (not necessarily distinct). We want to calculate the expected number of block signatures in which these positions have been set to "1".

We start by calculating Q, the expected number of block signatures in which one (say, the first) of the m bit positions has been set to "1". This bit position will be referred to by β. The probability

$$\text{Prob}\{w\} = \text{Prob}\{w \text{ words set } \beta \text{ to ``1'' in their word signature}\}$$

follows the binomial distribution

$$\text{Prob}\{w\} = \text{comb}(V, w)p^{w}(1-p)^{V-w} \tag{4}$$

with

$$p = \text{Prob}\{\text{an arbitrary word sets } \beta \text{ to ``1'' in its word signature}\}$$
$$= 1 - \left[1 - \frac{1}{F}\right]^{m} \approx \frac{m}{F}.$$

Q can be thought of as the expected number of false drops in a *single-bit* query. Let $\text{Word}_{j_1}, \text{Word}_{j_2}, \ldots, \text{Word}_{j_w}$ indicate the above w words, which appear in a_{j_1}, $a_{j_2}, \ldots, a_{j_w}$ blocks, respectively. Then, let

$$\vec{A} = (a_{j_1}, a_{j_2}, \ldots, a_{j_w}) \qquad \text{and} \qquad A = \sum_{i=1}^{w} a_{j_i}.$$

Otherwise stated, we pick a number w that follows the binomial distribution (eq. (4)) and then we choose w words from the vocabulary at random, without replacement. The sum of their occurrence frequencies gives the value of A. Conditioned on $\vec{A}$, the expected number of false drops is

$$Q_{\vec{A}} = S(A, M, D_{\text{bl}}).$$

Averaging over the possible values of $\vec{A}$ we have

$$Q = E[S(A, M, D_{\text{bl}})]$$

or, equivalently,

$$Q = \sum_{w=0}^{V} \text{Prob}\{w\} \frac{1}{\text{comb}(V, w)} \sum_{j_1, j_2, \cdots, j_w} \{S(a_{j_1} + a_{j_w} + \cdots + a_{j_w}, M, D_{\text{bl}})\}, \tag{5}$$

where the summation $\sum_{j_1, j_2, \cdots j_w}$ covers all the possible combinations of V choose w words. In the rest of this section we try to approximate eq. (5) with a simpler one. From Theorem 1 (eq. (A1) in Appendix A) we have

$$Q \approx S(\bar{A}, M, D_{\text{bl}}) + \frac{\sigma_A^2}{2} S''(\bar{A}, M, D_{\text{bl}}).$$

We again neglect the higher order terms and show that even the second term is negligible with respect to the first. We observe that $\bar{w} \approx mV/F \ll V$ (from Table IV we have that $m/F = 0.0167$). Thus we can assume that the w words are chosen with replacement. From Theorem 2 (eqs. (A2) and (A3) in Appendix A) we have

$$\bar{A} = \overline{wa} \qquad \sigma_A^2 = \sigma_w^2(\bar{a})^2 + \bar{w}\sigma_a^2.$$

From eq. (A4) (see Appendix A), we have

$$Q \approx M[1 - e^{-\bar{A}/M}] - \frac{\sigma_A^2}{2} M \frac{1}{M^2} e^{-\bar{A}/M}$$

or

$$Q \approx M - Me^{-\bar{A}/M}\left[1 + \frac{\sigma_A^2}{2M^2}\right].$$

From Table IV we have

$$\begin{aligned} p &\approx \frac{m}{F} = 0.0167, \\ \bar{w} &= Vp = 167, \\ \sigma_w^2 &= Vp(1-p) = 163, \\ \bar{a} &= 40. \end{aligned}$$

Assuming that the appearances a_j follow a geometric distribution (Assumption 5), we have

$$\sigma_a^2 = \bar{a}(\bar{a} - 1) = 1560.$$

Thus

$$\frac{\sigma_A^2}{2M^2} = \frac{163 \times 40^2 + 167 \times 1560}{2 \times 10{,}000^2} = 2.61 \times 10^{-3} \ll 1.$$

If the hashing function is "good," it is reasonable to assume that the bit positions are independently specified in the block signature of a nonqualifying block: If i is the block and β_1 and β_2 are two of the m bit positions specified by the query, then we assume that

Assumption 7. Prob{β_1 is "1"/β_2 is "1"} = Prob{β_1 is "1"}.

Thus the false drop probability for a single word query is

$$F_{\mathrm{d,sc,us}} = \left[\frac{Q}{M}\right]^m$$

or

$$F_{\mathrm{d,sc,us}} = [1 - e^{-(mD_{\mathrm{bl}}/F)}]^m.$$

Stiassny [32] optimized the above formula for m:

$$m_{\mathrm{opt}} = \frac{F \ln 2}{D_{\mathrm{bl}}},$$

where ln stands for the natural logarithm. In this case

$$F_{\mathrm{d,sc,us}} = [\tfrac{1}{2}]^{m_{\mathrm{opt}}}. \qquad (6)$$

5.1 Remarks

The conclusions that hold for SC are similar to those for the WS method. We have

$$F_{\mathrm{d,sc}} = [\tfrac{1}{2}]^{m_{\mathrm{opt}}}, \qquad (7)$$

$$m_{opt} = \frac{F \ln 2}{D_{\mathrm{bl}}}, \qquad (8)$$

both for successful and unsuccessful search, regardless of the occurrence and query frequencies, regardless of the vocabulary size V and the size of the database M. The assumptions are the same for WS: large vocabulary $V \gg D_{\mathrm{bl}}$, small selectivities $M \gg a_j$, independence assumption (Assumption 6) (words that hash on the same bit β are scattered independently over the (nonqualifying) blocks). Here we need an additional independence assumption (Assumption 7). This states that in the signature of an arbitrary, nonqualifying block the bit positions specified in the query are set to "1" independently of each other. We recall that Assumption 6 is still necessary (otherwise we can not use eq. (A4)).

The reader might have reservations about the independence assumptions. However, experimental work by the authors on a 3.3-Mbyte database of bibliographic entries indicates that eqs. (7) and (8) hold regardless of word interdependencies (see [8, figs. 7–9]). On the basis of these experiments, we have strong reasons to believe that the independence assumptions (Assumptions 6 and 7) hold for the SC method. Moreover, these results make us project that Assumption 6 will hold for the WS method, although no experimental data are available.

6. COMPARISON OF WS VERSUS SC

Equations (3), (7), and (8) have been derived with realistic approximations and describe the "screening" capacity of each method rather accurately. Initial graphs of the above functions showed that ln $F_{\mathrm{d,ws}}$ and ln $F_{\mathrm{d,sc}}$ are almost linear with respect to the signature size F, for typical values of F and D_{bl}. This is not accidental. The above equations can be simplified even more, in order to illustrate this linear behavior. Equation (3) gives, by eq. (VI-2) (see Table VI) and Assumption 1

$$F_{\mathrm{d,ws}} \approx 1 - \left[1 \frac{1}{S_{\max}}\right]^{D_{\mathrm{bl}}} \approx 1 - \left[\frac{D_{\mathrm{bl}}}{S_{\max}}\right]$$

or

$$\ln F_{\mathrm{d,ws}} \approx \ln D_{\mathrm{bl}} - \frac{F}{D_{\mathrm{bl}}} \ln 2. \tag{9}$$

For SC, eqs. (7) and (8) give

$$\ln F_{\mathrm{d,sc}} \approx -\frac{(\ln 2)^2}{D_{\mathrm{bl}}} F. \tag{10}$$

There are two parameters we can vary in the above formulas: F and D_{bl}. F determines the space overhead (size of the signature file) and D_{bl} determines the size of the "logical" blocks into which we divide our documents.

6.1 Variable Signature Size

First we consider a typical, constant value for D_{bl} ($D_{\mathrm{bl}} = 40$) and we vary F. The results are plotted in Figure 3. We see that SC is better for small signature sizes, but WS improves faster than SC. The crossing point for $D_{\mathrm{bl}} = 40$ corresponds to $F = 693$ bits, with $F_{\mathrm{d}} = 2.4 \times 10^{-4}$. The reason that the line for WS is steeper is not known. A possible explanation might be that WS makes full use of all the

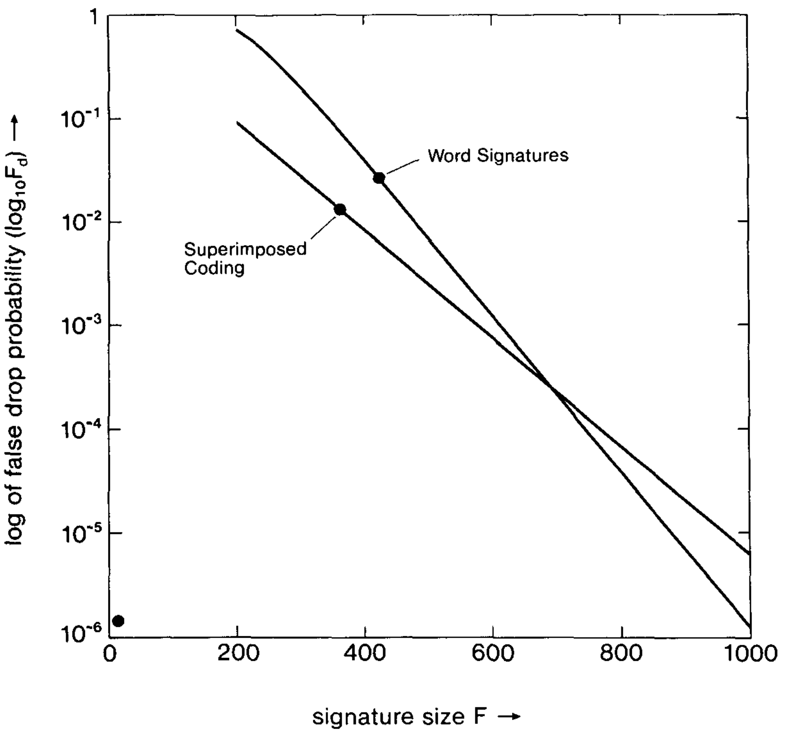

Fig. 3. $\log_{10}F_d$ versus F for WS and SC. $D_{bl} = 40$ = constant.

Table VII. Symbol Definitions

v	Space overhead: (size of a block signature in bits)/(size of a block in bits)
N	Number of (nondistinct) words in a document
bw	Expected number of bits to represent a word in the text file

available bit patterns (2^F) in a block signature, whereas SC has the restriction that half of the bits should be set to "1" (see eq. (8)), thus restricting the number of bit patterns to comb(F,F/2).

6.2 Variable Block Size

It is interesting to see how each method behaves when the size of the logical blocks changes. However, the comparison is not as easy as previously. We can not keep a constant signature size F. It is more reasonable to keep a constant space overhead v (For symbol definitions see Table VII.). The problem is to calculate the size of a logical block in bits, given that it contains D_{bl} distinct noncommon words. The answer to this problem depends on the contents of the documents. In order to obtain some intuition on the behavior of the two methods, we are willing to make the following assumptions:

—Average world length 5 characters [11]. Considering 1 byte for word delimiter, we need $bw = (5 + 1) \times 8 = 48$ bits per word.
—We do *not* delete the common words when deriving the signatures. This is going to add the *same* space overhead to both methods, and thus it will not affect the results of the comparison. Of course, we assume that queries on common words are not allowed (otherwise, the formulas for the false drop probabilities do not hold, since the assumption of small selectivites is violated).
—The words occur according to Zipf's distribution, even if the size of the text is small.

As shown in Appendix B, eq. (B1), the last assumption gives the relationship between the vocabulary of a block D_{bl} and the number of (nondistinct) words N in the block:

$$N = D_{bl} \ln(2D_{bl}) \qquad (11)$$

Equation (13) behaves as expected intuitively: The vocabulary D_{bl} increases more slowly than N. In addition, it has a number of strong points, compared with other functions that have been suggested in the literature [20, 26]:

—Dewey's results [10] seem to obey it ($N \approx 100{,}000$ and $D_{bl} = 10{,}119$).
—It is simple, without many magic constants.
—It is not derived empirically but is based on Zipf's law and some reasonable approximations.

Given the above symbol definitions and equations, we can express $\ln F_{d,ws}$ and $\ln F_{d,sc}$ as functions of the overhead v and D_{bl}. From eqs. (11) and (12) we have

$$\ln F_{d,sc} = -(\ln 2)^2 vbw \ln(2D_{bl}), \qquad \ln F_{d,ws} = \ln D_{bl} - (\ln 2) vbw \ln(2D_{bl}).$$

Figures 4 and 5 give the false drop probabilities as functions of D_{bl} on log-log axes for overheads $v = 10$ percent and $v = 8$ percent, respectively. From the above figures we see that SC becomes more and more competitive, as the block size D_{bl} increases, if the overhead v is small. For larger overhead, the WS method is better.

6.3 Remarks

The conclusions we can draw from the derived formulas and from Figures 3–5 are the following:

The handicap of the WS method is that a single word query is expanded to a disjunctive query. For example, searching for the word "free" is equivalent to the query

Word_1 = "free" or
Word_2 = "free" or
⋮
$\text{Word}_{D_{bl}}$ = "free"

and the false drops of the ORed terms are accumulated. (Word_k is the kth distinct, noncommon word of a block.) On the other hand, for large signature

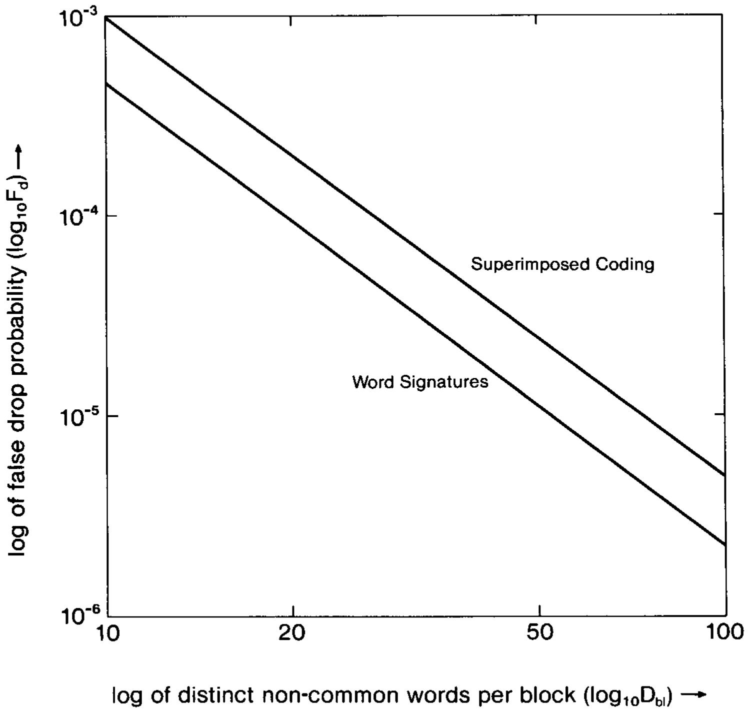

Fig. 4. F_d versus D_{bl} for overhead v = 10 percent. Both axes are logarithmic.

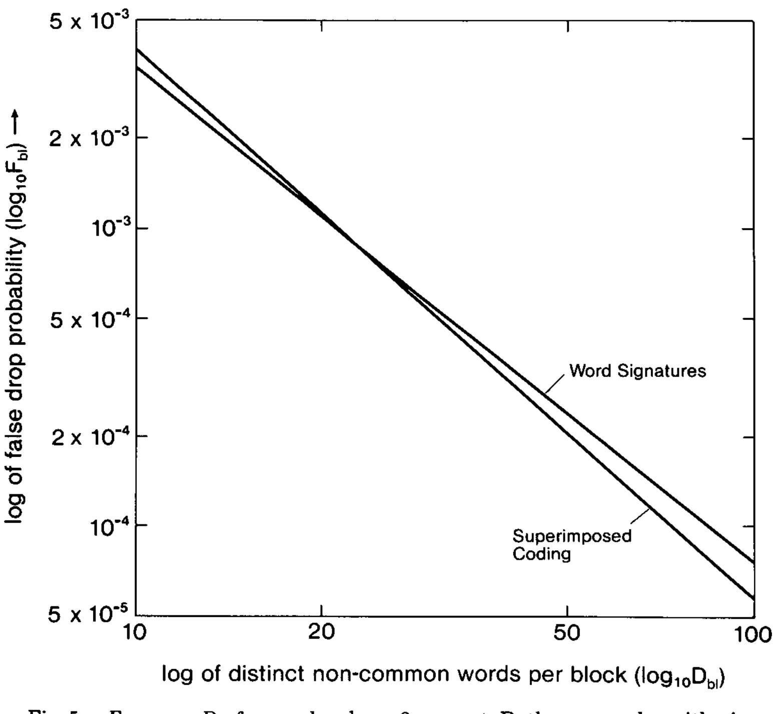

Fig. 5. F_d versus D_{bl} for overhead v = 8 percent. Both axes are logarithmic.

sizes ($S_{\max} = 2^f \gg V$), there are few collisions in the WS method and therefore few false drops.

The major conclusion is that SC is comparable to WS as far as the false drop probability is concerned. Some additional advantages of SC are the following [8]:

(1) SC requires a few simple bit operations on searching.
(2) It eliminates duplicates automatically.
(3) Using a hashing scheme based on triplets, queries on parts of words can be handled easily.
(4) The triplet-based hashing scheme cooperates well with an error-correcting method proposed by Angell, Freund, and Willet [2]. They suggested using the number of common triplets between two strings as a measure of their similarity.
(5) Bit-slice organization of the signature file like those developed for formatted data [27, 28] can greatly improve the I/O time and response time.

7. SUMMARY AND CONCLUSIONS

In this paper we have dealt with the problem of calculating the false drop probability of two signature extraction methods. Our goal was to study the factors that affect the screening ability of each method and to compare them. Elaborate formulas (eqs. (1) and (5)) have been derived, and it has been shown that they can be accurately reduced to formulas (3) and (6), respectively.

These two formulas lead to interesting conclusions about the factors that affect the false drop probability of each method: They show that it depends only on the signature size F and on the number of words per block D_{bl}. It is independent of the size of the database, the size of the vocabulary, the distribution of occurrence frequencies of the words, and the query frequencies. Moreover, the formulas are the same for successful and unsuccessful search. Experimental results by the authors [8] in the SC method seem to agree with eq. (6).

The comparison of the two methods showed that SC is better for small signature sizes, given a constant block size. Increasing the block size, SC becomes better and better for a constant, small space overhead.

The major contribution of this work is the derivation of eqs. (3) and (6), which give fast but accurate indication of the performance of each signature extraction method. By-products of this work are the comparison of the two methods in Section 6, as well as formula (B1), which relates the size of a document (in words) with the size of its vocabulary.

Future research could deal with

(1) Comparison of the two methods with respect to response time, for some typical hardware configuration.
(2) Analytical and experimental study of the response time of the bit-slice method [27], we well as its two-level extension [28].
(3) Searching for the best possible signature extraction method. Investigation of the relationship between the false drop probability and the entropy of the bits of the signature.

APPENDIX A

Here we mention two theorems from probability theory, as well as a formula to calculate the selectivities.

THEOREM 1 (e.g., [24, p. 151, eq. 5.61]). *If x is a random variable and $g(x)$ a function of it, then*

$$E[g(x)] = g(\bar{x}) + \frac{\sigma_x^2}{2} g''(\bar{x}) + \cdots + g^{(k)}(\bar{x}) \frac{\mu_k}{k!} + \cdots \qquad (A1)$$

where $g^{(k)} = d^k g/dx^k$ and μ_k is the kth-order central moment.

THEOREM 2 (e.g., [24, p. 249, eqs. 8.57–8.59]). *If a_j are independent, identically distributed random variables with mean $\bar{a}$ and variance σ_a^2 and w is an integer-valued random variable with mean $\bar{w}$ and variance σ_w^2, then for the random variable*

$$A = \sum_{j=1}^{w} a_j$$

we have

$$\bar{A} = \overline{aw}, \qquad (A2)$$

$$\sigma_A^2 = (\bar{a})^2 \sigma_w^2 + \sigma_a^2 \bar{w}. \qquad (A3)$$

We also need a formula $S(\cdot)$ that calculates the block selectivities: We have a database with M blocks, each containing D_{bl} distinct, noncommon words and we are processing the following w-term query:

Word_{j_1} OR Word_{j_2} OR $\cdots$ OR Word_{j_w}.

We want to find out how many blocks $Q_{\vec{A}}$ qualify on the average:

$$Q_{\vec{A}} = S(\vec{A}, M,, D_{\text{bl}})$$

where $\vec{A} = (a_{j_1}, a_{j_2}, \ldots, a_{j_w})$ with a_{j_i} being the number of blocks that Word_{j_i} appears in. According to the independence assumption (Assumption 6) and using a model *with* replacement we have

$$Q_{\vec{A}} = M\left[1 - \left[1 - \frac{a_{j_1}}{M}\right]\left[1 - \frac{a_{j_2}}{M}\right] \cdots \left[1 - \frac{a_{j_w}}{M}\right]\right].$$

We should recall that the duplicates of a word in a block are eliminated. According to Assumption 3 and eq. (VI-1) (see Table VI), we have

$$Q_{\vec{A}} = M\left[1 - \exp\left(\frac{a_{j_1} + a_{j_2} + \cdots + a_{j_w}}{M}\right)\right]$$

or

$$Q_{\vec{A}} = S(\vec{A}, M, D_{\text{bl}}) = M[1 - e^{-A/M}]$$

(where $A = \sum_{i=1}^{2} a_{j_i}$) or, to simplify the notation, we drop the vector from A:

$$S(A, M, D_{\text{bl}}) = M[1 - e^{-A/M}]. \tag{A4}$$

Notice that eq. (A4) is an approximation to the well-known formula suggested by Cardenas [5]. A more accurate formula for calculating the selectivity $Q_{\vec{A}}$ has been proposed by Christodoulakis [6]. However, formula (A4) yields accurate enough results for this environment, as it is discussed in the section with the conclusions. Therefore its use is justified.

APPENDIX B

Our goal here is to find the relationship between the number N of (nondistinct) words of a document (or a collection of documents) and the vocabulary V' (= number of distinct words, including the common words). The basic assumption is that Zipf's law is obeyed: If the ith most frequent word appears q_i times, then

$$q_i = \frac{C}{i}$$

where C is a normalizing constant. We have

$$\sum_{i=1}^{V'} q_i = N$$

or

$$C \sum_{i=1}^{V'} \frac{1}{i} = N.$$

But

$$\int_{0.5}^{V'+0.5} \frac{dx}{x} \approx \sum_{i=1}^{V'} \frac{1}{i}$$

according to the midpoint rule of integration. Thus, assuming that $V' + 0.5 \approx V'$, we have

$$C \approx \frac{N}{\ln(2V')}.$$

Since the least frequent word should appear approximately once, we have

$$q_{V'} \approx 1$$

or

$$N \approx V' \ln(2V'). \tag{B1}$$

REFERENCES

1. Aho, A. V., and Corasick, M. J. Efficient string matching: An aid to bibliographic search. *Commun. ACM 18*, 6 (June 1975), 333–340.
2. Angell, R. C., Freund, G. E., and Willet, P. Automatic spelling correction using a trigram similarity measure. *Inf. Proc. Manage. 19*, 4 (1983), 255–261.
3. Bird, R. M., Tu, J. C., and Worthy, R. M. Associative/parallel processors for searching very

large textual data bases. In *Proceedings of the 3rd ACM Workshop on Computer Architecture for Non-numeric Processing,* (May). ACM, New York, 1977, pp. 8-16.

4. BOYER, R. S., AND MOORE, J. S. A fast string searching algorithm. *Commun. ACM 20,* 10 (Oct. 1977), 762-772.
5. CARDENAS, A. F. Analysis and performance of inverted data base structures. *Commun. ACM 18,* 5 (May 1975), 253-263.
6. CHRISTODOULAKIS, S. Estimating block transfers and join sizes. In SIGMOD 83 Proceedings. *SIGMOD Rec. 13,* 4 (May 1983), 40-54.
7. CHRISTODOULAKIS, S. A framework for the development of a mixed-mode message system for an office environment. In *Proceedings of the 3rd Joint ACM-BCS Symposium on Research and Development in Information Retrieval* (1984). to appear.
8. CHRISTODOULAKIS, S., AND FALOUTSOS, C. Design considerations for a message file server. *IEEE Trans. Softw. Eng. SE-10,* 2 (March 1984), 201-210.
9. DATTOLA, R. FIRST: flexible information retrieval system for text. *J. Am. Soc. Inf. Sci. 30,* (Jan. 1979), 9-14.
10. DEWEY, C. *Relative Frequency of English Speech Sounds.* Harvard University Press, Cambridge, Mass., 1950.
11. GONNET, G. H. Unstructured Data Bases. Tech. Rep. CS-82-09, Computer Science Dept., Univ. of Waterloo, Waterloo, Canada, 1982.
12. HARRISON, M. C. Implementation of the substring test by hashing. *Commun. ACM 14,* 12 (Dec. 1971), 777-779.
13. HASKIN, R. L. Special-purpose processors for text retrieval. *Database Eng. 4,* 1 (Sept. 1981), 16-29.
14. HASKIN, R. L., AND LORIE, R. A. On extending the functions of a relational database system. *Proc. ACM SIGMOD.* ACM, New York, 1982, pp. 207-212.
15. HOLLAAR, L. A. Text retrieval computers. *IEEE Computer 12,* 3 (Mar. 1979), 40-50.
16. HOLLAAR, L. A., SMITH, K. F., CHOW, W. H., EMRATH, P. A., AND HASKIN, R. L. Architecture and operation of a large, full-text information-retrieval system. In *Advanced Database Machine Architecture,* D. K. Hsiao, Ed. Prentice-Hall, Englewood Cliffs, N.J., 1983, pp. 256-299.
17. IBM. *STAIRS/VS: Reference Manual.* IBM System Manual, 1979.
18. KNUTH, D. E. *The Art of Computer Programming.* Vol. 3, *Sorting and Searching.* Addison-Wesley, Reading, Mass., 1973.
19. KNUTH, D. E., MORRIS, J. H., AND PRATT, V. R. Fast pattern matching in strings. *SIAM J. Comput. 6,* 2 (June 1977), 323-350.
20. LARSON, P. A. A method for speeding up text retrieval. In *Proceedings of ACM SIGMOD Conference* (May), ACM, New York, 1983.
21. MCILROY, M. D. Development of a spelling list. *IEEE Trans. Commun. COM-30,* 1 (Jan. 1982), 91-99.
22. MCLEOD, I. A. A data base management system for document retrieval applications. *Inf. Syst. 6,* 2 (1981), 131-137.
23. MOOERS, C. Application of random codes to the gathering of statistical information. Bulletin 31, Zator Co., Cambridge, Mass., 1949.
24. PAPOULIS, A. *Probability, Random Variables and Stochastic Processes.* McGraw-Hill, New York, 1965.
25. PFALTZ, J. L., BERMAN, W. J., AND CAGLEY, E. M. Partial match retrieval using indexed descriptor files. *Commun. ACM 23,* 9 (Sept. 1980), 522-528.
26. RABITTI, F., AND ZIZKA, J. Evaluation of access methods to text documents in office systems. In *Proceedings of the 3rd Joint ACM-BCS Symposium on Research and Development in Information Retrieval.* 1984.
27. ROBERTS, C. S. Partial-match retrieval via the method of superimposed codes. *Proc. IEEE 67,* 12 (Dec. 1979), 1624-1642.
28. SACKS-DAVIS, R., AND RAMAMOHANARAO, K. A two level superimposed coding scheme for partial match retrieval. *Inf. Syst. 8,* 4 (1983), 273-280.
29. SALTON, G. *The SMART Retrieval System—Experiments in Automatic Document Processing.* Prentice-Hall, Englewood Cliffs, N.J., 1971.
30. SALTON, G., AND MCGILL, M. J. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

31. SEVERANCE, D. G., AND LOHMAN, G. M. Differential files: their application to the maintenance of large databases. *ACM Trans. Database Syst. 1*, 3 (Sept. 1976), 256–267.
32. STIASSNY, S. Mathematical analysis of various superimposed coding methods. *Am. Doc. 11*, 2 (Feb. 1960), 155–169.
33. TSICHRITZIS, D., AND CHRISTODOULAKIS, S. Message Files. *ACM Trans. Off. Inf. Syst. 1*, 1 (Jan. 1983), 88–98.
34. TSICHRITZIS, D., CHRISTODOULAKIS, S., ECONOMOPOULOS, P., FALOUTSOS, C., LEE, A., LEE, D., VANDENBROEK, J., AND WOO, C. A multimedia office filing system. In *Proceedings of the 9th International Conference on Very Large Data Bases*, (Oct.–Nov. 1983).
35. VAN-RIJSBERGEN, C. J. *Information Retrieval.* 2nd ed. Butterworths, London, 1979.

Received July 1984; revised September 1984; accepted September 1984