Design of a Signature File Method that Accounts for Non-Uniform Occurrence and Query Frequencies.

Christos Faloutsos
Stavros Christodoulakis

Computer Systems Research Institute
Univ. of Toronto

ABSTRACT. In this paper we study a variation of the signature file access method for text and attribute retrieval. According to this method, the documents (or records) are stored sequentially in the "text file". Abstractions ("signatures") of the documents (or records) are stored in the "signature file". The latter serves as a filter on retrieval: It helps discarding a large number of non-qualifying documents. We propose a signature extraction method that takes into account the query and occurrence frequencies, thus achieving better performance. The model we present is general enough, so that results can be applied not only for text retrieval but also for files with formatted data.

1. Introduction.

Traditional data base management systems (DBMSs) are designed for formatted records. Recently there are many attempts to extend these systems so that they will be able to handle unformatted, free text (Dattola 1979 [6]), (McLeod 1981 [16]), (Haskin & Lorie 1982 [11]), (Tsichritzis & Christodoulakis 1983 [22]), (Christodoulakis & Faloutsos 1984 [5]). 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 etc.) but also of text. In an automated office, these messages should be stored and retrieved automatically.

Another important application of a text retrieval method is the computerized library. The problem of handling queries on the contents has attracted much research interest in the past few decades (Salton & McGill 1983 [21]), (Van Rijsbergen 1979 [24]). As result of the above research activity many text retrieval methods have been proposed in the literature.

The outline of the paper is as follows: In section 2 we give a brief survey of some text retrieval methods, describe the characteristics of the office environment and give examples of how the signature file method works when we use superimposed coding. In section 3 we formulate the optimization problem. In section 4 we make some observations on the derived design formulas. In section 5 we demonstrate how our model could be applied to a formatted data base. In section 6 we discuss implementation problems and their solutions when we apply the method on textual data bases. In section 7 we present a summary and conclusions.

2. Overview of text retrieval methods.

The text retrieval methods that have appeared in the literature seem to form the following classes:

1) Full text scanning. Given a search pattern, the whole data base 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 (Aho & Corasick 1975 [1]), (Knuth, Morris & Pratt 1977 [15]), (Boyer & Moore 1977 [2]), scanning of a large data base may take too much time (Hollaar 1979 [12]).
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 commercial systems have adopted this approach STAIRS (IBM 1979 [14]), MEDLARS, ORBIT, LEXIS (Salton & McGill 1983 [21]) e.t.c.. The main advantage of the method seems to be its retrieval speed. However, it may require large storage overhead for the index: 50%-300% of the initial file size, according to Haskin (1981) [10]). 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 most of the non-qualifying 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, while its signature indicates the opposite. The method is faster than full text scanning but is expected to be slower than inversion (Rabitti & Zizka 1983 [18]). It requires much smaller space overhead than inversion (≈10% (Christodoulakis & Faloutsos 1984 [5])) and it can handle insertions easily.

4) Clustering. In this method, similar documents are grouped together to form clusters. Usually, they are stored physically together, too. Clustering is the dominating access method in library science (Salton 1971 [20]), (Van Rijsbergen 1979 [24]), (Salton & McGill 1983 [21]). However, it seems that clustering can not handle insertions easily: Van Rijsbergen (1979) [24, pp. 58-59] observes that "sound" clustering methods usually need $O(n)$ time to perform an insertion, while "iterative" clustering methods may need reorganization, which takes $O(n \log n)$ time. Moreover, the method allows false dismissals, that is, it may fail to retrieve some documents, even though they qualify ("recall" < 1).

5) Multiattribute hashing. Gustafson [9] proposed a multiattribute hashing scheme based on superimposed coding in 1969. His method is applicable on bibliographic data bases. A constant number of 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. Notice the difference between this method and signature files: In the latter, the signature is *stored* at the end of the signature file; during the query resolution, the signature file is scanned sequentially. The interesting property of multiattribute hashing 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.

One of the main characteristics of the office environment is the large insertion rate (Hollaar et al. 1983 [13]). Moreover, many documents are never retrieved after they have been filed. Gravina [8] reports that the access frequency decreases very fast with the age.

Under the above considerations, the signature file method seems to be a reasonable choice. Tsichritzis and Christodoulakis (1983) [22] analyze the advantages of the method in more detail. Tsichritzis et al. (1983) [23] describe a prototype, multimedia office filing system that applies the signature file approach both for attributes and text. Christodoulakis (1984) [4] applies this approach on image captions and text sections related to images in a document, in order to handle queries on images.

There exist a variety of signature extraction methods [7] one of the most promising ones [5] being based on superimposed coding [17]. It will be 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 of distinct, non-common words. The reason we have to do that is that performance deteriorates fast if there is a large variance on the number of words in a block. Each distinct, non-common word yields a bit pattern of size F. These bit patterns are OR-ed together to form the block signature. The concatenation of the block signatures of a document form the document signature. The word signature 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", while all the other bits are set to "0". For example, in Figure 1, the word "free" sets to "1" the 3-rd, 7-th, 8-th, and 11-th bits (m=4 bits per word). In order to facilitate searching for parts of words, each word is divided in successive overlapping triplets. Each such triplet yields one bit position that is set to "1".

Word	Signature
free	001 000 110 010
text	000 010 101 001
block signature	001 010 111 011

Figure 1
Illustration of the superimposed coding method.
It is assumed that each logical block
consists of D=2 words only.
The signature size F is 12 bits. m=4 bits per 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. Problem formulation.

In previous analyses [19, 5] it was assumed that the probability that any word appears in a users' query is uniform (uniform query frequency assumption), as well as that the words appear with equal frequency in the text (uniform occurrence frequency assumption).

Both these assumptions are unrealistic. It is well known that the frequency of word occurrence in large texts follows Zipf's law [25]. For example, if a database consists of e.g., technical reports about computers, then words such as 'data', 'program', 'algorithm' etc. will have rather high occurrence frequency, while words that do not belong to the computer science terminology will have lower occurrence frequency. On the other hand, the 'frequently' occurring words will probably not be used in queries often because they will retrieve too many documents.

We can improve the performance of the described method by allowing special treatment to words with high discriminatory power (high query frequency and low occurrence frequency): Each such word will be hashed to m_1 bit positions within the block signature

while the rest of the words will be hashed to $m_2 < m_1$ bit positions. Next we try to derive a formula that gives the optimum values of m_1, m_2.

We can partition the set S of all possible words of the text-file into two sets S_1 and S_2 such that $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \phi$. S_1 will contain the discriminatory words and S_2 the rest. We have to define a measure for the occurrence and query frequency. Let:

q_1 = Prob{ a single word query is about a word in S_1 }
and

$q_2 = 1-q_1$ = Prob{ a single word query is about a word in S_2 }.
Let D_1 be the average number of (distinct) words of the set S_1 in a block and similarly D_2 for S_2. Obviously, $D_1+D_2=D$: expected number of (distinct) words in a block.

As measure of the performance we choose the false drop probability F_d. This is the probability that a signature will qualify in a query, while the block does not actually qualify. Formally:

F_d=Prob{ a signature qualifies / the block does not}

To make the analysis tractable, we assume single word queries only. Note that we may partition S into more than 2 subsets without significant increase in complexity. Thus, assume that S is partitioned into n subsets $S_1,S_2,\ldots,S_n$, which are disjoint and whose union is S. The q_i 's and D_i 's are defined in the obvious way. The restrictions $\sum_{i=1}^{n} q_i = 1$ and $\sum_{i=1}^{n} D_i = D$ still hold. The definitions are summarized in Table I. Now we are ready to define the problem.

Given the query frequencies $q_1,q_2,\ldots,q_n$,
the expected number of distinct words per block $D_1,D_2,\ldots,D_n$ for each subset S_i and
the signature size F

Find the bits set to "1" per word $(m_1,m_2,\ldots,m_n)$ for each subset S_i, such that the false drop probability F_d is minimized:

$$F_d = q_1 w^{m_1} + q_2 w^{m_2} + \cdots + q_n w^{m_n} \qquad (1)$$

with

$$w = \frac{M}{F} \qquad (2)$$

the ratio of "1" s in a block signature.

The details of the analysis are presented in the Appendix. The results are repeated here (Eq. (A.4), (A.5), (A.9), (A.10)):

$$\frac{q_1 w^{m_1}}{D_1} = \frac{q_2 w^{m_2}}{D_2} = \cdots = \frac{q_n w^{m_n}}{D_n} \qquad (3)$$

$$w = \frac{1}{2} \qquad (4)$$

$$m_i = \frac{F \ln 2}{D} + \frac{1}{\ln 2}\left[\ln \frac{q_i}{D_i} - \frac{\sum_{k=1}^{n} D_k \ln \frac{q_k}{D_k}}{D}\right] \qquad (5)$$

Symbol definitions.	
F:	size of a block signature in bits
F_d:	false drop probability
q_i:	query frequency for subset S_i
D_i:	expected number of distinct words from subset S_i in a block (occurrence frequency).
D:	expected number of distinct words in a block.
m_i:	bit positions for each word of the i-th subset.
w:	expected ratio of "1" s in a block signature.
M:	expected number of "1" s in a block signature.

Table I.
Definitions of symbols.

$$\ln F_d = \ln D - \frac{F(\ln 2)^2}{D} + \sum_{i=1}^{n} \frac{D_i}{D} \ln \frac{q_i}{D_i} \qquad (6)$$

4. Remarks on the Derived Formulas.

1) An interesting observation is that the optimum ratio w of "1" s in the block signature is still 0.5 (Eq. (4)). This value is optimum for many variations of superimposed coding, e.g., [Stiassny '60], [Mooers, '59], [Roberts, '79]. This value corresponds to maximum entropy in the block signature, from an information-theory point of view.

2) Eq. (3) confirms the intuition about the dependency of the optimal m_i on the query and occurrence frequency q_i and D_i. Since $0< w <1$, large query frequency implies large m_i while large occurrence frequency implies small m_i.

3) If

$$\frac{q_1}{D_1} = \frac{q_2}{D_2} = \cdots = \frac{q_n}{D_n}$$

then all the subsets S_i enjoy similar treatment ($m_1=m_2=\ldots=m_n$). This means that if the occurrence frequency of each set is proportional to each query frequency, then the best we can do is to use the same number of bits m_i for every set.

4) It is easy to show that both Eq. (5) and (6) reduce to the corresponding formulas if there is no partitioning ($n=1$). (see, e.g., [5]).

5) Knowledge of the q_i 's and the D_i 's and use of Eq. (5) may be useful. For illustration, assume that the 80-20 rule holds: 20% of the vocabulary in the text file receives the 80% of the users' interest (queries). In that case, we have two sets of words S_1 and S_2 with q_1=.8, q_2= .2, D_1= 8, D_2= 32 (D = 40 distinct words per block). In order to compare the two methods we allow the same signature size F and compare the false drop probability F_d that each method can achieve. From Eq. (6) we see that the relative savings s in false drops are given by the formula

$$s = 1 - \left[\frac{q_1}{1-q_1}\right]^{(1-2q_1)} \qquad (7)$$

For q_1=0.8, the savings are
s=56.47 per cent
while for q_1=0.9 (90-10 rule) the savings are
s=82.75 per cent

7) Another strong point of our model is that it is general and several real situations can be considered as more specific cases of the general case. For example, if we do not distinguish between words at all, we can use this method with n=1 and q_1=1. The method can also handle even the extreme case, where each subset S_i consists of one only word. The "extremity" of this case lies in the fact that rarely is it cost effective to measure the query and occurrence frequency of each individual word.

5. Application in formatted data bases.

The proposed model can be easily applied to formatted records, with or without repeating groups. An example best illustrates how this can be done: Consider telephone directory records with the following attributes:

A1: NAME, consisting of 3 character strings on the average ("repeating group").

A2: ADDRESS, consisting of 5 alphanumeric strings on the average ("repeating group").

A3: TELEPHONE NUMBER, a 7-digit integer.

An example data base is depicted in Figure 2.

NAME	ADDRESS	TEL. #
John Smith	125 Queen Victoria St.	999.9999
Mike Queen	17 Nelson Rd.	777.7777

Figure 2.
Example data base.

Roberts [19] suggested that, e.g., the strings "Queen" as a last name and "Queen" as a street name yield different signatures. He also suggested that, in general, all strings enjoy the same m in their signatures, with the exception perhaps of frequently occurring strings: In that case he suggested a rule of thumb in order to find a (smaller) m to assign to these frequent strings.

Our model gives exact solution to the above problem, taking not only the occurrence but also the query frequency into account. In the example data base of Figure 2, the set of frequently occurring words could be S_1 = {John, Smith, Mike}, while the set of non-frequent words would be
S_2 = { Queen, 125, Victoria, St, 17, Nelson, Rd, 999.9999, 777.7777 }

Another use of the developed model is to provide help in assigning different m's to each of the attributes. For example, if most of the queries are on "NAME" (but we are not willing to maintain statistics for each individual name), it would be reasonable to assign larger m_i to all the strings (values) of this attribute. In this case, the sets of our model would be:

S'_1 = { John, Smith, Mike, Queen }
S'_2 = { 125, Queen, Victoria, St, 17, Nelson, Rd }
S'_3 = { 999.9999, 777.7777 }

The generality of the proposed model allows even more complicated situations. In the previous setting, it may be found that the name "Smith" is very common and deserves special treatment. Under this condition, the sets will become:

S''_1 = { Smith }
S''_2 = { John, Queen, Mike }
$S''_3 = S'_2$ = { 125, Queen, Victoria, St, 17, Nelson, Rd }
$S''_4 = S'_3$ = { 999.9999, 777.7777 }

6. Applications in text data bases.

The proposed method can be easily applied in a textual data base, using two sets S_1 and S_2. S_1 contains the discriminatory words while S_2 contains the rest and need not be stored. One might argue that it is not practical to use the proposed approach with more than two sets S_1 and S_2. The problems with many sets are:

1) We need look-up tables, in order to record which word belongs to which set. This might consume too much space.

2) Insertion of a new document may introduce some new words and therefore rewriting of the look-up tables may be necessary. Thus, insertions are not handled easily any more.

The remedy we propose to these problems is to use triplets of letters as hashing elements, instead of words. We have already seen that a hashing method based on triplets is useful in our environment [5]. Our model and all the derived formulas can be used without significant changes: A "logical block" should contain a constant number of triplets now, D_i is the number of times that the i-th triplet appears in a block, q_i is the probability that a ***single triplet*** query refers to the i-th triplet. F_d (or better $F_{d,triplet}$) is the false drop probability in a single-triplet query. Notice that a word with say, 4 triplets, will have a false drop probability of $F_{d,triplet}$**4.

Using triplets as hashing elements eliminates all of the above problems (each triplet forms its own, singleton, set):

1) We need a look-up table of manageable size: The possible number of triplets is 27^3 = 19683 (the alphabet size is 27, if we include the blank and ignore the case of the letters). A straightforward approach would be to store all the possible triplets in a table. However, we can sort the entries lexicographically, in which case we need not store the triplets themselves, but just the values of the m_i's. If one byte is enough to store each of the m_i s, we need ≈ 20 Kb of additional space, which is reasonable. Further savings can be achieved, if we use half a byte for each m_i (assuming that no m_i is greater than 16 = 2**4).

2) Insertions of new documents can not introduce new triplets, because our look-up table contains all the possible triplets.

The search time of the table might seem to be problem. However, only one disk access is required per triplet (or even none, if we store the table in main memory (20 Kb), during the signature extraction). Moreover, the search time of the table is not going to affect greatly the time to answer a query: The time to

extract the signature of the search-word is negligible with the respect to the time to scan the signature file.

The only disadvantage of using triplets versus using words as hashing elements is the increased number of false drops: When looking for a word, a block may contain all the relevant triplets, but not in consecutive positions. However, we feel that this will be rather improbable for words of reasonable size (5-6 letters long).

The most serious implementation problem of the proposed methods (as well as of any other method that takes the above frequencies into account) is the measurement of the occurrence and query frequencies D_i and q_i. One approach is to try to obtain representative samples of the documents and the queries. Based on statistics on the above sets, estimates of the D_i's and q_i's can be obtained. Another problem is that the frequencies may change in the future. An elegant method to achieve reorganization in this case is not known yet. However, we may reasonably hope that changes in the occurrence frequencies will be smooth. On the other hand, sharp changes in the query frequencies may indicate that the whole data base is obsolete (the interests of the office workers have changed drastically), in which case reorganization is out of question.

7. Summary - Conclusions.

In this paper we have examined the signature file approach as an access method for formatted records and text. Specifically, we have discussed the method of superimposed coding and we have proposed a modification to it. We suggest assigning a different number m_i (= number of bits set to "1") to each element, according to the occurrence and query frequency. We have developed a mathematical model and derived closed form formulas that allow the optimal choice of m_i for each set S_i of the elements.

The theoretical results are interesting: The derived formulas indicate that, regardless of the occurrence and query frequencies, the block (or record) signatures should have half of their bits set to "1", under optimal design. The savings in false drops can be significant (≈50 per cent if the 80-20 rule holds).

The proposed model can be applied to formatted records, as well as text. In the former case, the method is mostly appropriate for files with large number of attributes and frequent insertions, where indexing does not perform well. Our method can handle even repeating groups or queries on parts of words (attribute values). Finally, since the signature file is searched sequentially, the requests can be batched. Tree organizations can not exploit the advantages of batching well [3].

Signature files using superimposed coding have been used for text retrieval in an office environment [5]. The method proposed here results in better performance, when the query and occurrence frequencies are known.

Future research could deal with handling time-varying occurrence and query frequencies. Another interesting problem is to extend the multiattribute-hashing method proposed by Gustafson [9] to take into account the occurrence and query frequencies.

Appendix.

The problem is to calculate m_i $(i=1,\ldots,n)$, such that the false drop probability F_d

$$F_d = q_1 w^{m_1} + q_2 w^{m_2} + \cdots + q_n w^{m_n} \qquad (A.1)$$

is minimized. w is the proportion of "1"s in the signature and is given by the formula

$$w = \frac{M}{F}$$

where M is the expected number of "1"s in the signature. We have

$$M = F\left[1-\left[1-\frac{1}{F}\right]^{m_1 D_1 + m_2 D_2 + \cdots + m_n D_n}\right]$$

or

$$M \approx F\left[1-e^{-\frac{m_1 D_1 + m_2 D_2 + \cdots + m_n D_n}{F}}\right]$$

The approximation holds for large values of F (typically $F \approx 600$). Thus, w is given by the formula

$$w \approx 1-e^{-\frac{m_1 D_1 + m_2 D_2 + \cdots + m_n D_n}{F}} \qquad (A.2)$$

where F, D_i and q_i $(i=1,\ldots,n)$ are given. In order to find the values of m_i 's that minimize F_d we differentiate it with respect to m_i 's:

$$\frac{\partial F_d}{\partial m_i} = 0$$

or

$$q_i w^{m_i} \frac{w}{1-w} \frac{F}{D_i} \ln w + q_1 w^{m_1} m_1 + q_2 w^{m_2} m_2 + \cdots + q_n w^{m_n} m_n = 0 \qquad (A.3)$$
$$(i=1,\ldots,n)$$

This is equivalent to

$$\frac{q_1 w^{m_1}}{D_1} = \frac{q_2 w^{m_2}}{D_2} = \cdots = \frac{q_n w^{m_n}}{D_n} = \frac{F_d}{D} = K \qquad (A.4)$$

where K is a constant, independent of i. Substituting Eq. (A.4) into (A.3) we obtain:

$$F \frac{w}{1-w} \ln w + m_1 D_1 + m_2 D_2 + \cdots + m_n D_n = 0$$

which, due to Eq. (A.2)) gives:

$$\frac{w}{1-w} = \frac{\ln(1-w)}{\ln w}$$

$$w = \frac{1}{2} \qquad (A.5)$$

Eq. (A.5) and (A.2) give:

$$m_1 D_1 + m_2 D_2 + \cdots + m_n D_n = F \ln 2 \qquad (6)$$

and Eq. (A.4) and (A.5) give:

$$m_i = \frac{1}{\ln 2}\left[\ln \frac{q_i}{D_i} - \ln K\right] \qquad (A.7)$$

Eq. (A.6) and (A.7) can be solved for K and give:

$$\ln K = \frac{-F(\ln 2)^2 + \sum_{i=1}^{n} D_k \ln \frac{q_k}{D_k}}{D} \qquad (A.8)$$

which allows to solve for m_i 's:

$$m_i = \frac{F \ln 2}{D} + \frac{1}{\ln 2}\left[\ln \frac{q_i}{D_i} \frac{\sum_{k=1}^{n} D_k \ln \frac{q_k}{D_k}}{D}\right] \qquad (A.9)$$

Thus, we have calculated the optimal values for the m_i 's for a given F. From Eq. (A.5) and (A.9) we see that

$$\ln F_d = \ln D - \frac{F(\ln 2)^2}{D} + \frac{\sum_{k=1}^{n} D_k \ln \frac{q_k}{D_k}}{D} \qquad (A.10)$$

References

1. Aho, A.V. and M.J. Corasick, "Fast Pattern Matching: An Aid to Bibliographic Search," *Communications ACM*, vol. 18, no. 6, pp. 333-340, June 1975.
2. Boyer, R.S. and J.S. Moore, "A Fast String Searching Algorithm," *CACM*, vol. 20, no. 10, pp. 762-772, Oct. 1977.
3. Christodoulakis, S., "Access Files for Batching Queries in Large Information Systems," *Proc. ICOD II*, Aug. 1983.
4. Christodoulakis, S., "A Framework for the Development of a Mixed-Mode Message System for an Office Environment," *Proc. 3rd Joint ACM-BCS Symposium on Research and Development in Information Retrieval*, Cambridge, 1984.
5. Christodoulakis, S. and C. Faloutsos, "Design Considerations for a Message File Server," *IEEE Trans. on Software Engineering* , vol. SE-10, no. 2, pp. 201-210, March 1984.
6. Dattola, R., "FIRST: Flexible Information Retrieval System for Text," *JASIS*, vol. 30, pp. 9-14, Jan. 1979.
7. Faloutsos, C., "Signature Files: Design and Performance Comparison of some Signature Extraction Methods.," *ACM-SIGMOD*, 1985. to appear.
8. Gravina, C.M., "National Westminster Bank Mass Storage Archiving," *IBM Systems J.*, vol. 17, no. 4, pp. 344-358, 1978.
9. Gustafson, R. A., "Elements of the Randomized Combinatorial File Structure," *ACM SIGIR, Proc. of the Symposium on Information Storage and Retrieval*, pp. 163-174, Univ. of Maryland, Apr. 1971.
10. Haskin, R.L., "Special-Purpose Processors for Text Retrieval," *Database Engineering*, vol. 4, no. 1, pp. 16-29, Sept. 1981.
11. Haskin, R.L. and R.A. Lorie, "On Extending the Functions of a Relational Database System," *Proc. ACM SIGMOD*, pp. 207-212, Orlando, Florida, 1982.
12. Hollaar, L.A., "Text Retrieval Computers," *IEEE Computer Magazine*, vol. 12, no. 3, pp. 40-50, March 1979.
13. Hollaar, L.A., K.F. Smith, W.H. Chow, P.A. Emrath, and R.L. Haskin, "Architecture and Operation of a Large, Full-Text Information-Retrieval System," in *Advanced Database Machine Architecture*, ed. D.K. Hsiao, pp. 256-299, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.
14. IBM,, *STAIRS/VS: Reference Manual*, 1979. IBM System Manual
15. Knuth, D.E., J.H. Morris, and V.R. Pratt, "Fast Pattern Matching in Strings," *SIAM J. Comput.*, vol. 6, no. 2, pp. 323-350, June 1977.
16. McLeod, I.A., "A Data Base Management System for Document Retrieval Applications," *Information Systems*, vol. 6, no. 2, pp. 131-137, 1981.
17. Mooers, C., "Application of Random Codes to the Gathering of Statistical Information," Bulletin 31, Zator Co., Cambridge, Mass., 1949. based on M.S. thesis, MIT, January 1948
18. Rabitti, F. and J. Zizka, "Evaluation of Access Methods to Text Documents in Office Systems," *Proc. 3rd Joint ACM-BCS Symposium on Research and Development in Information Retrieval*, Cambridge, 1984.
19. Roberts, C.S., "Partial-Match Retrieval via the Method of Superimposed Codes," *Proc. IEEE*, vol. 67, no. 12, pp. 1624-1642, Dec. 1979.
20. Salton, G., *The SMART Retrieval System - Experiments in Automatic Document Processing*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1971.
21. Salton, G. and M.J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.
22. Tsichritzis, D. and S. Christodoulakis, "Message Files," *ACM Trans. on Office Information Systems*, vol. 1, no. 1, pp. 88-98, Jan. 1983.
23. Tsichritzis, D., S. Christodoulakis, P. Economopoulos, C. Faloutsos, A. Lee, D. Lee, J. Vandenbroek, and C. Woo, "A Multimedia Office Filing System," *Proc. 9th International Conference on VLDB*, Florence, Italy, Oct.-Nov. 1983.
24. Van-Rijsbergen, C.J., *Information Retrieval*, Butterworths, London, England, 1979. 2nd edition
25. Zipf,, *Human Behaviour and Principle of Least Effort*, Addison Wesley , Cambridge, Massachusetts, 1949.