Message Files

DENNIS TSICHRITZIS and STAVROS CHRISTODOULAKIS
University of Toronto

We describe a message-filing capability which allows for the retrieval of messages according to contents. Messages are organized in large, general files such that frequent reorganization is avoided. The user specifies a filter which restricts the attention to a manageable subset of messages. Messages within the subset are retrieved for a final check. We discuss file organization and access method, as well as performance and implementation considerations.

Categories and Subject Descriptors: B.1.5 [**Control Structures and Microprogramming**]: Microcode Applications—*special-purpose*; C.4 [**Computer Systems Organization**]: Performance of Systems—*modeling techniques*; H.2.2 [**Database Management**]: Physical Design—*access methods*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*abstracting methods*; H.4.3 [**Information Systems Applications**]: Communications Applications—*electronic mail*

General Terms: Design, Performance

1. INTRODUCTION

Consider an office information system and the messages which circulate within it. We will assume for the purposes of this paper that the messages are text messages in machine-readable form. We are interested in designing a message-filing capability which has the following characteristics.

(1) It can deal with a wide variety of messages.
(2) It can retrieve the messages in a flexible manner.
(3) It provides a simple and uniform interface to the user.
(4) It can be implemented efficiently for a large volume of messages.

We define messages as consisting of a *header* and a *body* [12]. The header contains formatted data representing the most important characteristics of the messages (e.g., sender, date, destination). The body is text consisting of a series of words. We will denote by $A_0, A_1, \ldots, A_n$ the attributes of the header. A_0 is a special attribute which contains a unique system-wide identifier for the message. The body will be denoted as an attribute B of type text. A particular message will be represented as $(a_0, a_1, \ldots, a_n, b)$. All messages do not have to be of the same type. Each type of message, however, is represented by a set of attributes and a

This work was supported in part by the Natural Science and Engineering Council of Canada under Strategic Grant GO668.
Authors' address: Computer Systems Research Group, University of Toronto, Toronto, M55 IAI, Canada.

body. In the two extreme cases, we have the message type (A_0, B), that is, documents, and the message type $(A_0, A_1, \ldots, A_n)$, that is, records. Forms as messages can be represented as $(A_0, \ldots, A_n, B)$ with the additional stipulation that the values of A's are dispersed within B.

2. MESSAGE STORAGE AND RETRIEVAL

People in offices find paper messages by filing them appropriately. A *message file* is a labeled container of messages which groups messages according to their meaning. For instance, messages from a particular person about a particular project are filed together. Notice that a message file in the office environement allows filing of different types of messages within it (unlike a computer file). To help pinpoint the right file name for retrieval, file names can be structured according to their relationships. A common structure is a hierarchical structure as provided, for instance, by folders within drawers of file cabinets. People search a message file by scanning sequentially to obtain the appropriate message(s). They are guided by a partial specification of the contents of the message and by a certain vague image of what the message looks like.

There are some characteristics of office message files which we would like to retain for electronic filing. However, there are also limitations which we can and should avoid. First, we observe that it would not be difficult to duplicate the same kind of manual message filing in terms of electronic messages. For example, UNIX files provide hierarchical directories for file names [14]. Within each file, messages can be stored sequentially as byte strings. We only need to implement a sequential scan of messages which takes a sequence of bytes and displays it as a message. We would expect to find such capabilities associated with any kind of file server as part of an office information system. However, there are some limitations in such a filing facility.

First, the structure of the files is seldom static. The world changes, the message characteristics change, and their filing method should change accordingly. Even if we have achieved a perfect filing method encapsulated in a directory structure, it will soon be out of date. Reorganization of the files is difficult even in a computer environment. Messages will have to be reassigned to different files in a new directory. Second, many messages can potentially refer to many subjects and should be associated with many files. In such a case, we either have to duplicate the messages (with associated consistency problems) or we need to have pointers to messages rather than the messages in many files. Finally, no matter what file organization we choose we will not easily be able to divide messages into small groups according to contents. As a result, if the volume of the messages becomes high, the files will be large and sequential scan will be time consuming.

Clustering approaches have been used extensively in library environments [15]. A possible disadvantage of this approach is that the selection of index terms to be used has to be done by well-trained persons. We cannot expect people in an office environment to be trained or even to be willing to do this job. A second possible disadvantage of the clustering approach is that it is static. An office environment is a changing one, and reclustering is a very expensive operation.

We should augment the filing capability with an access method which can retrieve messages according to contents. In this way, messages can be organized

in general files rather than complex directories (e.g., 1981 general correspondence, project X memos). The information retrieval capability enables the user to retrieve the desired message from the file quickly. Since the messages are filed in very general terms, frequent reorganization of files is also not needed.

It would be nice to retain the properties of sequential scan; that is, messages are still retrieved one at a time simulating a sequential scan. It is, however, a "lucky" sequential scan. The user specifies a *filter*. Messages which do not qualify according to the filter are skipped. Messages which qualify are displayed. It is also important that they are displayed in a format recognizable to the user. In this way, the user can visually check the message before he accepts it.

In this paper we will concentrate on the specification, analysis, and implementation of such a filtering capability. We will assume that all messages are stored in large, general files according to user and role. The specified filter restricts the attention to a manageable subset of the messages. Messages within the subset are obtained sequentially for a final check by the user.

A major thesis of this paper is that the filtering capability does not have to be exact. We assume that the user will seldom be able to specify an absolutely tight filter. His specification in terms of contents will allow more messages to qualify in addition to the ones he absolutely wants. His visual inspection will finally pinpoint the desired messages. If the specification of the filter is not exact, its implementation does not have to be exact. In other words, suppose the specification of the filter allows 10 percent nonrelevant messages. If the implementation of the filter adds another 0.5 percent of nonrelevant messages the user will not care. The system could eliminate this extra 0.5 percent with some additional processing. However, it is as easy for the user to deal with 10.5 percent as to deal with 10 percent unwanted messages. This resiliency is present because we pass the results of the query to the user and not to a program.

3. FILE ORGANIZATION

Given a specification of the filter in suitable internal form, we need a file organization which restricts our attention to the set of appropriate messages.

The specification of values for A_i's can make use of standard indexes (e.g., B-trees) for their evaluation. The specification and evaluation of a pattern is more difficult. We could use indexing methods for words but we will need much space to store the word indexes [11]. In addition, merging the pointers obtained from the indexes is a time-consuming problem [11]. We propose instead a method of reducing the sequential search to a manageable level. The organization of the messages is sequential. The search is still sequential, but we search an auxiliary file rather than the message file.

Consider a word W appearing in the body of a message. We will denote by $S(W)$ a bit string which represents (not uniquely) the word W. There are many different algorithms for transforming W into $S(W)$. One method is to partition W into triplets or quadruplets of letters and to hash each partition of letters into a fixed number of bits. The bits are concatenated (up to a maximum length) to form the signature $S(W)$. A fixed number of bits may precede the signature $S(W)$ to indicate its length. Alternatively, signatures of words may be of fixed length.

Consider a set of messages of the form $m = (a_0, a_1, \ldots, b)$. Suppose that we physically file all the messages sequentially as they come into a general physical file F. Let i_k be a pointer to a message within F; that is, the file system will deliver the message on the basis of i_k. We will also costruct signature files which will aid the search for the message.

When the user stores a message he will specify one (or more) logical files in which he wants the message filed. For each logical file f there will be an associated physical file f_s which stores not the message but a representation of the message which we will call the *signature* of the message. The signature of the message will consist of $(i_k, t, S_a(a_0), \ldots, S_a(a_n), S_t(b))$ where

i_k is a pointer to where the message is stored,
t is the type of the message,
$S_a(a_0), \ldots, S_a(a_n)$ are exact representations of $a_0, \ldots, a_n$,
$S_t(b)$ is the signature of the body.

The signature of the body is a sequence of bits which approximately represents the significant words of the message's body. This is obtained by the following algorithm. Traverse sequentially the body b. For each word, check whether it is one of a set of common words (the, that, a, etc). If it is, set $S(W_u) = \varphi$, that is, skip the word. If it is not, substitute the word W with its signature $S(W)$. Skip also the blanks and formatting symbols.

The sequence of words comprising b is represented by the bit string $S_t(b)$ in the following manner. If a word W is in b, then $S(W)$ will be a substring of $S_t(b)$, provided that W is not a common word. Notice that the opposite is not true; that is, $S(W)$ may match a substring of $S_t(b)$ without W being a part of b. This situation may arise for two reasons. First, two words W and W' may have the same signature, $S(W) = S(W')$. We may, therefore, find W' rather than W in b. Second, if we use variable length word signatures without a prefix that specifies the length of the signature, the bit string $S(W)$ may match the suffix and prefix of the signature of the words W' and W''. This situation arises because blanks are eliminated.

Consider a series of messages $\{m_1, \ldots, m_t\}$ with bodies $\{b_1, \ldots, b_t\}$ and a word W. The set $\{m_i \mid S(W)$ matches $S_t(b_i)\}$ is a superset of the set $\{m_j \mid W$ matches $b_j\}$. It is important that the difference between the cardinalities of the two sets is small. We will come back to this point.

4. ACCESS METHOD

A user specifies a query to the message filing facility by first indicating the logical file f and the type of message t he is accessing. The system displays the appropriate template for t. The user optionally specifies some values for $A_0, A_1, \ldots, A_n$ and a pattern P by partially filling the template. Values entered for $A_0, A_1, \ldots, A_n$ are interpreted as selection conditions. The pattern P can be a complicated expression of words involving arbitrary Boolean combinations of regular expressions of words. Messages that satisfy the query are those that satisfy the conjunction of all selections specified, and the pattern P of words appears in the body of the message.

To find the set of qualifying messages, the system scans sequentially the signature file f_s associated with f. For each message signature, it checks first the type. If the type does not match, it skips the signature. If it does match, it tries to match first the given values for the attributes and then the signature of the body.

Consider the pattern P as expressed by a regular expression of words. The signature of the pattern $S_q(P)$ is a regular expression which is obtained by substituting $S(W)$ for each significant word in P, retaining the regular expression operators, dropping common words, and substituting $\{0 \cup 1\}^*$ for each unconstrained interval of words. The matching of message bodies for P is done by matching the signature of P, $S_q(P)$, against the signature of the bodies, $S_t(b)$. The signatures $S_t(b)$ are binary strings, and $S_q(P)$ is a regular expression of binary strings. Testing $S_t(b)$ for $S_q(P)$ is equivalent with testing whether

$$S_t(b) \in \{0 \cup 1\}^* S_q(P) \{0 \cup 1\}^*$$

This is a recognition problem for finite automata [1]. It can be shown that for a specific regular expression R the membership of a string in R can be checked by an NFA that has a number of states which is at most double the number of symbols in R [1]. It follows that given a pattern P testing for $S_q(P)$ in the bit strings $S_t(b)$ can be done by an NFA that has a number of states which is a linear function of the symbols in the pattern P.

5. PERFORMANCE CONSIDERATIONS

The performance of the file organization described depends on the percentage of unwanted messages retrieved, the relative cost of the sequential scan of a block of the signature file versus the cost of accessing a block of text, and the relative sizes of the signature file and the text files. The messages that qualify by testing the signature $S_t(b)$ for matching with the signature of a pattern P, $S_q(P)$, is a superset of messages that qualify for P. Larger signatures of words will result in less collisions (because the probability that two different words hash into the same signature becomes smaller). However, larger signatures imply extra cost for sequentially scanning the signature file. Thus, there is a trade-off between the sequential scan of a long signature file and the retrieval of a large superset of blocks from the message file.

In the following, we will assume that signatures of words have a fixed length of L bits. This eliminates the need for a prefix in the signature to specify its length, and it does not favor long words. Similar analysis can be applied for non-fixed-length signatures by using statistics on the lengths of words in printed English [2]. Let D be the number of distinct words in the text, and C the number of possible word signatures. For L bit word signatures, $C = 2^L$. Given a word W with signature $S(W)$, we want to know what is the expected number of other words W such that $S(W) = S(W')$. This can be modeled as a selection with replacement problem: For each distinct word W we select randomly a signature $S(W)$. The probability that k distinct words are mapped into a given signature is

$$p(k) = \binom{D}{k} \left(\frac{1}{C}\right)^k \left(1 - \frac{1}{C}\right)^{D-k}. \tag{1}$$

Thus the probability that more than one word W' has the same signature $S(W')$

$= S(W)$ with a given word W is given by

$$p(\text{collision}) = \frac{\sum_{k=2}^{D} p_k}{\sum_{k=1}^{D} p_k} = \frac{1 - p(0) - p(1)}{1 - p(0)} = 1 - \frac{(D/C)(1 - 1/C)^{D-1}}{1 - (1 - 1/C)^{D}}.$$

The number of distinct words can be directly found from the text. Thus, for given C and D the probability that more than one word has the same signature can be estimated. One would think that for a large amount of text, D would be very large. However, words in printed English follow Zipf's law [18]. Shannon has used the formula $p_n = 1/n$ to relate word frequency to rank, and estimated $D = 8727$ [16]. A more detailed estimation [9] gave $D = 12{,}370$ and Dewey's tables drawn over a sample of more than 100,000 words found 10,119 distinct words [7]. In comparison $C_1 = 2^8 = 256$ for 8-bit signatures and $C_2 = 65{,}536$ for 16-bit signatures.

Let l be the average occurrences of a distinct nontrivial word in the text file. A set of kl nontrivial words of the text file corresponds to the k distinct words that have the same signature. These words are placed into a number of blocks of text, $B(kl, N, M)$, where N is the total number of nontrivial words, and M is the total number of blocks of the text file. Given k, l, N, and M, $B(kl, N, M)$ can be estimated as the result of an experiment which selects, without replacing, kl balls from a pool of N balls containing balls of M different colors [3, 4, 5, 17]. A closed-form formula for randomly placed words among the blocks of the file [17] is

$$B(kl, N, M) = M\left(1 - \prod_{r=0}^{kl-1}\left(\frac{N - N/M - r}{N - r}\right)\right).$$

Therefore the expected number of blocks retrieved by a signature of a word W of text (given that at least one distinct word exists) is given by

$$\begin{aligned} E(B) &= \sum_{k=1}^{D} \frac{p(k)}{1 - p(0)} B(kl, N, M) \\ &= M \sum_{k=1}^{D} \frac{\binom{D}{k}}{1 - (1 - 1/C)^{D}} \left(\frac{1}{C}\right)^{k} \left(1 - \frac{1}{C}\right)^{D-k} \\ &\quad \times \left(1 - \prod_{r=0}^{kl-1}\left(\frac{N - N/M - r}{N - r}\right)\right). \end{aligned}$$

To avoid excessive calculations, $E(B)$ can be approximated by expanding $B(kl, N, M)$ around the expected value of kl [13]. Keeping only the first term of the expansion, we obtain

$$E(B) = B(n, N, M) = M\left(1 - \prod_{r=0}^{n-1}\left(\frac{N - N/M - r}{N - r}\right)\right), \tag{2}$$

where the expected number of words n that are retrieved from the signature $S(W)$ of an existing word W is given by

$$\begin{aligned} n &= \frac{\sum_{k=1}^{D} lk\binom{D}{k}(1/C)^{k}(1 - 1/C)^{D-k}}{\sum_{k=1}^{D} \binom{D}{k}(1/C)^{k}(1 - 1/C)^{D-k}} \\ &= \frac{l(D/C)}{1 - (1 - 1/C)^{D}} \\ &= \frac{N/C}{1 - (1 - 1/C)^{D}}. \end{aligned} \tag{3}$$

Since a document may span more than one block, this formula assumes that all the blocks of messages contain at least one occurrence of the word W in the pattern, or another W with $S(W) = S(W')$. For small documents, this assumption will not introduce serious errors.

Given a text file of N nontrivial words, with D distinct words, occupying M blocks of secondary storage, the expected number of blocks that are retrieved when a single word pattern is specified can be estimated using eqs. (2) and (3). Let $B(n, N, M)$ be the average number of blocks that are retrieved when a signature of a single word is specified. Users may specify patterns that are regular expressions of signatures. In the case that regular expressions are restricted to conjunctions, disjunctions (for synonyms), or sequencing (words that are consecutive in the text file), the average number of blocks retrieved by a query is given by

$$BT = rB(n_1, N, M) + \sum_{i=2} c_i \frac{B^i(n_1, N, M)}{M^{i-1}} + \sum_{i=2} s_i B(n_2, N, M) + \sum_{i=2} d_i B(n_3, N, M), \tag{4}$$

where

r is the frequency of single word patterns,
c_i is the frequency of patterns specifying conjunctions of i words of text,
d_i is the frequency of patterns specifying disjunctions of i words of text,
s_i is the frequency of patterns specifying sequences of i words of text,

and

$$n_1 = \frac{N/C}{1 - (1 - 1/C)^D},$$

$$n_2 = \frac{N/C}{1 - (1 - 1/C)^D} \left(\frac{1}{C(1 - (1 - 1/C)^D)} \right)^{i-1},$$

$$n_3 = \frac{N/C}{1 - (1 - 1/C)^D} C(1 - (1 - 1/C)^D) \left(1 - \left(1 - \frac{1}{C(1 - (1 - 1/C)^D)} \right)^i \right).$$

The first summation in eq. (4) corresponds to conjunctive patterns. The second summation corresponds to sequencing patterns. Its derivation is based on the observation that given a sequence of signatures the next signature is one of the existing distinct signatures in the signature file. From eq. (3) we see that the expected distinct signatures in the signature file is given by

$$D_s = \frac{N}{n} = C(1 - (1 - 1/C)^D).$$

The third summation corresponds to disjunctive patterns. Its derivation is based on the fact that we select signatures for i words from an existing set of $C(1 - (1 - 1/C)^D)$ distinct signatures. (This also can be modeled as a selection with replacement problem.) The above formulas assume uniform frequency of words and independence of the words specified. These are pessimistic assumptions [4, 6].

The total average cost is

$$C = CS \times BS + CT \times BT,$$

or

$$C = CS \times BS + CT \times \left(rB(n_1, N, M) + \sum_{i=2} c_i \frac{B^i(n_1, N, M)}{M^{i-1}} + \sum_{i=2} s_i B(n_2, N, M) + \sum_{i=2} d_i B(n_3, N, M)\right), \tag{5}$$

where CS is the cost of sequentially scanning a block of the signature file, CT is the cost of randomly accessing one block of the text file, BS is the number of blocks that are used for storing the signature file, and BT is the average number of blocks accessed from the text file as given by eqs. (4), (3), and (2).

One way to reduce the cost of the sequential scan of the signature file is to reduce the size of the signature file, BS, by removing all duplicate word signatures from the signature of the body of a message. Thus the signature of the body of a message will be composed of a set of distinct word signatures. In this case, queries involving sequencing of words can be satisfied by retrieving the messages containing the conjunction of the given words and discarding the nonqualifying messages. Since the number of messages that qualify in conjunctions of words is small on average, this technique may reduce the average cost C.

In eq. (5), CT/CS depends on the devices where the signature file and the text file are stored. It also depends on the size of the buffer and the block prefetching and replacement algorithms used. Finally, it depends on the way that data blocks of the signature file are stored on the secondary device (if logical order corresponds to physical order). All these factors can affect considerably the ratio CT/CS. Unfortunately the operating system support for these functions is not satisfactory. (For example, LRU is the worst possible replacement algorithm for scanning the signature file.) However, the use of large blocking factors in the signature file helps because this results in natural prefetching.

The size of word signatures should be chosen such that eq. (5) is minimized. Figure 1 shows the average cost $C/CS \approx BS + F \times B(n_1, N, M)$ as function of the number of bits used for the signatures of words. In this function, F takes into account the ratio CT/CS of costs of accessing a block of the text file versus the cost of accessing a block of the signature file, as well as the frequency of various types of patterns in queries. (This formula is accurate only when a small number of words in disjunctions is specified.) The optimum choice of the size of signatures depends on all the above factors. However, as the figure shows, for a large range of values of the parameter F, the minimum cost is achieved for signatures near 15 or 16 bits. This is also true for larger values of F that are not shown in the figure, and it is due to the fact that for this signature size the probability of two words having the same signature becomes small. Additional experimentation has shown that the choice of the optimal size is also relatively insensitive to the change of the text block size. For 8-bit byte machines, the choice of 16-bit signatures has the additional important advantage that only byte comparisons are used, and thus the cost of comparisons per bit is reduced. Thus, in such an environment, 16-bit signatures seem to be the best choice.

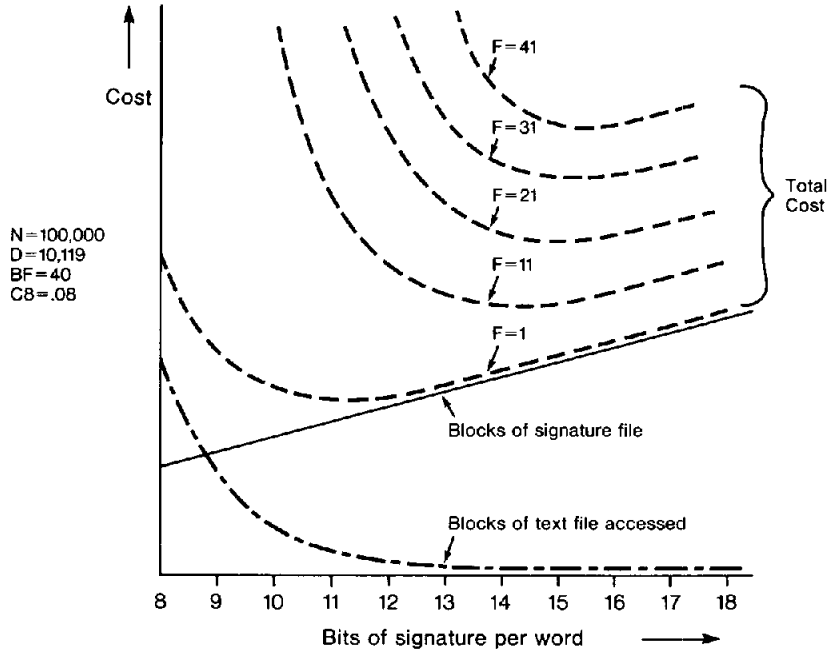

Fig. 1. Average cost, $C = CS + F \times B$, as a function of the signature size of words.

Above, we have analyzed a pure document retrieval environment, that is, messages are only retrieved on the basis of a pattern of words P. Extensions of this analysis to include the case where values for some attributes are also specified in the retrieval have to take into account the average selectivities of the attributes specified [4, 10].

To conclude this section, we mention that this file organization requires less storage than word-indexing methods, makes efficient use of small degrees of multiprogramming, uses just one pass of the signature file in order to retrieve pointers to documents even when many words are used in patterns for identifying documents, and finally, avoids merging pointers. Moreover, insertion of new documents is very easy (just append the signature of the new document in the existing signature file), while if indexes are used the index has to be updated for every word of the new document. More detailed performance comparisons with a word-indexing method can be done by using eq. (5).

6. IMPLEMENTATION

We are implementing a message-filing facility as discussed in this paper based on UNIX. The access method based on signatures has been implemented and tested. The user can retrieve documents based on patterns of

(1) A single word,
(2) A sequence of words,
(3) All the words of a given set of words, and
(4) One or more words of a given set of words.

Presently, combinations of these patterns are not allowed.

The signature file is created after the elimination of a set of 100 common words. In this implementation, signature file and text file reside on the same device. The facility will be enhanced with statistics-gathering mechanisms for studying its performance. We are going to integrate this facility into a message management system.

We envision the facility in three different environments.

(1) In the first environment, we assume a workstation running UNIX (e.g., ONYX, Z-lab, LSI 11/24, with a small 20-Mbyte disk). Suppose that we can devote up to 10 Mbytes of the disk for message filing. We feel that we should be able to store on the order of 10,000 messages, each of less than a page length on the average. The messages can be organized, for example, in approximately 20 different logical files which will hold about 500 letters each. The signature of such logical files will be approximately 50 kbytes which can be searched quite fast even by a workstation processor.

(2) In the second environment, we assume a message file server operating on a high-power microprocessor (e.g., M68000-based system, like APOLLO, with a 300-Mbyte disk). This facility can serve approximately 20 users connected to the server via a local network. Each user could store on the average 10,000 messages, with similar organization and response as in the first environment. The total capacity of the message file server is on the order of 200,000 messages, which is probably adequate for a department or groups of persons.

(3) In the third environment, we envision a high-power microprocessor with a 300-Mbyte disk storing only the signatures. In such an environment, we can go up to millions of messages. The messages themselves are stored on microfilm using an addressable reader printer (e.g., KODAK). In such an environment, the user specifies his request over a network and gets the associated messages using a telefax facility. Such an environment will probably be well suited for organizational archives of messages.

Notice that in all three cases, the software for searching for the messages is the same. We can see, therefore, a nice combination of facilities or a natural transition from one environment to the other.

We should also mention that if some messages are not in machine-readable form they can still be read using OCR, which is probably centralized. In such a case, messages can go as they came, on, say, microfilm, while their signatures are produced and retained for searching purposes. We should also point out that since the problem of document retrieval is reduced to the membership problem of regular expression, the whole technique can be cast into VLSI [8].

Finally, we believe that this technique is already in use in existing information retrieval systems, but it is not often studied as a serious alternative to indexing methods. Sequential scan may be linear, but we feel it is very appropriate in many cases of document retrieval.

REFERENCES

1. AHO, A., HOPCROFT, J., AND ULLMAN, J. *The Design and Analysis of Computer Algorithms.* Addison Wesley, Reading, Mass., 1974.
2. BOURNE, C. AND FORD, D. A study of statistics of letters in English words. *Inf. Control 4* (1961), 48–67.

3. CARDENAS, A. Analysis and performance of inverted data base structures. *Commun. ACM 18*, 5 (May 1975), 253–263.
4. CHRISTODOULAKIS, C. Estimating Selectivities in Data Bases. Ph.D. dissertation (Tech Rep. CSRG 136), Univ. of Toronto, Toronto, Canada, 1981.
5. CHRISTODOULAKIS, C. Estimating block selectivities. Submitted for publication.
6. CHRISTODOULAKIS, C. Implications of certain assumptions in data base performance evaluation. Submitted for publication.
7. DEWEY, C. *Relative Frequency of English Speech Sounds.* Harvard University Press, Cambridge, Mass., 1950.
8. FLOYD, R. AND ULLMAN, J. The compilation of regular expressions into integrated circuits. In *Proc. 21st Symp. on Foundations of Computer Science* (Oct. 1980), pp. 260–269.
9. GRIGNETTI, M. A note on the entropy of words in printed English. *Inf. Control 7*, (1964), 304–306.
10. HAMMER, M. AND CHAN, A. Index selection in a self-adaptive data base management system. In *Proc. Int. Conf. on Management of Data.* (Washington, D.C., June 2–4, 1976) pp. 1–8.
11. HASKIN, R. Special purpose processors for text retrieval. *Database Engineering 4*, 1 (Sept. 1981), 16–29.
12. KIRSTEIN, P. New text and message services. In *Proc. IFIP Congress 1980* (Tokyo, Melbourne).
13. PAPOULIS, A. *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, New York, 1965.
14. RITCHIE, M. AND THOMPSON, K. The UNIX time-sharing system. *Commun. ACM 17*, 7 (July 1974), 365–375.
15. SALTON, G. AND WONG, A. Generation and search of clustered files, *ACM Trans. Database Syst. 3*, 4 (Dec. 1978), 321–346.
16. SHANNON, G. Prediction and entropy of printed English, *Bell Syst. Tech. J. 30*, 1 (1951), 50–64.
17. YAO, S.B. Approximating block accesses in database organizations. *Commun. ACM 20*, 4 (April 1977), 260–261.
18. ZIPF, G.K. *Human Behavior and Principle of Least Effort.* Addison-Wesley, Reading, Mass., 1949.

BIBLIOGRAPHY

1. AHO, A., KESNIGHAM, B., AND WEIBERGER, P. Awk-A pattern scanning and processing language. 2nd ed. Bell Laboratories, Murray Hill, N.J., Sept. 1978.
1a. CHRISTODOULAKIS, C. Estimating record selectivities. *Inf. Syst. 8*, 2, 1983 (to appear).
2. OROSZ, G. AND TAKACS, L. Some probability problems concerning the making of codes into superposition field. *J. Doc. 12*, 4 (Dec. 1956), 231–234.
3. SEVCIK, K. Data base system performance prediction using an analytical model. In *Proc. Conf. on Very Large Data Bases* (Cannes, France, Sept. 9–11), ACM, New York, 1981 pp. 182–198.
4. SLONIM, J., MACRAE, L., MENNIE, L., DIAMOND, N. NDX-100: An electronic filing machine for the office of the future. *Computer* (May 1981), 24–36.
5. STIASSNY, S. Mathematical analysis of various superimposed coding methods. *American Documentation 11* (Feb. 1960), 155–169.
6. ROBERTS, C. Partial-match retrieval via the method of superimposed codes. *Proc. IEEE 67*, 12 (Dec. 1979), 1624–1642.
7. TSICHRITZIS, D. Integrated data base and message systems. Submitted for publication, 1981.
8. TSICHRITZIS, D. OFS: An intregrated form management system. In *Proc. Conf. on Very Large Data Bases* (Montreal, Canada, Oct. 1–3), ACM, New York, 1980 pp. 161–166.
9. TSICHRITZIS, D. AND LOCHOVSKY, F. *Data Models.* Prentice-Hall, Englewood Cliffs, N.J., 1981.

Received January 1982; revised September 1982; accepted September 1982