A SOFTWARE TECHNIQUE FOR FAST TEXT SEARCHING BASED ON COARSE INDEXING OF A DATA BASE

Ernst J. Schuegraf
Department of Mathematics and Computing Sciences
St. Francis Xavier University
Antigonish, Nova Scotia. B2G 1C0. Canada

ABSTRACT

Searching large textual data bases with a computer can be achieved by indexing, which generally results in a large index-term vocabulary. A new technique, based on the idea of coarse indexing is described which uses a vocabulary of character strings for indexing. These string-index-terms are used to calculate a block address at which a particular record should be stored. This divides the data base into a number of blocks, which may be chosen to match constraints on performance or data base size. Queries are processed in a similar fashion to the record. Addresses of blocks which cannot contain a hit can be determined. Searching can thus be restricted to a fraction of the original data base. This approach is especially suited to searching with associative devices, or other specialized search processors. Some experimental results are given to prove the effectiveness of the method.

UNE TECHNIQUE LOGICIELLE POUR LA RECHERCHE TEXTUELLE RAPIDE BASEE SUR L'INDEXATION SOMMAIRE D'UNE BASE DE DONNEES.

RESUME

On peut interroger des bases de données volumineuses sur un ordinateur en les indexant, ce qui nécessite habituellement un énorme dictionnaire de descripteurs. L'auteur décrit une nouvelle technique basée sur le principe d'une indexation sommaire utilisant un vocabulaire constitué de chaînes de caractères. Ces descripteurs-chaînes sont utilisés pour calculer l'adresse du bloc où doit être stocké chaque dossier. Cette technique permet de diviser la base de données en une série de blocs qui peuvent être ensuite utilisés pour comparer les entraves à la performance ou au volume de la base de données. Les recherches sont traitées de la même façon que le dossier. Les adresses des blocs qui ne peuvent stocker des réponses pertinentes peuvent être identifiées. Par conséquent, il devient possible de restreindre la recherche à une partie seulement de la base de données intégrale. Cette technique est particulièrement utile pour les recherches utilisant des unités associatives ou toute autre unité de traitement spécialisée. L'auteur présente quelques résultats d'expériences effectuées pour démontrer l'efficacité de la technique.

SEARCHING BASED ON COARSE INDEXING

INTRODUCTION AND PROBLEM STATEMENTS

The role of the digital computer has changed dramatically since its beginning. From a tool designed to assist the scientist and engineer in his calculations, it has invaded other areas and is now considered a general information processing device. It is employed with great success in library science, the humanities and the business world in addition to its traditional functions in science. Common to applications in the non-science disciplines is the need for storage of large amounts of textual data with quick access to individual records.

The last few years have seen considerable progress in the problem area of storage and accessing of large data bases, as there is much new technology for auxiliary storage devices. Magnetic disks have become faster and their capacity has expanded tenfold or more. However, this expansion was spurred by the large increase in the size of data bases. Storage requirements for on-line data bases seem to be always a small step ahead of available capacity. There is still potential for developing disks with larger storage capacities, but great improvements in the speed of access cannot be expected to come from hardware developments.

It is the organization and structure of a data base that have the most profound effect on the access speed. Choosing a data base organization that is especially suited for the operations most frequently carried out can produce great gains in access speed. In retrieval systems that utilize large textual data bases, as they occur in document retrieval and library applications, the operations of searching and retrieving are the most common. Quick access to the desired records is of utmost importance, especially if the system operates on-line. Patience of on-line users is a valuable commodity and should not be abused.

Provision of fast access to data base records has been an issue that has been well studied (London, 1973). Two different approaches have been taken to solve the problem, namely, by software and hardware.

SOFTWARE SOLUTIONS

One approach is to employ software techniques and extensive data manipulation prior to searching. It involves indexing of records by a set of index terms and creation of auxiliary files. The index term set may be restricted, or may be unrestricted and include every word in the data base. The set of index terms is normally stored in a file called the "dictionary." Another auxiliary file is constructed, normally an index or tree-structured file, that -- for every index term -- stores pointers to the records which contain that particular term. As this auxiliary file more-or-less duplicates the data in the original file, storage requirements are generally of the same order of magnitude. In fact, they may even be larger than for the original file (Cardenas, 1975), and special problems arise from maintaining it.

To retrieve specific records, a user's query is analyzed, the auxiliary files accessed and manipulated to produce a list of record numbers. These records are hits or potential hits for satisfying the user's query. The latter is a function of the operators in the query, as it is well-known (Heaps, 1978) that proximity operators in the query necessitate a character-by-character examination of the record. In this case determination of a hit is delayed until each candidate record has been examined for the proximity conditions.

The software approach provides fast access to desired records, especially with complex queries. Indexing serves as the link between content and physical location of the record. However, the disadvantages are serious:

- large storage requirements for dictionary and auxiliary files
- difficult maintenance of auxiliary files
- large amounts of computer time spent on preprocessing, and setting up of auxiliary files
- not all queries can be handled by indexing and auxiliary files.

HARDWARE SOLUTIONS

One approach in providing specialized hardware for improved access to textual data bases has been the suggestion of employing a backend-processor for this task (Canaday, 1974). A small processor attached to auxiliary storage devices, but connected to the main processor is responsible for carrying out the functions of data base access and retrieval. Auxiliary files are still used, but the backend-processor carries out all the file manipulations. The main processor initiates only the search by the backend-processor and receives the results, thus freeing its capabilities for other activities.

The shift of data base access and retrieval functions from the main to the backend-processor has progressed logically by suggestions to attach processing logic to disk drives. These "intelligent" peripherals have become known as Logic-per-track devices (Slotnick, 1970; Parker, 1971). If the hardware attached to the devices operates associatively, these devices are known as associative file stores. Several associative file stores have been reported in the literature, namely, RAPID (Parhami, 1972), RAP, developed by the University of Toronto (Ozkarahan, 1975), and ICL's commercially available CAFS system (Maller, 1979). Preprocessing of the data base and auxiliary files may be eliminated with intelligent peripherals.

For information retrieval applications there is a group of specialized text processors (Hollaar,1979), which view the data base as a collection of variable-length records of character strings. Query formats, generally, can be quite complex as the data base is normally subjected to a full sequential search carried out in parallel and quite often associatively. Two systems are available on the open market; OSI's Associative File Processor (Bird, 1979), and its successor, the High Speed Text Searching System (HSTS) (Moore,

Michels, 1980). The HSTS system incorporates special hardware as well as new software techniques for searching (Aho, 1975), and can search one million characters per second. A newly announced product (Electronics, 1982), the data base engine of Textarcana claims to be capable of searching eight million characters per second.

In these systems conventional indexing of records is unnecessary, auxiliary files are redundant and the data base can grow without significant constraints. However, for large data bases, it is still advisable to do some grouping of data base records, as it will reduce retrieval times even further. The groups can be generated by a fairly simple method which is becoming known as coarse or shallow indexing (Schuegraf, 1980).

Coarse Indexing

The basic function of coarse indexing is to provide a grouping of "similar" records in a cell -- or scan block -- of an intelligent peripheral. Similarity between records can be defined in various ways, but should be tailored to the application. However, there is subtle difference between conventional and coarse indexing. The former indicates where desired records are to be found, and the latter identifies blocks which cannot contain possible hits.

To retrieve the desired records, all cells not specifically excluded by coarse indexing are accessed. All records in the cell are examined and if they meet the selection criteria are passed on for further processing. Intelligent peripherals can, in effect, search at disk-transfer rates; thus searching a block of similar records does not require much more effort than accessing it.

In a pilot implementation of a telephone directory enquiry system with CAFS, shallow indexing was employed, based on the first four characters of the fields of a subscriber's record. A small index file establishing a link between the cells and the "index terms" was stored. For compound queries the indexes could be manipulated prior to searching in order to reduce the number of cells which had to be accessed.

Coarse indexing has also been suggested for use in an associative file store (Schuegraf, Lea, 1981). The major improvement which is apparent is the elimination of secondary indexes. Cells that have to be accessed and searched can be determined algorithmically rather than by index lookups. The software technique employed can also be used for fast searches of a data base by eliminating certain blocks from a search. Obviously, the method works best in connection with intelligent peripherals.

SEARCHING BASED ON COARSE INDEXING

The new method to be described can be divided into three parts: Data base sectioning based on coarse indexing; allocation of records to scan blocks; and query processing.

Data Base Sectioning

The data base must be divided into a number of equally-sized scan blocks. The number of scan blocks is determined by the size of the data base and constraints on response time, but it should be a convenient power of two. Coarse indexing determines which record is allocated to which scan block. The elements used for coarse indexing are variable length character strings called fragments or n-grams (Clare, 1972). Fragments have been found to be suitable elements for data base compression (Schuegraf, Heaps, 1974; Barton, 1974), and for indexing (Schuegraf, Heaps, 1976). A set of fragments chosen by one of many available algorithms (Clare, 1972; Schuegraf, Heaps, 1973; McCarthy, 1974; Wagner, 1973) from a data base sample form a fragment dictionary. If the character set of the data base is included in the dictionary all data base records can be represented by concatenation of dictionary elements. A subset of a dictionary generated from an INSPEC data base is shown in Figure 1.

ET	IAL	IM
EX	IB	IN
F	IC	INC
FE	ICAL	INE
FI	ID	ING
FORM	IES	ING-
G	IF	ING-THE
GE	IGH	INT
H	IGN	INTER
I	IL	ION

Figure 1: Partial listing of a word fragment dictionary.

When inspecting the dictionary it is obvious that single characters and certain character strings such as "AND-THE", "FOR-THE" do not have any content bearing. The subset of fragments in the dictionary suitable for indexing are the content- or index fragments only. The underlined fragments in Figure 1 are considered index fragments.

Dictionary size can be limited to a size convenient for the system used. The set of index fragments can be chosen by inspecting the dictionary or can be selected automatically by using its frequency of occurrence in the data base as an indicator. Elimination of high- and low-frequency dictionary

fragments produces an approximately equi-frequent set suitable for indexing. This result is consistent with the demands placed on good index terms (Salton, 1975).

Record Allocation to Scan Blocks

To allocate a record to a specific scan block it is necessary to determine all content fragments in the record. This is somewhat cumbersome, but it has to be done only once. It can be derived as a by-product of text compression with fragments and can benefit from the use of associative hardware (Lea, 1978). Once all content fragments have been determined a bit map of content fragments is generated for each record, which, for obvious similarities, can be called a "signature" (Harrison, 1971). If each content fragment is assigned a specific position in the bit map, the presence or absence of the fragment can be indicated by a single bit. Human signatures characterize and identify the writer, but a record signature is an indication of the record's content. An example of a record and its signature is shown in Figure 2.

RECORD: CLUSTERING METHODOLOGIES IN EXPLORATORY DATA ANALYSIS

FRAGMENTED
RECORD: CL/U/STE/R/ING-/METH/O/DO/LOGI/ES-/IN-/EX/P/LOR/AT/O/RY-
/AN/A/LY/S/IS

COMPRESSED
RECORD: 16/241/228/185/65/123/148/24/85/31/62/36/162/91/7/148/191
/21/2/1/5/2/102/218/77

RECORD
SIGNATURE: 1 0 . . 0 1 0 0 ... 0 1 0 ... 0 1 1 1 ... 0 1 0
... 0 1 0 0

NOTE: FOR CLARITY ONLY SOME INDEX FRAGMENTS
ARE SHOWN IN THE SIGNATURE

Figure 2: Record and Sample Signature

The signature could now be interpreted as the address of a scan block, but it is too long. If there are s-bits in the signature and m-bits in the address of a scan block, then it is necessary to map the s-bits into the m-bits of the address. The signature mapping eliminates the need for secondary indexes, but its operation is crucial to the performance of the entire system. It is not possible to choose an arbitrary mathematical function; there are restrictions (Hodes, Feldmann, 1978).

Restrictions which must be placed on the mapping function can be summarized as:

R1. The same bit position in the signature must always be mapped to the same position in the scan block address;

R2. A one-bit in a signature position must always be mapped to a one-bit in its corresponding address position;

R3. All scan block addresses should be generated from the set of signatures with approximately the same frequency.

The last two restrictions can be implemented with relative ease, but finding which positions in the signature should be mapped onto the same address bit is difficult. If signature positions are mapped arbitrarily into address bits, there is a tendency that the addresses generated will contain mostly ones. This is a consequence of the second restriction, but conflicts with the third. For this reason it is necessary to map content fragments that frequently co-occur in a record into the same address bit. This represents a form of clustering. All content fragments in one cluster are mapped into the same address bit. The number of clusters is fixed by the number of bits in the scan block address. Clustering content fragments, in fact, establishes "similarity" between records. Even though clustering is a computationally expensive operation, methods available for cluster generation (Dubes, Kain, 1980) provide control over cluster size and number of clusters. Control over cluster size is essential and can be used to satisfy the third restriction placed on the mapping function.

Query Processing

For each term in the query a signature and scan block address is generated. Logic connections between query terms are processed by applying the logic to the scan block address. NOT operators must be ignored. The set of scan block addresses resulting from processing the query determines the scan blocks to be searched.

A one-bit in a certain bit position in the scan block address indicates the presence of a content fragment from a cluster. The addresses generated by the query processing will have ones in certain positions, implying that elements from corresponding clusters must be present in the records of that scan block. All scan blocks which have a zero in those address positions can be excluded from the search, because they do not contain an element from that cluster. This fact is a consequence of restriction R2 and reduces the fraction of the data base that must be searched considerably. If a single-term query produces an address with a single one-bit, only half the blocks must be searched; if there are two one-bits, only a quarter, and so on.

Even with specialized hardware such a reduction in the number of scan blocks to be accessed is significant. As accessed scan blocks may contain possible hits, each record must be checked if it satisfies the query. Hits can be output to the user as the search progresses, thus reducing response times.

Experimental Results

A data base of 1537 records from an INSPEC tape with a total of 102,600 characters was used in the experiment. A dictionary of 256 text fragments was analyzed and 149 fragments were thought to be suitable index fragments. A record signature was thus a bit vector with 149 bit positions. It was decided to use 256 scan blocks for sectioning the data base as this resulted in a convenient 8-bit address. The mapping function had to map the 149 bits of the signature to the 8-bits of a scan-block address subject to restrictions R1, R2, R3, as previously mentioned.

The mapping was accomplished in two stages. A clustering algorithm (Gotlieb, Kumar, 1968) was applied to the index fragments. As the original measure described in the paper did not prove to be suitable, the strength of association between index fragments was measured by the standard correlation coefficient. Twenty-four clusters of index fragments were generated. The clusters had widely-varying frequencies of occurrence of index fragments. These 24 clusters were then assigned to one of the positions in the scan-block address. The goal of this mapping was to achieve a more even distribution of index fragments assigned to specific positions.

After the mapping function had been established, all record signatures were generated, signatures mapped to scan-block addresses and records inserted in that block. A more detailed description of the experimental procedure is currently in preparation (Schuegraf, Lea, 198?).

To test the effectiveness of the coarse indexing search technique a total of 32 queries were made up and processed. The sequence of scan blocks that had to be accessed was generated by increasing the Hamming distance from the scan-block address generated by the query. A typical query is shown in Figure 3.

Query: (THICK OR THIN) AND FILM AND RESISTOR

Query Term	SCAN BLOCK ADDRESS FOR TERM
THICK	00000000
THIN	00000000
FILM	00100000
RESISTOR	01000000
SCAN BLOCK ADDRESS FOR QUERY	01100000

Results: 64 Buckets had to be searched
768 Records had to be examined
6 Hits were found.

Figure 3: Typical Query

A brief summary of the experimental results is provided in Table 1.

#Query Terms	#Queries	Buckets Searched			Records Searched			# Hits		
		MIN	MAX	AVG	MIN	MAX	AVG	MIN	MAX	AVG
1	5	32	120	90	483	1057	860	1	59	26
2	4	32	125	82	526	1214	887	5	22	9
3	13	16	120	70	327	1050	715	2	39	10
4	8	31	125	66	451	1214	739	6	24	12
5	2	110	177	143	1106	1339	1222	5	12	8

Table 1. Summary of Query Simulation

The results show that the number of records that must be searched is a fraction of the original data base, but larger than the fraction represented by the number of buckets that must be searched. This is due to the uneven distribution of records in scan blocks. The most significant reductions were achieved by queries which contained several AND conditions rather than OR's. Work is currently in progress which studies, the effect of, the dictionary, the clustering algorithm and the mapping function on the performance of this technique.

Conclusion

From the results presented it is apparent that the method is effective in reducing the amount of data to be searched. Its major advantage over similar methods described in the literature (Hickey, 1977; Roberts, 1979) is the elimination of special signature files which must be searched, reduces the number of blocks that must be searched significantly. Specialized peripherals, especially associative file stores, are particularly suited to this technique.

Acknowledgements

This work was supported by a grant from the National Sciences and Engineering Research Council of Canada. The discussions with Mike Lea of Brunel University and Peter Willett of Sheffield University are gratefully acknowledged, as well as the programming assistance provided by Bonnie Quinn.

REFERENCES

AHO, A.V. et CORASICK, M. J. "Efficient String matching - An Aid to Bibliographic Search," in Comm ACM, Vol. 18, No. 6, (June 1975), pp. 333-340.

BARTON, Ian J., CREASY, Susan E., LYNCH, Michael F., SNELL, Michael J. "An information theoretic approach to text searching in direct Access Systems," in Comm. ACM, Vol. 17, No. 6, (June 1974), pp. 345-351.

BIRD, R. M. "The Associative File Processor: A Special Purpose Hardware System for Text Search and Retrieval," in Proceedings of the IEEE 1979 National Aerospace and Electronics Conference, Long Beach IEEE 1979, pp. 433-439.

CANADAY, R. H.; HARRISON, R. D., IVIE, E. L.; RYDER, J. L.; WEHR, L. A. "A backend computer for data base management," in Comm. ACM, Vol. 17, No. 1, (January 1974), pp. 53-56.

CARDENAS, A. F. "Analysis and performance of inverted data base structures," in Comm. ACM, Vol. 18, No. 5, (May 1975), pp. 253-263.

CLARE, A. C.; COOK, E. M.; LYNCH, M. F. "The identification of variable length, equifrequent, character strings in a natural language data base." Computer Journal, Vol. 15, No. 3, (August 1972), pp. 259-262.

DUBES, Richard; JAIN, A. K. "Clustering Methodologies in Exploratory Data Analysis" in "Advances in Computers" Vol. 19, (1980), pp. 113-228.

Electronics, Product Description, April 7, 1982, p. 176.

GOTLIEB, C. C.; KUMAR, S. "Semantic Clustering of Index Terms" in Journal of the ACM, Vol. 15, No. 4, (October 1968), pp. 493-513.

HARRISON, Michael C. "Implementation of the substring test by hashing" in Comm. ACM, Vol. 14, No. 12, (December 1971), pp. 777-779.

HEAPS, H. S. "Information Retrieval: Computational and Theoretical Aspects." New York: Academic Press, 1978.

HICKEY, Thomas. "Searching linear files on-line" in On-line Review, Vol. 1, No. 1 (1977), pp. 53-58.

HODES, Louis; FELDMANN, Alfred. "An efficient design for chemical structure searching. II. The File Organization," in Journal of Chemical Information and Computer Science, Vol. 18, No. 3, (September 1978), pp. 96-99.

HOLLAAR, Lee. "Text Retrieval Computers" in Computer, Vol. 12, No. 1, (March 1979), pp. 40-50.

LONDON, Keith R. "Techniques for direct access" Auerbach Publ., Philadelphia, 1973.

MALLER, V. A. J. "The content-addressable file store -- CAFS" In Internat. Computers Ltd. Technical Journal, November 1979, pp. 265-279.

McCARTHY, J. P. "Automatic File Compression," in Proceedings of the International Computing Symposium, North Holland, 1973, pp. 511-516.

MOORE, G. B.; MICHELS, L. S. "OSI's High Speed Text Search System: A hardware approach to full text searching." In Proceedings ASIS Annual Meeting, Vol. 17, 1980, Knowledge Industries Ltd., pp. 335-337.

OZKARAHAN, E.; SCHUSTER, S. A.; SMITH, K. C. "RAP: An associative processor for data base management." In Proceedings ACM Natl. Conference, Vol. 44, (1975), pp. 379-387, AFIPS Press.

PARHAMI, B. "A highly parallel computer system for information retrieval" in Proceedings Fall Joint Computer Conference, Vol. 41, 1972, pp. 681-690, AFIPS Press.

PARKER, I. L. "A logic-per-track retrieval system." Proceed. IFIPS Congress, North Holland Publ., pp. 146-150, 1971.

ROBERTS, Charles. "Partial-match retrieval via the method of superimposed codes" in Proceedings IEEE, Vol. 67, No. 12, (December 1979), pp. 1624-1642.

SALTON, Gerard A. "A Theory of Indexing." Philadelphia, Society for Industrial and Applied Mathematics, 1975.

SCHUEGRAF, E. J.; HEAPS, H. S. "Selection of Equifrequent Word Fragments for Information Retrieval," Information Storage and Retrieval, 1973, Vol. 9, No. 12, (December 1973), pp. 697-711.

_____ "A comparison of algorithms for Data Base Compression by use of Fragments as Language Elements." Information Storage and Retrieval, Vol. 10, No. 9, (September 1974), pp. 309-319.

_____ "Query Processing in a retrospective document retrieval system that uses word fragments as language elements." Information Processing and Management, Vol. 12, No. 4, (August 1976), pp. 283-292.

SCHUEGRAF, E. J. "Indexing for Associative Processing." Canadian Journal of Information Science, Vol. 5, (May 1980), pp. 93-101.

SCHUEGRAF, E. J.; LEA, R. M. "An associative file store using fragments for run-time indexing and compression," in Proceedings on Future Directions in Information Retrieval, Butterworths, 1981, pp. 321-335.

_____ "A proposal for an associative file store with run-time indexing. PART II: Experimental Evaluation of Software Algorithms" (In preparation, 198?).

SLOTNICK, D. L. "Logic-per-track devices." In Advances in Computers, Vol. 10, 1970, Academic Press, pp. 291-296.

WAGNER, R. A. "Common phrases and minimum-space text storage." In Comm. ACM, Vol. 16, No. 3, (March 1973), pp. 148-152.