Connections

CONTENTS

Connections
Exchange of Container Vessel Stowage Information Using EDIFACT ........................................ 7
Ulrich Spinde and Dirk Steenkolen
Measuring Electronic Data Interchange in Primary Care
COPA: Communication Project Apeldoorn ................................................................. 9
Peter J. Branger and Joppe Duisterkoudt
Computer Aided Teleconferencing: Bridging the Training Gap
VA Develops Innovative Training Technique ........................................................... 13
Vincent L. Fiscella and Sherry H. Buell
Hear MUMPS Talk ............................................................................................... 17
Alfred Garcia and Robert Kirsch

State of the Art
The Design of a Text-Based Information Storage and Retrieval System in MUMPS .................. 21
Kevin C. O'Kane
System Protection Techniques within the Hospital Information System at the Hospitals of the University of Würzburg ............................................................. 27
Dr. Wolfgang Doekelstol

Medical Informatics
Medical Information, Knowledge, and Expert Systems .............................................. 37
Bruce L. Blum and Ralph D. Sommel

Opinions
The 20th MUG NA Annual Meeting: Renaissance for MUMPS, Some Personal Remarks ........ 41
Wolfgang Giere, M.D.
Re: Harvest Time or New Spring? ........................................................................ 44
Gretchen Bradfield and Ed de Moel
Summertime ........................................................................................................ 49
W. Lloyd Milligan, Ph. D.

Senior Editor: Ruth E. Dayhoff, M.D.
Managing Editor: Terry Hess
Production: Dianne Barker, Ann Sprague
Cover Design: Christopher J. Brannigan
Administrative Staff: Maureen Lilly, Sandra Reynolds, Maureen Hagon, Judith Brown, Lyn Brown, Faith Hall, Ed Hudson, Peggy Hoffman, Angela Hooker, Joy Moses, Patrish Mint, Joanna Panteukin, Don Picone, Jean Sheppard

Copyright 1991 by the MUMPS Users' Group, College Park, MD.
MUMPS is a registered trademark of Massachusetts General Hospital.
State of the Art

The Design of a Text-Based Information Storage and Retrieval System in MUMPS

Kevin C. O'Kane
Department of Computer Science
The University of Alabama
Tuscaloosa, Alabama 35487

Introduction

In recent years, CD-ROM drives have become more affordable and their use more widespread. Similarly, the number of titles available on CD-ROM disks has grown rapidly. Books, reference libraries, and periodicals are now readily available on CD-ROMs at increasingly lower prices. A typical CD-ROM drive for a PC-class computer retails for about $600 and includes several sample CD-ROM disks. Recently, a mail-order house offered an external PC-compatible CD-ROM package that included six CD-ROM disks. Among the disks were a 21-volume encyclopedia with over 2,000 illustrations; an eight-volume reference library, world and U.S. atlases, a 450-title library and 17 translation dictionaries in 12 different languages. The total price was $609, well within the budget of many home computer owners. As prices continue to drop, the available inventory of disks will grow rapidly as will the number of computers with installed CD-ROM drives.

However, while the inventory of disks available continues to grow and the costs of acquisition decline, there has been relatively little practical work in recent years toward addressing critical questions concerning data access. Presently, most systems for CD-ROM text retrieval are based on brute force indexing techniques or full text search. For most systems, retrieval is accomplished by means of inverted files of manually preassigned keyword indices accessed by software that performs Boolean operations on the citations associated with each reference keyword. Some systems permit text matching and various degrees of wildcard searching. The programming needed to implement a system based on this model is not complex.

Systems based on this model have been in use for many years. These include MEDLARS from the National Library of Medicine, BRS from Bibliographic Retrieval Services, ORBIT from System Development Corporation, and DIALOG from Lockheed Information Systems. Several microcomputer-based systems are available, such as Computer Library from Ziff-Davis Publishing.

One problem with this approach, however, concerns the assignment of index terms (or keywords) to the documents in the database. In many cases a person must read each document and assign to it keywords selected from a controlled list. In some cases, the author of the document provides the keywords. In others, documents are scanned by computers to detect the presence of words from a preselected list. If manual indexing is used, keyword assignment is costly and time consuming. When authors assign keywords to their own works, personal bias tends to inflate the work's relevance. Simple computer scanning for preselected keywords is efficient and inexpensive, but it is unable to weigh the relative importance of a keyword or detect the presence of important words not contained on the preselected keyword list.

Queries in keyword-based systems are normally stated as Boolean expressions such as:

\[(\text{mumps} \& \ (\text{windows} \lor \text{interactive} \lor \text{graphics}) \) \ (\text{user} \lor \text{interface})\]

Each document in the system is represented by a unique index number. Attached to each keyword in the system is a list of all the document numbers in which the word appears. The retrieval program retrieves the sets of document numbers associated with each keyword and performs the intersections (\&), unions (\lor), and other operations stated in the query to produce a final set of document numbers. These numbers are then used to retrieve the source documents. There are many variations, including wild cards, keyword proximity, and other techniques which improve retrieval effectiveness.

These systems are effective when the keyword vocabulary is precise and well-controlled. For example, in medicine there is a well-defined keyword vocabulary and a high degree of precision of meaning. In other disciplines, the nomenclature is not as well controlled. Computing, for example, has a constantly evolving terminology. Many terms are composite, built from smaller terms which individually may convey little meaning (e.g. database systems, artificial intelligence, expert systems, object-oriented programming). Many concepts are referred to in more than one way. Boolean keyword-based retrieval techniques are less effective when the keyword vocabulary permits multiple alternative terms.

Further, when the results of the search are presented to the user, there is generally no ranking of the documents to indicate which is more or less likely to satisfy the query. All documents are presented as equally relevant. This often leads the user to formulate increasingly complex queries in order to reduce the size of the document set retrieved so as to minimize subsequent inspection or print-out of the results. As the queries become more restrictive, there is an increase in the probability that important documents will be lost.
Indexing

The goal of automatic indexing is to develop computer algorithms to replace the manual arts of indexing, cataloging, classification, or abstracting. There are several approaches to this problem but the most widely investigated are based on statistical techniques. The fundamental hypothesis of these systems is that frequency of word usage in documents is an indicator of document content. Salton [1] quotes a statement made in an article in 1958 by D.L. Smith, one of the earliest advocates of this approach, as follows:

The justification of measuring word significance by word frequency is based on the fact that a writer normally repeats certain words as he advances or varies his arguments and as he elaborates on an aspect of a subject. This means of emphasis is taken as an indicator of significance...

A main objective of automatic indexing techniques is to translate both queries and documents into a common, intermediate representation and to develop measurement techniques to calculate the coefficients of similarity between the queries and the documents. Documents are ranked according to their computed similarity to the query, and presented to the user in rank order with the most similar presented first and the least similar last.

One way of viewing this process, suggested by Salton [2], involves viewing the documents and queries as points in an n-dimensional hyperspace (see figure 1). In this view, the similarity coefficient is calculated as the distance between a point represented by the query and the points represented by documents. Those documents which lie within a user-determined distance of the query (a hypersphere) are retrieved with the documents whose points are closest to the query present first, and those furthest away displayed last. Documents outside the sphere are considered irrelevant to the query although the dimension of the sphere can be enlarged or contracted to increase or decrease recall.

In this model, the documents are represented by vectors. Each element of the vector represents the presence of one of the indexing terms (keywords) and the weight or importance of the term in the document. The axes of the hyperspace are the terms. The weight of a term in a document determines its position on an axis. The point representing a query or document is thus its position in the space based on term usage in the document. In figure 1, a three-dimensional space is depicted. In practice, there are usually several hundred possible terms in the vocabulary of a given document collection.

Table 1 gives several well-known similarity measures as listed in reference 1. These essentially measure the degree to which document and query vectors are similar to one another. They are based on the number of words that co-occur between vectors and the relative weights of the terms in the vectors. The term $Freq_{ij}$ is the frequency of occurrence or weight of term $i$ in document $j$. Documents with a large number of highly weighted terms in common with one another will have high coefficients of similarity. Likewise, documents with few terms in common will have low coefficients. For more detailed discussions, see [1,2,3,4,5].

![Figure 1: Query Vector in Document Hyperspace.](image)

Table 1: Similarity Measures (See Salton and McGill, Introduction to Modern Information Retrieval, page 201 (4)).

Basic Indexing Procedures

In a simple model, terms are selected by scanning the document collection. Each term in each document is compared to the terms in a stop list. The stop list contains those terms which convey no meaning (e.g., a, the, but, for, and so forth). Terms from documents that are contained in the stop list are ignored. Prefixes and suffixes are removed and the terms are truncated into stems. The number of occurrences of each term in the collection as a whole and in each document is recorded.

The terms are then reevaluated to determine their suitability as indexing terms. Those terms of high frequency and breadth, uniform distribution are discarded as weak indicators of content. Terms of very low frequency are also discarded. The remaining terms—those of middle frequency—are retained. Terms are ranked according to their ability to discriminate between documents. Those terms whose distribution tends to be clustered are...
used above those whose distribution is more uniform across document collection.

The document vectors are re-analyzed in light of the term discrimination coefficients and a weight for each term in each document is assigned. A term with a high discrimination coefficient that occurs many times in a document is given a high weight while a term with a low discrimination coefficient and a low frequency is given a low weight. Thus a set of weighted document vectors is constructed. Queries are processed in much the same manner.

There are many refinements to the above procedure. Among these are the construction of term hierarchies. These can be calculated by building a term-term correlation matrix. The columns and rows of the matrix represent individual terms. The entries in the matrix indicate the number of times the terms co-occur with one another in documents in the collection. Terms with a high degree of correlation have a high degree of probability of being related to one another. Document vectors and query vectors can be augmented with closely related terms in order to enhance retrieval effectiveness. Term hierarchies can be hierarchically organized. [1,2]

Retrieval

When a query is put to the system, it calculates the similarity coefficient between the documents and the query using one of the similarity formulas. Rather than calculate all similarity coefficients, only those documents with terms contained in the query need to be evaluated. This is done by means of an inverted document-term matrix in which— for each term— each document in which it occurs is recorded.

![Figure 2: Document Clusters](image)

Another technique involves clustering the documents [6]; that is, identifying in advance which documents are related to one another. This is illustrated in figure 2. For each cluster, a centroid vector is calculated. This vector is an average (or center of gravity) of the document vectors in the cluster. Hierarchies of clusters can be constructed. During retrieval, the query vector is compared initially only with cluster centroid vectors. Upon selecting the centroid vectors that are close to the query vector, the individual documents from the clusters can be displayed. Using centroid vectors reduces the amount of time required to answer a query since each centroid vector represents several document vectors. In these systems where centroid vectors are hierarchically clustered, the user has the option during retrieval to move up or down the cluster hierarchies to capture more specific or general references.

Programming Languages

Automatic indexing systems have largely been experimental and confined to research laboratories. Until quite recently—with the advent of powerful desktop computers with large disk drives and the low cost availability of text databases on CD-ROM—most work dealt only with relatively small collections of documents. Although there are experiments to be effective [14], they were too expensive in terms of computing resources to justify widespread use.

Although cost has been a major factor in the widespread use of automatic indexing, another factor concerns the complexity of the programming required to build such a system. There have been implementations in FORTRAN, PASCAL, PL/I, and other high-level languages. These high-level languages lack features that facilitate the type of programming needed to implement automatic indexing algorithms. The indexing and retrieval procedures outlined above require extensive string manipulation and the construction of very large, fixed-sized data structures which can be simulated only with difficulty in ordinary record-oriented file systems.

To this end, the MUMPS language is ideally suited to information storage and retrieval applications. Its built-in string handling, flexible data types and global array-based database greatly simplify implementation of the algorithms outlined above.

In order to demonstrate this, a demonstration automatic indexing and retrieval system was written in MUMPS, the details of which are given below. The flexibility of MUMPS with regard to file structure made this task vastly simpler than implementing the same procedures in some other high-level language such as C, PASCAL or PL/I. In many cases, modules which would have required several hundred lines of code in other languages, were reduced in size to only 30 to 50 lines. MUMPS is a language in which it is easy to express many of the complex structures required by information retrieval programming. This makes it possible to undertake more challenging applications than would be the case in other languages.

A MUMPS-based Information System

The overall flow chart for the system is given in two parts. The first part, shown in figure 3, concerns reading and indexing the documents. The second part, given in figure 4, refers to the retrieval functions. Figure 5 gives an overview of global array processing and the corresponding calculations associated with them. The system was tested using a collection of documents concerning computer science. Each document consists of a title, reference information, and an abstract of approximate 25 lines. Altogether, there were 5,614 documents with 120,479 word occurrences of which—not counting stop list words—10,206 words were unique with an average frequency of one per word of 14.
The title and each line of the text are decomposed into individual words. Each word is checked to see if it exists in a stoplist. If it does, it is discarded. Words not discarded are processed to remove suffixes, converted to upper case and truncated to five character terms. Each term is stored in a global array Ndc(Term) where the value stored is the count of the number of times the term has occurred in the collection. The term is also stored in the document-term matrix @doc(DocNr,Term) where the values stored is the number of times this term has occurred in this document. The relative position of each term in each document is stored in a global array Ndoc(DocNr,Term,Position) where Position is the term's position number in the document.

At the end of phase 1, the total number of terms is stored in Nterm, the total number of document occurrences (excluding terms found in the stop list) is stored in Ncount(0), and the number of distinct terms in the collection (i.e., terms of low frequency were discarded).

In phase two, the document frequency of each term is calculated. The document frequency is the number of documents each term occurs in and it is stored in global Ndf(Term). In our experiment—after discarding for frequency (high and low)—863 unique terms remained at this point with an average frequency of usage of 90.

![Figure 3: Preprocessor Flow Chart](image)

The title and each line of the text are decomposed into individual words. Each word is checked to see if it exists in a stoplist. If it does, it is discarded. Words not discarded are processed to remove suffixes, converted to upper case and truncated to five character terms. Each term is stored in a global array Ndc(Term) where the value stored is the count of the number of times the term has occurred in the collection. The term is also stored in the document-term matrix @doc(DocNr,Term) where the value stored is the number of times this term has occurred in this document. The relative position of each term in each document is stored in a global array Ndoc(DocNr,Term,Position) where Position is the term's position number in the document.

At the end of phase 1, the total number of terms is stored in Nterm, the total number of document occurrences (excluding terms found in the stop list) is stored in Ncount(0), and the number of distinct terms in the collection (i.e., terms of low frequency were discarded).

In phase two, the document frequency of each term is calculated. The document frequency is the number of documents each term occurs in and it is stored in global Ndf(Term). In our experiment—after discarding for frequency (high and low)—863 unique terms remained at this point with an average frequency of usage of 90.

![Figure 4: Retrieval Flow Chart](image)

In phase three, the discrimination coefficients for each term are calculated. These are stored in a @disc(Term). The formula is given in figure 3. Terms whose discrimination value falls below a given limit are discarded from all global arrays in which they appear. The document frequencies and term count data are recalculated to take into account term deletions. At this point in our experiment, there were 300 unique terms with an average frequency of usage of 86.

In phase four, the term-term correlation matrix @corr(Term1,Term2) is calculated. This is a square matrix whose rows and columns represent terms. A given position in the matrix indicates the degree of co-occurrence of any two terms in the system. For each document, each pair of terms is examined using data in @inc(DP,Term1,Term2). Terms which are too far from one another are not considered. For those which are sufficiently
...hanced by inclusion of synonyms from the $\phi(r)$ matrix. The user selects which similarity function is to be used and the query vector is processed against the centroid vectors. The 16 highest ranking document titles and their similarity coefficients are displayed in rank order.

The user may select a document from this list for display. Upon displaying a document, the user may request lower or higher ranking documents from the document set, or the user may request documents related to the currently displayed document. In this case, the current document's vector becomes a new query vector and the process is repeated. This constitutes navigating or browsing through the database. The initial query provides a starting point from which the user may move to related documents. The process is recursive so that previous results may be returned to and intermediate queries are retained in a library.

Other functions include display of documents by document number (and subsequent navigation of the database); display of known terms; display of documents by terms; display of terms by document; and Boolean retrieval by terms.

![Program for Term-Term Correlation Matrix](image)

Figure 6: Program for Term-Term Correlation Matrix

An example of the retrieval operation can be seen in figure 7. In this case, the query vector only consisted of the term CD-ROM. Using only one term highlights the differences in the similarity coefficients. The query vector was compared directly with document vectors. As can be seen, even though each displayed document contained the query term, there was a difference of more than seven to one in the resulting coefficients. The top 16 ranking titles are displayed (with titles truncated) along with their coefficient of similarity with the query vector. Altogether, 58 documents satisfied the query. The user at this point is free to recall an article or revisit the search with different terms and parameters.

The system presently operates under DEC or MS-DOS machines with adequate hard disk storage and under UNIX 3.0 on
The MUMPS interpreter was locally developed and is in C. Some of the MUMPS programs contained non-standard language features which were translated by a preprocessor into standard MUMPS interpreter calls. Retrieval times vary depending upon the speed of the machine but, even with the large database used in this experiment—it usually takes no more than a few seconds from query to display of titles on an Intel 386 33 MHz processor. The system consists of approximately 2,000 lines of code.

Conclusions

Those familiar with information retrieval applications will easily appreciate the advantages of the MUMPS environment to programming applications of this type. The most important of these is due to the manner in which MUMPS integrates its database into the language. The MUMPS global array facility is clearly ideally suited to the type of operations required in this type of programming. Further, the sparse matrix nature of the database greatly reduces the overall storage requirements. For example, the $M(t)$ matrix is conceptually very large but, in fact, the number of elements that actually exists is very much smaller. Likewise, the $N(t)$ matrix would be impossibly large were it not for the fact that most possible elements do not exist.

As CD-ROM drives proliferate and the volume of text databases expands, there will be an increasing need for more sophisticated software to access the data, especially on personal computers. Already, PC's are being sold with CD-ROM's as standard equipment. The need to develop improved indexing and retrieval applications will grow, but this growth is limited by the facilities of the traditional PC-based language environments. No other language environment is as well suited to this task as MUMPS. Developers who recognize this fact will be in a more competitive position to deliver their products to the market.

References


SBS - International

Software Consultants
805 Forest Avenue
Palo Alto, California 94301
415.321.3435
Fax: 415.323.4005