CATALOGING SOFTWARE PACKAGES FOR AUTOMATIC DOCUMENT PROCESSING

G. Salton

TR 76-268

March 1976

Department of Computer Science
Cornell University
Ithaca, New York 14853

Cataloging Software Packages for

Automatic Document Processing

G. Salton*

Abstract

Considerable advances have been made in the last few years in the development of hardware structures useful for the implementation of real-time document processing systems and on-line information retrieval. At the same time, relatively little is known about the software packages most appropriate for the new technology. Following certain proposals for the creation of national and international registries of software packages, a sample catalog of software routines is developed for use in modern automatic documentation systems.

1. Background

Over the last few years, a variety of national and international organizations have been concerned with the generation of policies and procedures designed to facilitate the dissemination of knowledge and to simplify the acquisition of information by interested groups of users. The main purpose has been the creation of suitable conditions in various countries and political subdivisions allowing for free and unfettered information dissemination and exchange. The following suggestions have been made among many others [1,2]

1. The increased use of information technology for bibliographic production designed to achieve better bibliographic control over the published resources in each county.

---

* Department of Computer Science, Cornell University, Ithaca, NY 14853.

2. The generation of improved national telecommunications systems.
3. The support of international bibliographic standards, and the development of cooperative projects in bibliographic control, documentation practice and technical procedures.

In the area of hardware structures, a good deal of progress has already been made in suggesting standards and common objectives. Attention has been given in particular to information processing components such as reprography, computer-output-microfilm (COM), photo composition, and computer products such as tapes, microstorage devices, and various input and output devices. The aim in each case is the simplification of information exchanges by supplying bibliographic control standards together with approved computing and transmission technologies.

More recently proposals have been made to extend these cooperative endeavors also to the software packages which actually manipulate and process the information files. In the remainder of this report a number of processing routines are examined which may be included in the proposed software registry.

2. Information Processing Tasks

The concern with information processing software arises from the fact that technology on the one hand, and formatting and descriptive standards on the other are insufficient by themselves for the supply of information resources to the existing user populations. Needed in addition is information about the availability of the software packages which actually process the information resources, and the formulation of standards for the information processing software.

Proposals have accordingly been made by a number of experts for the collection of information about existing software packages. Two main steps appear to be required: [3]

1. The identification and registration of operational computer software relevant to library, archives and documentation.
2. The establishment of a central register of data banks, and the acquisition of the relevant tapes .... and of computer services for searching and processing these tapes.

The premise underlying these ideas is that a great many information resources already exist in various places and that more or less standardized "off-the-shelf" information software could be made available to process these existing data collections, provided only that information became available about the operational status of the various programs, the conditions of availability, the necessary program documentation, and data about machine compatibility, reliability and cost.

Most experts in information processing would agree that the use of standardized program packages would greatly facilitate the distribution and processing of information resources. The problem which arises with the proposal is that "off-the-shelf" software can only rarely be used off-the-shelf in actual fact. Each computer system, even within a common series, has its own particular component set and operating characteristics, so that compatibility between different installations is always difficult to achieve. Furthermore, software packages may exhibit peculiarities and special provisions which render their use impossible outside the special environment for which they were originally designed. A typical example is provided by the many different on-line search and retrieval packages which are incompatible for practical purposes, even though they all use the same basic file organization (inverted file directories) and the same search and retrieval techniques (Boolean query processing).

A useful software inventory must therefore include information not only about the information processing tasks which are performed in each case, but also about the underlying data structures and the approaches used in handling the data manipulations.

A number of well-known information processing tasks are listed in Table 1, including the main file generation systems which underlie all other processes; the construction of information products such as dictionaries and indexes that prove useful in many document processing environments; certain language processing routines which normally form the basis for automatic text processing and indexing systems; the query processing systems used to identify relevant information following a search of the stored records; the output display processing systems which interact with the system during information search and retrieval; and finally system evaluation and file maintenance facilities.

The function of many of these tasks becomes clear by examining the uses and applications shown for each task on the right-hand side of Table 1. All the listed data and document handling procedures necessarily depend on the file arrangements used to organize the stored records. Five basic file generation methods suggest themselves:

a) sequential, where the records are unordered and a complete scan of the file is used to retrieve the relevant information;

b) list structures, where the records exhibiting a common characteristic, or key, are either chained together as in the well-known multilist storage, or collected in a common storage area as in the inverted implementation;

c) scatter storage files, where the record address corresponding to a given key can be directly calculated using a so-called "hash" function; scatter storage techniques are usable in an information retrieval environment to access directories, dictionaries and thesauruses;

d) tree structures, where hierarchical (inclusion) relations are defined between the stored items; tree structured files arise naturally in the manipulation of hierarchical dictionaries, library classification schedules, and syntactically analyzed written texts; and, finally,

e) clustered files, where the records exhibiting sufficiently large similarities with each other are assembled into classes, or clusters, and treated as a group during subsequent processing.

Each of these file organization systems imposes different demands on the processing sequences, as will be seen when the software packages are described in the next section.

A number of well-known information products such as printed indexes, reports, and bibliographic tape distribution systems are listed in Table 1 following the file organization systems. All of these products involve mainly sorting, editing, and general information packaging methods. Various administrative tasks, including the generation of statistical data and of cost information are also mentioned in the Table, as are certain new information network processing tasks.

The main information processing methods of interest in a document handling environment consist of the manipulation of small portions of the individual records. This is the case in particular for the indexing and cataloging operations, the information search, and the retrieval procedures. The main indexing and content analysis operations are listed in Table 1 under the general title of language processing, the assumption being that the indexed items are bibliographic records available in natural language form. Among the tasks that may be included in an automatic indexing system are

a) the recognition of individual text words;

b) the assignment of an importance index (weight) to each text unit, term importance being assumed to be a function of the occurrence pattern of the words in the individual documents as well as in the collection as a whole;

c) the identification of specific, narrowly defined content identifiers by combining broader individual terms into phrases; this can be done by using statistical measurements of term cooccurrence, or syntactic language analysis methods;

d) the identification of broad content descriptions obtainable by grouping individual terms into classes of related terms and assigning such thesaurus classes as content identifiers;

e) the utilization of syntactic and semantic analysis methods as part of the automatic text indexing system.

Text abstracting is related to indexing, since the determination of individual term importance coefficients leads to the generation of scores for complete sentences, and hence to the inclusion in a text abstract of sentences judged to be important for content representation.

The information retrieval tasks proper include first the information search — that is, comparison of query formulations with certain stored items — and then the retrieval of those stored records judged to be sufficiently similar to the user queries. In modern retrieval environments, provisions are always made for reformulating queries that do not lead to satisfactory search output. This normally involves the display of portions of the stored information — titles, abstracts, term frequency information, text excerpts, and the like — followed by the generation of new, improved query statements.

Many document processing systems also include system evaluation routines. The retrieval component may be evaluated in terms of objective measures such as recall and precision which are defined, respectively, as the proportion of relevant material retrieved and the proportion of retrieved material that is relevant.

The last, and possibly the most important, document processing component includes the file maintenance operations used to keep the collection current following the addition of new or altered information, or the deletion of obsolete items. These operations depend on the file organization system in use and on operational parameters such as amount of file growth, updating frequency, and the like.

Some of the important software packages that may be used to implement the foregoing tasks are briefly examined in the next section.

3. Software Packages

It is not useful in the context of the present study to list hundreds of software components that may conveniently be included in a given document processing environment. Instead, it appears more important to concentrate on those processes which are already widely used in existing operational or experimental systems, and which may thus form the basis for a future catalog of compatible information processing software. A number of important routines are listed in the summary of Table 2.

Sorting operations are obviously essential. They produce ordered lists of records in accordance with an ordering criterion, and they form the basis for the production of several automatic search aids, such as KWIC and KWOC (keyword-in-context and keyword-out-of-context) indexes. In retrieval systems in which a similarity coefficient is calculated following each query-document comparison, sorting routines can bring the retrieved items to the user's attention in a convenient decreasing relevance order, that is, in decreasing query-document similarity order. In recent years, a number of efficient sorting methods have been devised for which the number of comparisons between stored items is proportional to n log n when n items are sorted. Among these are the quick sort and the heap sort processes which operate by exchanging at each step two or more items known to be out of order in their respective sequences. [4] Sophisticated external tape merging systems are also known. Standards for fast sorting routines are not difficult to establish, and their inclusion in a software catalog should pose no problems.

The best known retrieval technique is one based on the so-called inverted file organization. A simple example is shown in Fig. 1. Typically, each query term (key) is first decoded in a table (the directory) which points to the address of a list of record identifiers corresponding to the given key. Each record identifier in the (inverted) list file in turn points to the actual data record stored in the data file. The organization of Fig. 1 makes it possible to restrict the number of accesses in the data file to only those records which actually agree with the query specification. This is made possible by processing the record identifier lists corresponding to the several keys contained in the query statement instead of the actual records. Thus, all record identifiers pertaining to keys "K_i and K_j" are obtained by intersecting the K_i and K_j lists; records corresponding to keys "K_i or "K_j" are similarly identified by a list merging operation. Efficient methods are known for performing these list intersection and merge processes.

Fast procedures must also be available to manipulate chained record lists of the kind used in so-called multilist organizations. An example is given in Fig. 2. It may be seen that the list file used in the inverted file of Fig. 1 is replaced in Fig. 2 by a chain of all the records belonging to a common key. The directory file thus points to the first record in the data file for each given key; this record in turn identifies the next record in the list, and so on down to the last record. The manipulation of chained lists is used extensively in computing practice, and efficient procedures exist for traversing short lists and for updating the chains when records are added or removed. [5]

In addition to the sorting and list-processing operations, extensive use must also be made in a document processing environment of vector and matrix manipulations. This stems from the fact that an indexed document collection can be regarded as a matrix where each row (each term list) represents a document, and each column identifies the assignment of a term to the documents of a collection. An example of a term-document list is shown in Fig. 3.

It is clear from Fig. 3 that similarity coefficients between pairs of records (documents) can be obtained by comparing the corresponding matrix rows; similarities between terms are obtained in the same way by comparing pairs of columns. Such similarity computations between the row and column "vectors" of the matrix form the basis for automatic thesaurus construction methods, and for document classification activities. [6]

A matrix is also a convenient symbolism for the representation of the records relevant to a given query (query-document relevance matrix) and for the specification of the document classes which determine a given classification system (document-class matrix). Vector and matrix manipulation packages are, of course, used in many areas of computer practice.

Classified files in which items judged to be sufficiently similar are placed into a common class have long been used in many environments. The main virtue of a classification system is that the information search can be restricted to only a small number of classes. Relatively inexpensive automatic methods, requiring approximately n log n operations for the classification of n items, are available for the construction of automatic classification systems. Thus starting with the document vectors of Fig. 3 one can construct classes of similar records illustrated in the simple example of Fig. 4(a). [7]

It may be seen from the example that individual records, represented by x's in Fig. 4, are grouped into classes identified by a central item (centroid). The centroids representing individual document classes may in turn be grouped into higher level classes represented by super centroids, as shown in the organization of Fig. 4.

The strategy used to traverse or search a clustered file is equivalent to the well-known B-tree search used in various computer applications. A tree search is illustrated in Fig. 4(b). [4] An incoming query is first compared with the super centroids labelled 1 and 2 in the example of Fig. 4. If the query is similar to super centroid 1, the next comparison is made with the second-level centroids 3 and 4; otherwise the comparison is with centroids 5 and 6. These second-level centroids in turn give access to lower level centroids until eventually the comparison is made with the individual documents. This type of tree-search can be used for narrow searching by looking at very few centroids at each step, or for broader searches when a greater number of classes must be included in the search. Both the construction of the cluster tree (the classification) and the cluster search itself can be efficiently implemented as processes of order n log n for n file items.

Most language processing systems are restricted to statistical text manipulations only, using the term frequency distributions in the documents of a collection. A typical set of such operations is listed in Fig. 5. The idea is that good content identifiers exhibit average document frequency characteristics — they occur in the collection neither too frequently nor too rarely. Frequency transformations are useful to generate content identifiers with the appropriate average frequency characteristics by assembling broad, high-frequency terms into term phrases, and collecting narrow, low frequency terms into term classes as shown in Fig. 5. [7]

The operations included in Fig. 5 are based only on term weighting and frequency counting procedures. It is possible to supplement an automatic indexing system by adding syntactic structure recognition or even semantic analysis procedures. When that is done a new data structure becomes important, namely an abstract graph. It is well-known that syntactically analyzed text excerpts can be represented in tree or graph form. Examples are shown for two sample sentences in Fig. 6, where grammatically dependent structures are attached to the corresponding governing elements on the next higher level. To obtain a match between sentence elements the node labels must match as well as the connections between the nodes. For the sentences of Fig. 6 it is seen that the fragment "information retrieval" constitutes a grammatically acceptable phrase in the bottom tree, since "information" is grammatically dependent in that case on "retrieval" to which it is attached. The phrase "information retrieval" is, however, not recognized in the tree at the top of Fig. 6, since "information" and "retrieval" are not structurally dependent on each other in the corresponding sentence.

Various graph matching procedures have been described in the literature. [8] In an information retrieval environment these are used advantageously to improve the search precision by taking into account during the search the the syntactic and semantic relationships which may exist between individual terms.

The last two software components included in Table 2 are "table look-up" and "vector matching" processes. It is worthwhile to implement these routines as efficiently as possible, because the corresponding operations are included in most of the other packages. Table look-up routines are used, for example, in the inverted list implementation of Fig. 1 to find the addresses of the various keyword lists; vector matching operations prove useful in searching, classification, retrieval, and file updating. Efficient table look-up methods using hash function techniques are described in the literature; the same is true for the many vector processing methods that have been used over the years. [5,6,7,8]

To summarize: the software packages of main interest in an automatic document processing environment include sorting routines, list processing operations, vector and matrix manipulations, clustering and tree search procedures, and graph matching methods. Examples are given in this study to illustrate the use of many of these techniques. The components listed in Table 2 are independent of actual implementations; they may therefore be expected to form the basis for future catalogs of information retrieval software.

References

[1] Intergovernmental Conference of the Planning of National Documentation, Library and Archives Infrastructures, National Information Systems (NATIS) - Objectives for National and International Action, Report COM. 74/NATIS/3, UNESCO, Paris, September 1974.

[2] Report by the Director-General on the Intergovernmental Conference on the Planning of National Documentation, Library and Archives Infrastructures, UNESCO General Conference, Eighteenth Session, Report 18c/92, UNESCO, Paris, October 1974.

[3] J. Becker and L.G. Burchinal, Planning for Information Technology, Intergovernmental Conference on the Planning of National Documentation, Library and Archives Infrastructures, Report COM. 74/NATIS/Ref. 4, UNESCO, Paris, July 1974.

[4] D.E. Knuth, The Art of Computer Programming, Vol. 3, Sorting and Searching, Addison-Wesley Publishing Co., Reading, Mass., 1973.

[5] D.E. Knuth, The Art of Computer Programming, Vol. 1, Fundamental Algorithms, Addison-Wesley Publishing Co., Reading, Mass., 1968.

[6] G. Salton, Automatic Information Organization and Retrieval, McGraw Hill Book Co., New York, 1968.

[7] G. Salton, Dynamic Information and Library Processing, Prentice-Hall Inc., Englewood Cliffs, NJ, 1975.

[8] A.T. Berztiss, Data Structures — Theory and Practice, Second Edition, Academic Press, New York, 1975.

Information Processing Tasks	Principal Use and Application
1. File Generation	
sequential files	tape distribution services, selective dissemination of information (SDI) services
list structures	keyword lists, multilists and combined indexes
scatter storage files	symbol table processing, tabular search
tree structures	syntactic processing, directory storage, hierarchical dictionary, library classification
clustered files (B-trees)	document associations, term associations, broad and narrow searching
2. Tape Distribution Services	recording of bibliographic information in standard form, search capability using various keys
3. Printed Index Production	keyword-in-context (KWIC) indexes, citation indexes, keyword lists, catalogs
4. Report Generation	statistical reports, activity reports
5. Administrative Computation	statistics, usage costs, log-in and log-out
6. Network Computations	transmission processing, switching operations, communications
7. Language Processing	
indexing and text analysis	term identification, automatic content analysis, generation of term relevance and term frequency lists
term discrimination computations	determination of term importance in collection

Principal Information Processing Tasks

Table 1

Information Processing Tasks	Principal Use and Application
phrase generation	determination of term combinations
thesaurus generation	term associations, term clustering and grouping, term relationships
syntactic computations	text analysis
cognitive manipulations	semantic processing, inference making, semantic map processing
8. Text Abstracting	sentence evaluation, text reduction
9. Searching	query processing for exact match, partial match, Boolean match, best match, nearest neighbor match
10. Query Reformulation	feedback processing, use of relevance assessments
11. Display Processing	title-abstract-citation display, tutorial operations, term frequency display vocabulary processing text structuring, editing and reformatting error processing
12. Retrieval Evaluation	recall-precision computations, relevance processing, cost analysis
13. File Maintenance	addition and/or deletion of keywords, addition and/or deletion of records, document modification, file growth and retirement

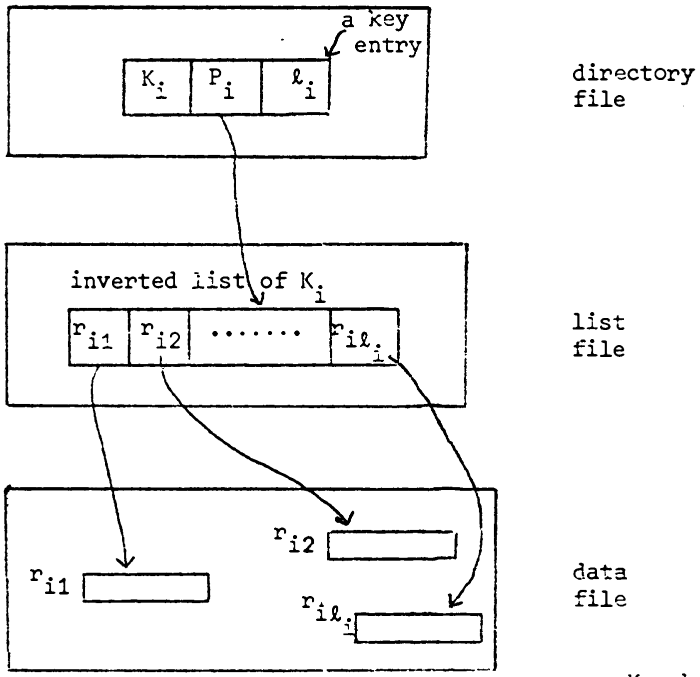
Principal Information Processing Tasks

Table 1
(cont.)

Software Package	Main Usage
1. Fast Sorting Method (heap sort, quick sort)	printed index production, ranking of search output in decreasing query similarity order
2. Inverted List Operations	keyword list merging, key decoding during search
3. Chained List Processing	tree searching, overflow processing, list tracing for multilists
4. Term-Document Matrix Processing	classification matrix, relevance matrix, term-term associations, document associations
5. Space Density Measurements	determination of dense areas in collection, clustering and classification
6. Cluster and B-Tree Processing	term and document classification, cluster searching
7. Text Processing	word truncation, word juxtaposition for phrase formation, structural analysis
8. Statistical Text Manipulation	term weighting, automatic abstracting, sentence weighting
9. Graph Matching	syntactic analysis of sentences, retrieval of structured data
10. Table Look-up	thesaurus manipulation, key decoding
11. Vector Matching	automatic classification, information search, file updating, query processing

Principal Software Packages

Table 2



Typical Inverted File Organization

K_i key

P_i pointer

ℓ_i list leng

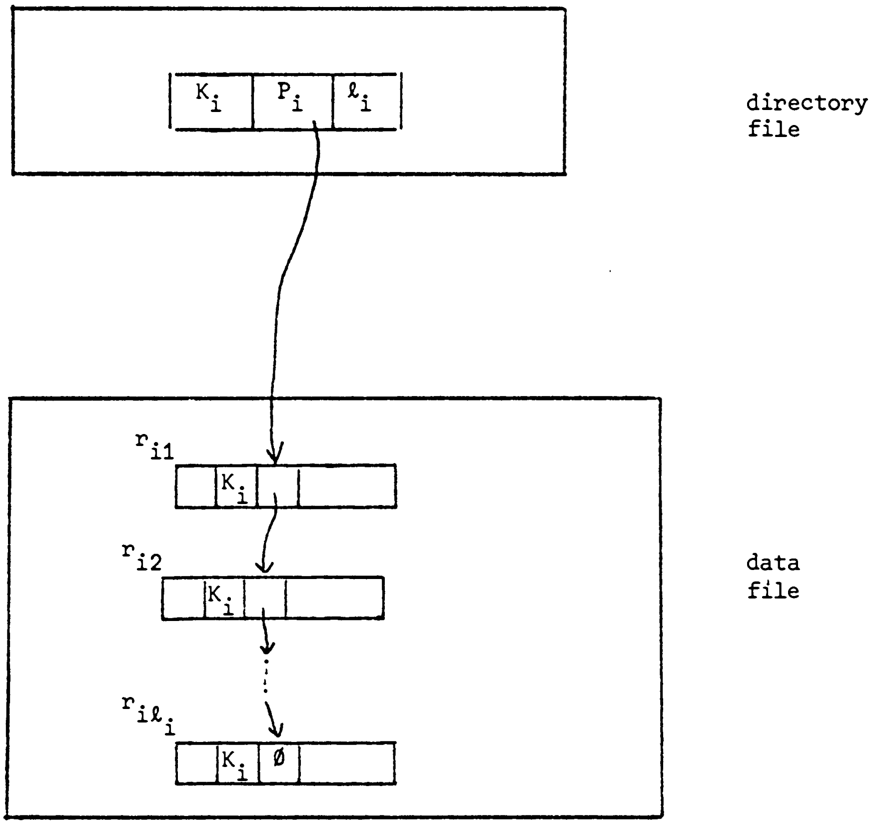
r_{ij} identifie
jth reco
K_i list

a) Intersection of two or more from list file to obtain the records common to these lists.

b) Merging of two or more lists from list file to obtain a single list of direct items .

Typical Inverted List Processes

Fig. 1



Typical Multilist Organization

K_i key

P_i pointer

ℓ_i list length

r_{ij} jth record in K_i list

a) Retrieval of all records included in K_i-chain to determine those which match the query statement.

b) Simultaneous tracing of two or more chains corresponding to several keys.

c) List updating by removal and/or addition of keys and records.

Typical Multi-list Processes

Fig. 2

$$\begin{array}{r c c} & & \overbrace{\begin{matrix} T_1 & T_2 & \cdots\cdots & T_t \end{matrix}}^{\text{terms}} \\ \text{documents} & \left\{\begin{matrix} D_1 \\ D_2 \\ \vdots \\ D_n \end{matrix}\right. & \begin{bmatrix} d_{11} & d_{12} & \cdots\cdots & d_{1t} \\ d_{21} & d_{22} & \cdots\cdots & d_{2t} \\ & & & \\ d_{n1} & d_{n2} & \cdots\cdots & d_{nt} \end{bmatrix} \end{array}$$

d_{ij} is weight of term j in document i

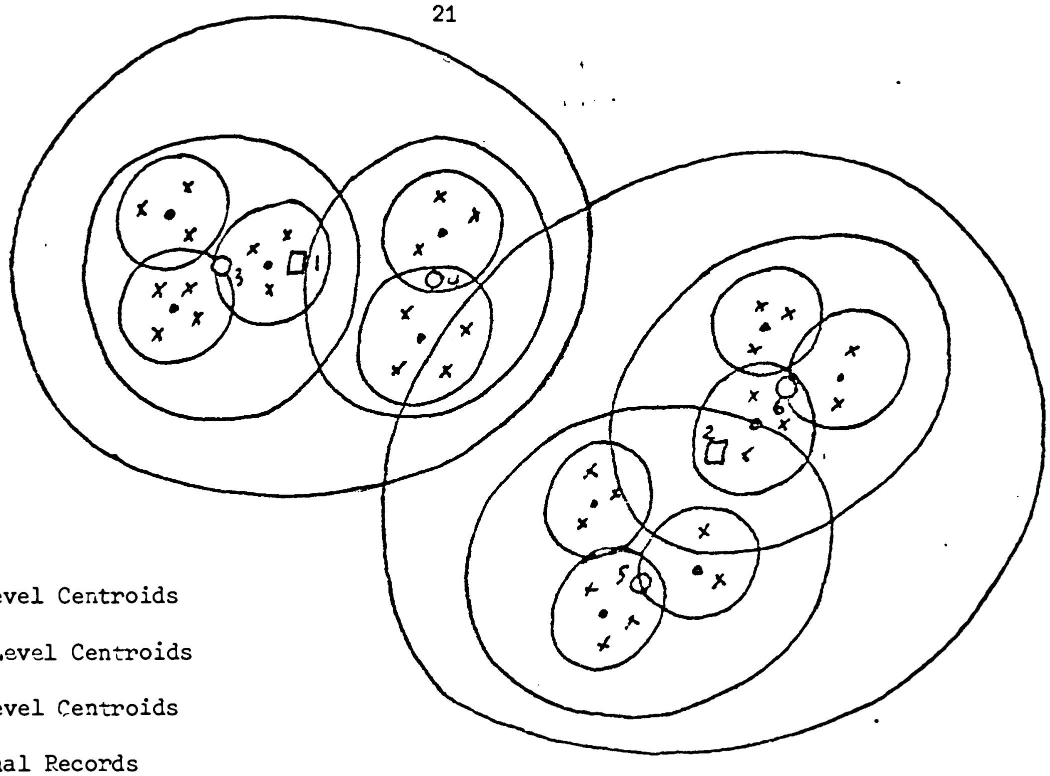
Term-Document Matrix

a) Comparison of two rows of matrix, or equivalently computation of similarity measure $S(D_i, D_j)$ produces document similarities.

b) Comparison of two columns of matrix, or equivalently $S(T_k, T_\ell)$ produces term similarities.

Typical Term-Document Matrix Process

Fig. 3



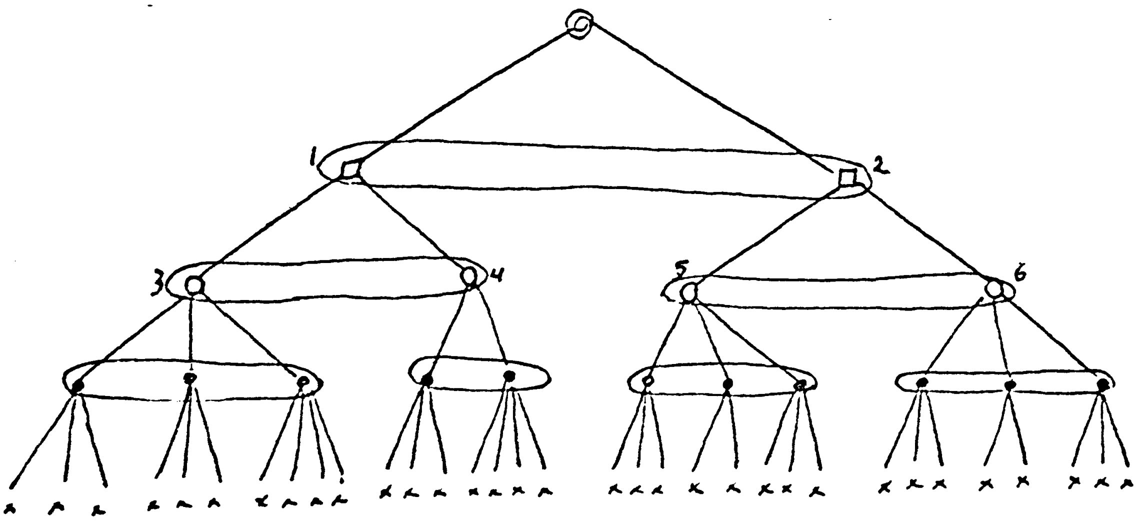
□ First Level Centroids

○ Second Level Centroids

● Third Level Centroids

✗ Individual Records

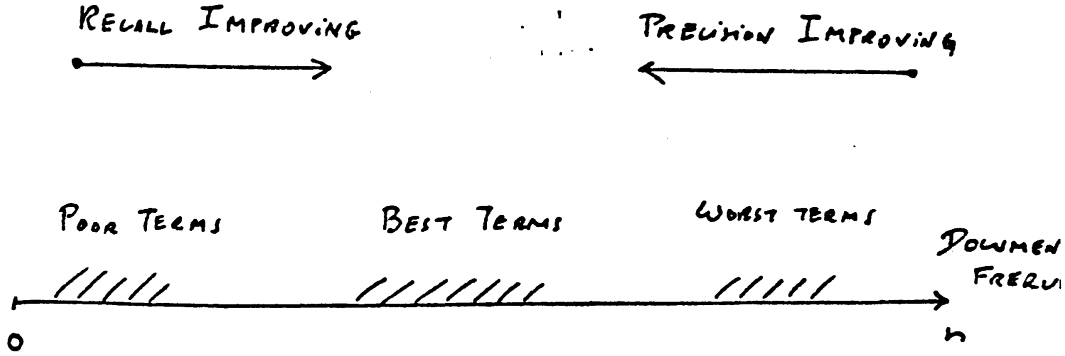
a) Clustered File Organization



Storage Bucket

b) Search Strategy for Clustered Files

Fig. 4



Term Frequency Characteristics

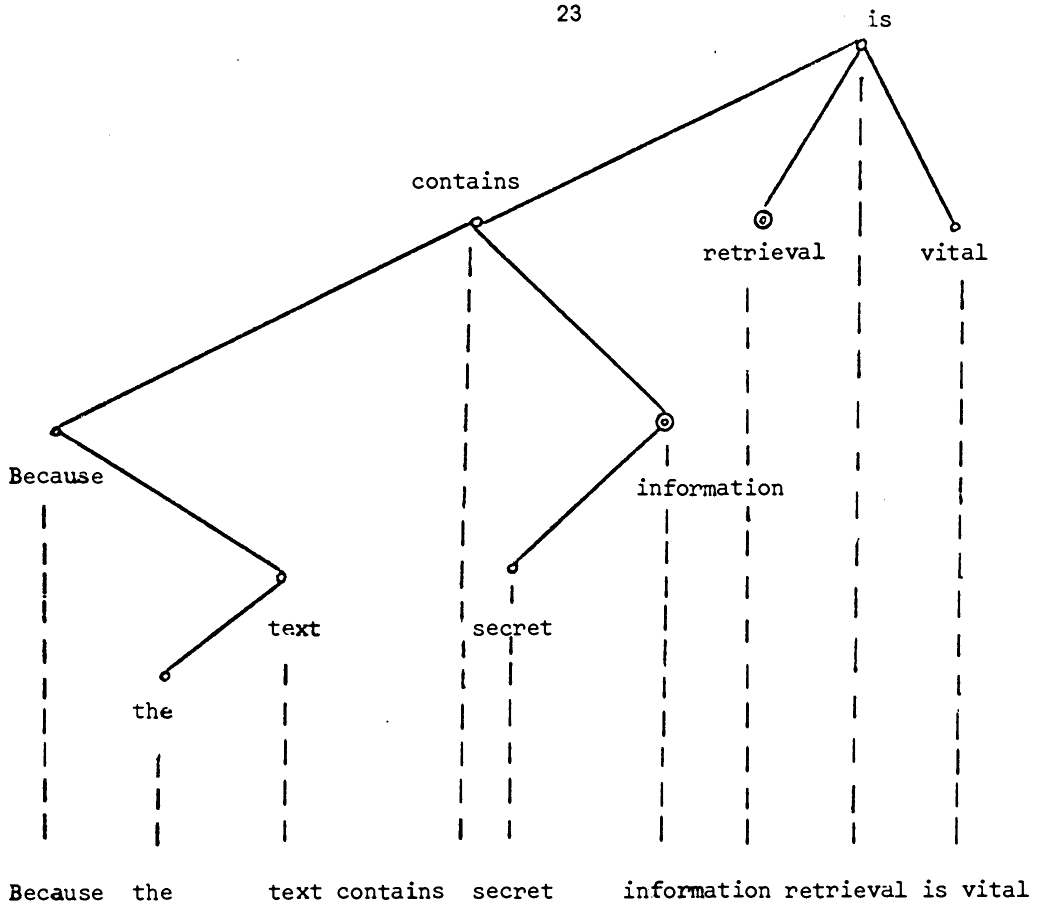
a) Assemble poor low frequency terms into thesaurus classes thereby creating broader content identifiers to enhance recall.

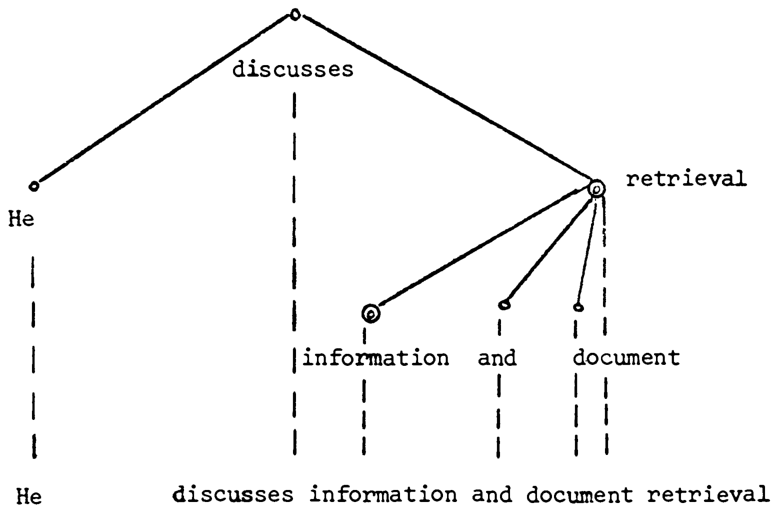
b) Break-up high frequency terms to form term phrases thereby creating narrower content identifiers to enhance precision.

c) Assign term weights proportional to term frequency in individual documents and inversely proportional to frequency in the whole collection.

Typical Text Processing Methods

Fig. 5





Nonmatching Sentence Graphs

Fig. 6