UNCLASSIFIED

AD 408 934

DEFENSE DOCUMENTATION CENTER
FOR
SCIENTIFIC AND TECHNICAL INFORMATION

CAMERON STATION, ALEXANDRIA, VIRGINIA

UNCLASSIFIED

NOTICE: When government or other drawings, specifications or other data are used for any purpose other than in connection with a definitely related government procurement operation, the U. S. Government thereby incurs no responsibility, nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.

AFCRL-63-134

THE COMPUTATION LABORATORY

Harvard University

Cambridge, Massachusetts

AF19(604)-8509

Project Nos. 5632, 5633

SCIENTIFIC REPORT NO. ISR-3

INFORMATION STORAGE AND RETRIEVAL

prepared for

AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS

April 1, 1963

Gerard Salton
Project Director

©

Copyright, 1963

By the President and Fellows of Harvard College

Use, reproduction, or publication, in whole or in part, is permitted for any purpose of the United States Government.

Requests for additional copies by agencies of the Department of Defense, their contractors, and other government agencies should be directed to:

DEFENSE DOCUMENTATION CENTER (DDC)
ARLINGTON HALL STATION
ARLINGTON 12, VIRGINIA

Department of Defense contractors must be established for DDC services or have their "need-to-know" certified by the cognizant military agency of their project or contract.

All other persons and organizations should apply to:

U. S. DEPARTMENT OF COMMERCE
OFFICE OF TECHNICAL SERVICES
WASHINGTON 25, D. C.

STAFF OF THE COMPUTATION LABORATORY AND OF THE COMPUTING CENTER

Lynda Addison
Deborah D. Aulenback
Walter Broderick
Arlene Brown
Robert J. Burns
Mark Cane
Ronald Chayer
Alan Cohen
Evelyn Cone
Isabel Corbató
Milan Dabcovich
Jean M. D'Agostino
Richard Delery
Arthur Dolan
David Drew
Frank Engel, Jr.
Robert Eubanks
Shimon Even
Patrick Fischer
Alan Gillis
Sheila Greibach
Teruhiro Hayata
Inez Hazel
Mary E. Hester
Yu-Chi Ho
Kenneth Iverson
John J. Jackson
Eric Johansson
Lawrence Kelley
Elena Kirsch
Susumu Kuno
Beverly Lee
Alan Lemmon
Michael Lesk
Joseph I. Lewko
Irina Lynch
Mary Lynch
Edward Lyons
Rita B. Mahony
Paul Marques
George McManus
Jeri Mignault
Charlotte Modahl
Maxine Nealley
Edward Nelson
Anthony G. Oettinger
Marguerite Pandt
Richard Powers
Mary Ranelli
Diane Redonnet
Joseph Rocchio
Gerard Salton
Jacquelin Sanborn
Tina Simeone
Carol Smith
Edward H. Sussenguth, Jr.
Margaret Thompson
Rodney Thorpe
C. Cynthia Tukis
John van Bemmel
Richard S. Varga
Elizabeth Vliet
Hao Wang
Richard Whalen
Stephen Young
Penelope W. Zimmerman

LIST A

Code	Organization	No. of Copies
AF 5	AFMTC AFMTC Technical Library - MU-135 Patrick Air Force Base Florida	1
AF 18	AUL Maxwell Air Force Base Alabama	1
AF 32	OAR RROS, Col. John R. Fowler Tempo D 4th and Independence Avenue Washington 25, D.C.	1
AF 33	AFOSR OAR (SRYP) Tempo D Fourth and Independence Avenue Washington 25, D. C.	1
AF 43	ASD (ASAPRD - Dist.) Wright-Patterson Air Force Base Ohio	1
AF 124	RADC (RAALD) Attention: Documents Library Griffiss Air Force Base New York	1
AF 139	AF Missile Development Center (MDGRT) Holloman Air Force Base, New Mexico	1
AF 314	Headquarters OAR (RROSP) Attention: Major Richard W. Nelson Washington 25, D. C.	1
Ar 5	Commanding General USASRDL Attention: Technical Documents Center SIGRA/SL-ADT Fort Monmouth, New Jersey	1

Code	Organization	No. of Copies
Ar 9	Department of the Army Office of the Chief Signal Officer Attention: SIGRD-4a-2 Washington 25, D. C.	1
Ar 50	Commanding Officer Attention: ORDTL-012 Diamond Ordnance Fuze Laboratories Washington 25, D.C.	1
Ar 67	Redstone Scientific Information Center U. S. Army Missile Command Redstone Arsenal, Alabama	1
G 2	Defense Documentation Center (DDC) Arlington Hall Station Arlington 12, Virginia	20
G 31	Office of Scientific Intelligence Central Intelligence Agency 2430 E Street, N.W. Washington 25, D.C.	1
G 68	Scientific and Technical Information Facility Attention: NASA Representative (S-AK/DL) P. O. Box 5700 Bethesda, Maryland	1
G 109	Director Langley Research Center National Aeronautics and Space Administration Langley Field, Virginia	1
N 9	Chief, Bureau of Naval Weapons Attention: DLI-31 Department of the Navy Washington 25, D. C.	2
N 29	Director (Code 2027) U. S. Naval Research Laboratory Washington 25, D.C.	2

LIST A (continued)

Code	Organization	No. of Copies
I 292	Director, USAF Project RAND Thru: AF Liaison Office The Rand Corporation 1700 Main Street Santa Monica California	1
M 6	AFCRL, OAR (CRXRA - Stop 39) Laurence G. Hanscom Field Bedford, Massachusetts	25
AF 253	Technical Information Office European Office, Aerospace Research Shell Building 47 Cantersteen Brussels, Belgium	1
AF 318	Aero Research Laboratory (OAR) AROL Library AFL 2292, Building 450 Wright-Patterson Air Force Base Ohio	1
Ar 107	U. S. Army Aviation Human Research Unit U. S. Continental Army Command Attention: Major Arne H. Eliasson P. O. Box 428 Fort Rucker. Alabama	1
G 8	Library Boulder Laboratories National Bureau of Standards Boulder, Colorado	2
G 9	Defense Research Member Canadian Joint Staff 2450 Massachusetts Avenue, N. W. Washington 8, D. C.	3
M 63	Institute of the Aerospace Sciences, Inc. Attention: Librarian 2 East 64th Street New York 21, New York	1

LIST A (continued)

Code	Organization	No. of Copies
M 84	AFCRL, OAR (CRXR, J. R. Marple) Laurence G. Hanscom Field Bedford, Massachusetts	1
N 73	Office of Naval Research Branch Office, London Navy 100, Box 39 F.P.O., New York, New York	10
U 32	Massachusetts Institute of Technology Research Laboratory Attention: John H. Hewitt Building 26, Room 327 Cambridge 39, Massachusetts	1
U 431	Alderman Library University of Virginia Charlottesville, Virginia	1
M 80	ESD (ESRDD, Major W. Harris) Laurence G. Hanscom Field Bedford, Massachusetts	1
	Remaining copies to:	
	Headquarters AFCRL, OAR (CRB, Harry Blum) Laurence G. Hanscom Field Bedford, Massachusetts	10

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

page

SECTION II (continued)

SECTION III

TABLE OF CONTENTS (continued)

page

SECTION III(continued)

SECTION IV

SECTION V

SECTION VI

SUMMARY

The present report, number ISR-3, contains a record of the continuing investigation of various techniques for automatic content analysis, and for the storage and search of structured information. In particular, some storage allocation techniques are examined which are useful in the manipulation of natural language data; a variety of machine programs and experiments are then described for the processing of bibliographic citations; and two models for a completely automatic document retrieval system are outlined.

Section I by G. Salton outlines two automatic document retrieval systems based on the usual word frequency counting procedures, supplemented by a number of auxiliary aids which replace a complete semantic analysis of the language. The principal aids consist of a hierarchical storage arrangement between certain subject categories, a set of cross references between related subjects, and sets of related or synonymous words attached to certain subject categories. Some basic retrieval procedures which make use of these structures are outlined.

Section II by M. Thompson includes a detailed description of a program designed to match bibliographic citations. The program is divided into two parts: the first part normalizes the format of each citation, and the second part performs the actual comparison between citations. This program, when completed, should be useful for the

automatic construction of citation indexes, and for the manipulation of bibliographic data in general.

Section III by E. H. Sussenguth, Jr., contains a description and evaluation of a chained tree structure useful for the storage of natural language data. The proposed storage organization permits a search procedure which is almost as efficient as that provided by a binary search, and yet is easily adapted to file changes, such as additions and deletions.

Sections IV, V, and VI by M. Lesk cover experiments performed with bibliographic citations for purposes of content analysis, as originally described in Section III of report ISR-2. Section IV is a description of the basic interpretive program used to perform the required matrix manipulations; the program accommodates matrices of variable size up to the capacity of the available core storage, and the symbolic operation codes reduce the program specification to an almost trivial operation.

Section V contains an analysis of citations and index term similarities obtained for a closed document collection for which citations from documents outside the collection are eliminated; the data are then compared with the citation similarities obtained for the corresponding open document collection. The similarity coefficients are found to be smaller for the open document collection, but the basic results are in general comparable for both collections.

Section VI describes a clustering experiment in which documents were grouped in accordance with similarities in the index terms attached to the documents, and in the corresponding citation sets. The two types of groupings are compared and are found to give dissimilar clusters for the document collection under investigation.

I. SOME HIERARCHICAL MODELS FOR AUTOMATIC DOCUMENT RETRIEVAL

Gerard Salton

1. Introduction

Automatic systems designed to furnish references or specific data in answer to search requests are known as information retrieval systems. If such systems are to perform effectively, provision must be made for the execution of the following types of operation:

(a) the analysis of information items and of search requests;
(b) the generation of identifiers used to represent information content;
(c) the normalization of information identifications to conform with some classification system;
(d) the storage of information items and identifications so as to simplify access to related items;
(e) the matching of information requests with information identifications, and the search for relevant items.

In order to perform a satisfactory analysis of the information, it is generally necessary to identify the basic elements which are used to represent the information and to recognize the rules by which the basic elements can be combined into larger units. For example, if the items of information to be stored consist of chemical compounds, the basic elements are atoms and bonds, and these can in turn be combined into larger molecules in accordance with certain structural rules.

If the items of information to be dealt with are documents or books, the basic elements are words in the natural language, and the larger units are sentences, paragraphs or chapters. The analysis of information consists in this case of the identification of document content. Unhappily, though it is somewhat easy to isolate the individual words in a text, the interpretation of the meaning of the words is much more difficult. Furthermore, no well-defined set of rules is known by which the individual words in the language are combined into meaningful word groups or sentences. Specifically, the correct identification of the meaning of word groups depends at least in part on the proper recognition of syntactic and semantic ambiguities, on the correct interpretation of homographs, on the recognition of semantic equivalences, on the detection of word relations, and on a general awareness of the background and environment of a given utterance.

Because of the many difficulties which arise in the semantic analysis of the natural language, certain simplifications are normally introduced before automatic analyses are attempted. These simplifications take the form of restricting the permitted area of discourse, or, alternatively, of limiting the types of linguistic structures permitted in the texts to be analyzed. Occasionally it is suggested that a special unambiguous language be used in preference to the natural language.

In many cases these restrictions may not be realistic, since it may be difficult effectively to control the area of discourse, or the types of structures being used. Moreover, no simplified or artificial language is likely to prove generally acceptable. The present report is therefore principally concerned with document retrieval methods using the unrestricted natural language for the representation of information. All processes are based on methods which can be carried out automatically including, in particular, those dealing with the identification of document content. A complete semantic dictionary giving all contexts for each word in the language is not constructed in view of the many difficulties inherent in the required linguistic analysis. For similar reasons most of the relations between words and sentences in a text are not explicitly identified. Instead, standard word counting and syntactic analysis techniques are used in conjunction with a small number of table look-up operations. The construction of the required tables is further described in the following sections.

2. Retrieval Structures

Since the principal input to the retrieval system consists of texts in the natural language, the words used in a given document must of necessity constitute the basic units to be dealt with. It is also useful to identify the principal classes of relations between words; specifically two main relational classes are normally distinguished: the generic or inclusion relations, sometimes known as analytic relations, and the nongeneric or synthetic relations. The first class of relations is exemplified by the hierarchical subject arrangements provided in most library classification schedules. For example, the term "artery" may be included under "heart system," which may be included under "organs of the body," which may in turn be included under "physiology," and so on. The second class of relations may consist of real dependence relation between two terms, such as the cause-effect relation between "poison" and "death," or it may be a formal relation, such as when two normally unrelated words are identified as equivalent in a given context, or are joined by a coordinating conjunction.

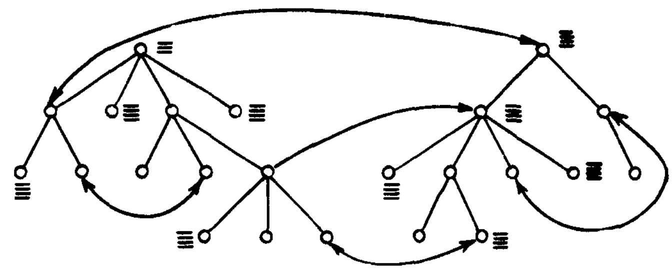
In order to perform a reasonably effective content analysis, it is thus useful to take into account not only the presence of certain words in a text, but also the principal analytic and synthetic word relations. If this is to be accomplished without exhaustive semantic analysis of each word, it is necessary to avail oneself of a number of auxiliary aids. The following structures are useful:

(a) a hierarchical arrangement between certain subject categories as provided by many library classification systems;

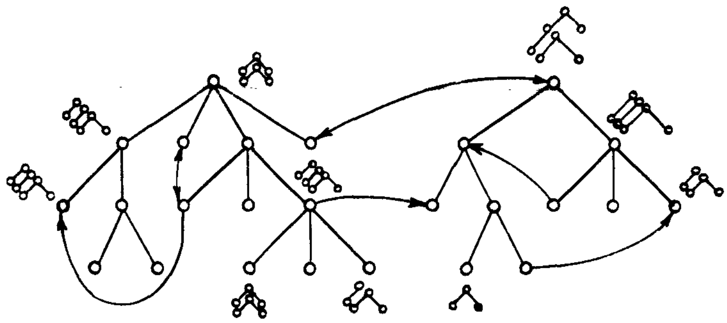
(b) a set of cross references between related subjects such as those specified by many classified or alphabetized word indexes;

(c) a set of related or synonymous words attached to each subject category to identify terms which may be used in similar contexts.

The hierarchical subject arrangement identifies some of the generic relations, and the cross references and lists of related words specify the dependence and equivalence relations between terms. The cross references are analogous to the "see also" references available in many library classification systems, and the related word lists are similar to the "see" references.



(a) Abstract Hierarchical Structure with
Cross References and Synonym Lists



(b) Abstract Hierarchical Structure with
Cross References and Criterion Trees

Key

⟶ cross reference

—o generic (inclusion) relation

o keyword or phrase with category indicator

≡ synonym list

criterion trees

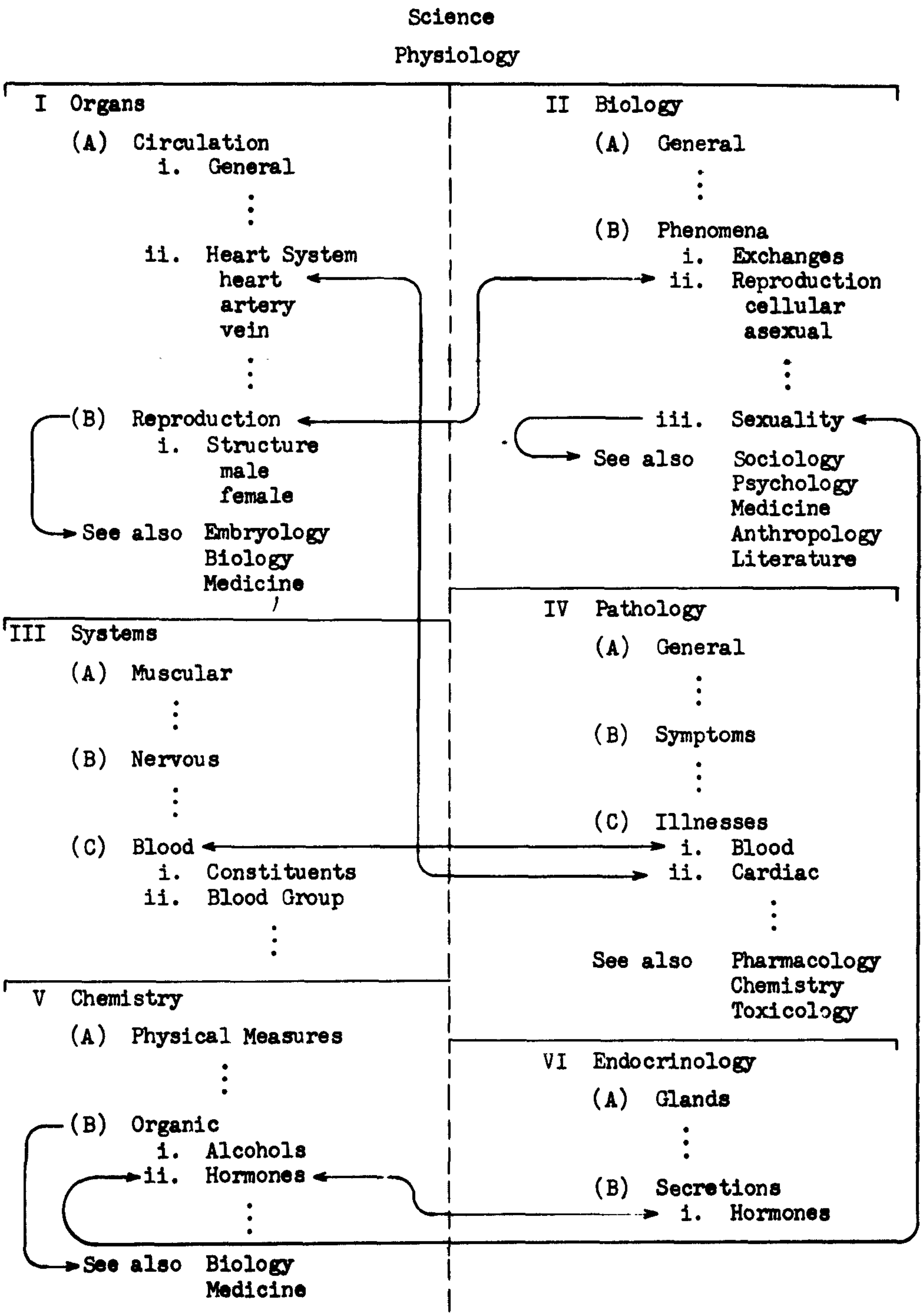
Hierarchical Structures

Figure 1

The complete arrangement is represented abstractly by the structure of Fig. 1(a). Each subject category is denoted by one or more nodes in the structure. Inclusion relations defined for certain subjects are represented by nonhorizontal branches between the corresponding nodes, the "included" subject appearing on a physically lower level on the page than the "including" subject. Cross references are represented by directed lines between specified nodes, and the related word lists by sets of short parallel line segments.

Consider as an example the hierarchical structure shown in Fig. 2 which includes certain terms from the field of physiology. One of the cross references connects the term "heart" considered as an organ to the term "cardiac illness" classified under pathology. It is clear from Fig. 2 that a given term may appear in various places within the hierarchical arrangement. This is true in particular when a subject may be considered from a variety of viewpoints, in which case different generic relations and different related word lists normally apply. The various subject classification provided by the Library of Congress (LC) classification system for the term "sexuality" are shown as an example in Fig. 3. Sample related word lists are included in Fig. 3 as are the LC subject indicators.

A storage arrangement of the type shown in Fig. 1(a) lends itself to reasonably effective retrieval procedures. Indeed, document identifications and request identifications may easily be normalized before being used by substituting for each original identifier the subject indicator corresponding to the next higher node in the hierarchical arrangement. Documents can then be classified into subject categories, and document clusters of similar documents can be generated, based not on the words originally extracted from the documents or on the terms originally chosen by a variety of more or less reliable methods, but based on the subject categories corresponding to the nodes within the hierarchies. The normalization procedure will thus make it easier to match documents and requests



Excerpt from Classification Schedule with Selected Cross References

Figure 2

Sexuality

1. As a social science (H), including statistics (HB), family problems (HQ), social pathology (HV), etc; marriage, love, abortion, ethics, men, women, bed, birth, birth control,...	4. Medical and health aspects (R) including gynecology (RG), medical practice (RC), public health (RA), etc; instinct, operation abortion, hygiene, contraception,...
2. From the physiological viewpoint (Q), including sexual organs (QL,QM), reproduction (QP,QH), etc; sex organs, reproduction, instinct, puberty, anatomy heredity,...	5. From the point of view of cultural differences among races (G), including ethnology (GN) and folklore (GR); marriage, folklore, incest,...
3. Psychological and religious aspects (B), including reproduction (BF), moral theology (BX), sex worship (BL); reproduction, Oedipus complex, sex worship, emotion, passion, excitement, mystery,...	6. From the point of view of development in the literature (P), including literary history (PN), and English literature (PR); magazine, Freud, sex crimes,...
	7. As an art (N), including erotica, etc.

Sample Related Word Lists with
Library of Congress Subject Indicators

Figure 3

for documents since the use of related terms will always refer back to the same subject identified either through the related word lists or by the cross-referencing process.

The retrieval process will also benefit from various other possible transformations. Retrieval requests may, for example, be broadened by replacing the original terms by new terms which appear on a physically higher level in the hierarchy, and by following up the cross references.

Contrariwise, requests may be narrowed or refined by including terms which appear on a physically lower level. The matching procedure may also be extended by considering as equivalent any term within the same list of related words, or any term within a given "distance" from some other term in the hierarchy.

It is often useful to include as part of the normal retrieval structures not only individual disconnected words or terms, but also word pairs or triples, or indeed complete phrases and sentences. A document might, for example, be identified more closely by a set of phrases than by a set of individual words. Each list of related terms included in the hierarchy of Fig. 1(a) might therefore be supplemented or replaced by a "list of related phrases" which identifies the subject indicator of the corresponding node. It is convenient to represent the syntactic structure of each phrase in tree form, in such a way that each word is represented by a node of the tree and the syntactic dependencies by the branches of the tree.[1,2] The trees corresponding to the various phrases attached to a given node in the hierarchy are called criterion trees, and their addition to the system gives rise to the structure of Fig. 1(b).

A sample set of related phrases is shown in Fig. 4. The examples of Fig. 4 refer to the subject "Philosophy of Education" and include phrases such as

liberal [education],
[education] for *,
[controversy] in [education],

-	:	[goal] (n.)
-PR	:	of
-PØ	:	[educational] (n.)
1...A:		(adj.)
1...	:	[education], means
-A	:	(adj.)
-	:	approach (n.)
-PR	:	to
-PØ	:	[education] (n.)
-	:	[education] (n.)
-PR	:	for
-PØPM:		-
-A	:	liberal (adj.)
-	:	[attempt], [suggestion] (n.), [education] (n.)
-S	:	[education] (n.)
-V$	:	[desirability]
-VDVR:		to
-VDV	:	- (v.)
-S	:	[education] (n.)
-V$	:	[desirability]
-ØVR	:	to
-OV	:	- (v.)

-S	:	[education] (n.)
-Ø	:	[world]
-V$	:	[desirability]
-V	:	[assist] (v.)
-Ø	:	[public]
-	:	[controversy]
-PR	:	in, about, ever
-PØ	:	[education] (n.)
-	:	[development]
-PR	:	of
-PØ	:	[ability], [knowledge] (n.)
-Q	:	[education]
-Q	:	(n.)
-	:	[education] (n.)
-PR	:	for, in, of
-PØ	:	[ability]
-A	:	vocational
-	:	guidance, [education] (n.)
-V	:	[education]
-VPR	:	for
-V	:	[education]
-VPVR:		to
-VP	:	

Sample Criterion Trees for
"Philosophy and Aims of Education"

Figure 4

and so on. The bracketed items represent special synonym classes or sets of related terms, and the asterisk stands for any arbitrary word. For the class [education] it is thus possible to substitute any one of a number of terms such as "education", "school", "university", "college", "academy", "lecturing", "explaining", and so on.

As a result, each one of the criterion phrases of Fig.4 represents a number of actual word strings related to one of the subject indicators in the hierarchy. The individual words of each phrase are listed in Fig.4 with a special syntactic code, and the set of codes corresponding to any given phrase can be used to represent the syntactic tree structure of the phrase. Furthermore the set of syntactic codes can be produced automatically as output of a syntactic analysis routine, and it is possible automatically to transform the tree structure into the code structure and vice-versa.[3] The terms "criterion phrases" and "criterion trees" can therefore be used interchangeably.

The criterion trees can be used in conjunction with syntactic analysis techniques for the classification of documents. It is necessary to obtain only the syntactic tree form of the sentences in a document, and to match these analyzed document excerpts with the criterion trees. Wherever a match occurs, the subject indicator attached to the corresponding node of the hierarchical structures is assigned to the given document. This will be further detailed in the next section.

The techniques of language normalization and of request alterations which were previously mentioned in connection with the model of Fig. 1(a) can be used unchanged with the tree structures of Fig. 1(b). In addition to the generic and cross-reference substitutions previously described, the model of Fig. 1(b) also makes possible syntactic substitutions of various types before search requests are matched with document identifications. Various kinds of syntactic equivalences may, for example, be defined; phrases or words exhibiting specific syntactic indicators may be ignored in the matching process; and completely unspecified nodes may be admitted as part of the tree structures.

In the next section, machine programs are outlined which make use of the retrieval structures of Fig. 1 for the classification of information, for the generation of document clusters, and for the matching of document identifications with search requests.

3. Retrieval Procedures

A. The Quantitative Model

It is well known that word frequency counting procedures are being used extensively for the identification of document content and for a variety of other documentation tasks, including in particular, automatic indexing, automatic abstracting and the generation of word and document associations.[4] The basic procedure consists in performing a frequency count of all the words in a document, rejecting certain high-frequency function words, such as prepositions and conjunctions, combining varying forms of words with similar stems, and using the remaining high-frequency words to represent the document content. Each document is thus identified by a specific set of high-frequency words $W_1, W_2, \ldots, W_n$.

It is often desirable to use word groups instead of individual words for the identification of document content. Such word groups are generated by identifying those sets of words which tend to occur jointly in similar contexts. More simply, the assumption is made that if two high-frequency content-words co-occur in several sentences of a text, they are related in some sense, and may therefore be grouped.

A typical method for the generation of word groups is:

(a) construct a word-sentence incidence matrix C which lists content words against sentences; matrix element C_j^i is defined to be equal to n if and only if sentence j contains word i exactly n times;

(b) define a coefficient of similarity between words based on frequency of co-occurrence between pairs of words in the sentences;

(c) generate a word-word similarity matrix R which exhibits all similarity coefficients between pairs of content words;

(d) define word groups corresponding to those word pairs whose associated similarity coefficient is greater than some stated threshold value.

A typical word-sentence incidence matrix C is shown in Fig. 5(a). To obtain a coefficient of similarity between two words based on the frequency of co-occurrence in the various sentences of a document, it is only necessary to perform a pairwise comparison of the corresponding rows of $\underline{C}$. A useful coefficient of similarity between rows of a numeric matrix is the cosine of the angle between the corresponding m-dimensional vectors.[5] The similarity coefficients can be displayed in an $n \times n$ symmetric word similarity matrix $\underline{R}$, where the coefficient of similarity $\underline{R}_j^i$ between word W_i and word W_j is

$$\underline{R}_j^i = \underline{R}_i^j = \frac{\sum_{k=1}^{m} \underline{C}_k^i \underline{C}_k^j}{\sqrt{\sum_{k=1}^{m} (C_k^i)^2 \sum_{k=1}^{m} (C_k^j)^2}}$$

A typical word-similarity matrix $\underline{R}$, corresponding to the word-sentence matrix $\underline{C}$, is shown in Fig. 5(b). Since $\underline{R}$ is symmetric, it is necessary to scan only the right (or left) triangular part in order to detect word pairs with large similarity coefficients. The word grouping procedure may be refined by various methods, including in particular:

(a) the use of a normalizing procedure which deletes word suffixes and combines the various forms of words with identical stems;

(b) the use of a dictionary of synonyms or thesaurus, which would permit the replacement of each high-frequency word by the corresponding thesaurus head;

(c) the generation of complete phrases, instead of only word pairs, by extracting from the text the related word pairs together with their context.

$$
\begin{array}{r|ccccc}
\text{Sentence} \diagdown \text{Word} & S_1 & S_2 & \cdots\cdots\cdots & & S_m \\
\hline
W_1 & \underline{C}^1_1 & \underline{C}^1_2 & \cdots & \underline{C}^1_j & \cdots \quad \underline{C}^1_m \\
W_2 & \vdots & & & \vdots & \vdots \\
\cdot & \underline{C}^i_1 & \underline{C}^i_2 & \cdots & \underline{C}^i_j & \cdots \quad \underline{C}^i_m \\
\cdot & \vdots & & & \vdots & \vdots \\
\cdot & & & & & \\
W_n & \underline{C}^n_1 & \underline{C}^n_2 & \cdots & \underline{C}^n_j & \cdots \quad \underline{C}^n_m
\end{array}
= \underline{C}
$$

(a) Typical Word-Sentence Incidence Matrix $\underline{C}$

($\underline{C}^i_j = n \Leftrightarrow$ Sentence S_j contains word W_i exactly n times)

$$
\begin{array}{r|cccc}
\text{Word} \diagdown \text{Word} & W_1 & W_2 & \cdots & W_n \\
\hline
W_1 & R^1_1 & R^1_2 & \cdots & R^1_n \\
W_2 & \vdots & & & \vdots \\
\vdots & & & & \\
W_n & R^n_1 & R^n_2 & \cdots & R^n_n
\end{array}
= \underline{R}
$$

(b) Typical Word-Word Similarity Matrix R

$$
\left(\underline{R}^i_j = \underline{R}^j_i = \frac{\sum_{k=1}^{m} \underline{C}^i_k \underline{C}^j_k}{\sqrt{\sum_{k=1}^{m} (C^i_k)^2 \sum_{k=1}^{m} (C^j_k)^2}} \right)
$$

Matrices Used for Word Grouping

Figure 5

If it is desired to generate <u>document associations</u> or <u>document clusters</u> instead of word associations, the same procedures can be used with some slight modifications. Instead of starting with a word-sentence matrix $\underline{C}$, as shown in Fig. 5(a), it is now convenient to construct a word-document matrix $\underline{F}$, listing frequency of occurrence of word W_i in document D_j. Specifically, $\underline{F}_j^i = n$ implies that document D_i contains word W_j exactly n times.

Document similarities can now be computed as before by comparing pairs of rows, and obtaining similarity coefficients based on the frequency of co-occurrence of the content words included in the given document.[5] This procedure generates a document-document similarity matrix which can in turn be used for the generation of document clusters, for example, by defining a cluster as including all those documents whose similarity coefficients with all other documents in the same cluster exceed a given threshold value.[6]

Consider now the problem of document retrieval in the present context. Specifically, it may be desired to identify those documents whose lists of high frequency content words exhibit similarities with the terms used in a given search request. The preceding model can then be used unchanged by adding to the $m \times n$ document-word matrix $\underline{F}$ a special row $\underline{F}^{m+1}$ which includes the terms used in specifying the search request. Specifically, $\underline{F}_k^{m+1}$ is set equal to w if term W_k is used in the search request with weight w; if word W_k is not used in the given search request $\underline{F}_k^{m+1}$ is set equal to 0. If no weights are specified by the requestor, the values of the elements of row $\underline{F}^{m+1}$ are of course restricted to 0 and 1.

A typical document-word matrix $\underline{F}$ of dimension $(m+1) \times n$ is shown in Fig. 6. To obtain the set of all "relevant" documents, it is only necessary to compute the similarity coefficients between row $\underline{F}^{m+1}$ on the one hand, and each of the other document rows $\underline{F}^1, \underline{F}^2, \ldots, \underline{F}^m$, on the other. The procedure previously outlined can be followed, in that the set of relevant documents can be defined to include all those documents whose similarity coefficients with the search request exceed a given threshold value. The terms used in the search request can also be normalized by the thesaurus look-up method described previously for the high-frequency text excerpts.

Document \ Term		W_1	W_2	$\ldots$ W_n
Document	D_1	$\underline{F}_1^1$	$\underline{F}_2^1$	$\ldots$ $\underline{F}_n^1$
	D_2	$\underline{F}_1^2$	$\underline{F}_2^2$	$\ldots$ $\underline{F}_n^2$
	$\vdots$			
	D_m	$\underline{F}_1^m$	$\underline{F}_2^m$	$\ldots$ $\underline{F}_n^m$
Search Request	D_{m+1}	$\underline{F}_1^{m+1}$	$\underline{F}_2^{m+1}$	$\ldots$ $\underline{F}_n^{m+1}$

$= \underline{F}$

Document-Term Matrix $\underline{F}$ including Extra Row
Specifying Search Request

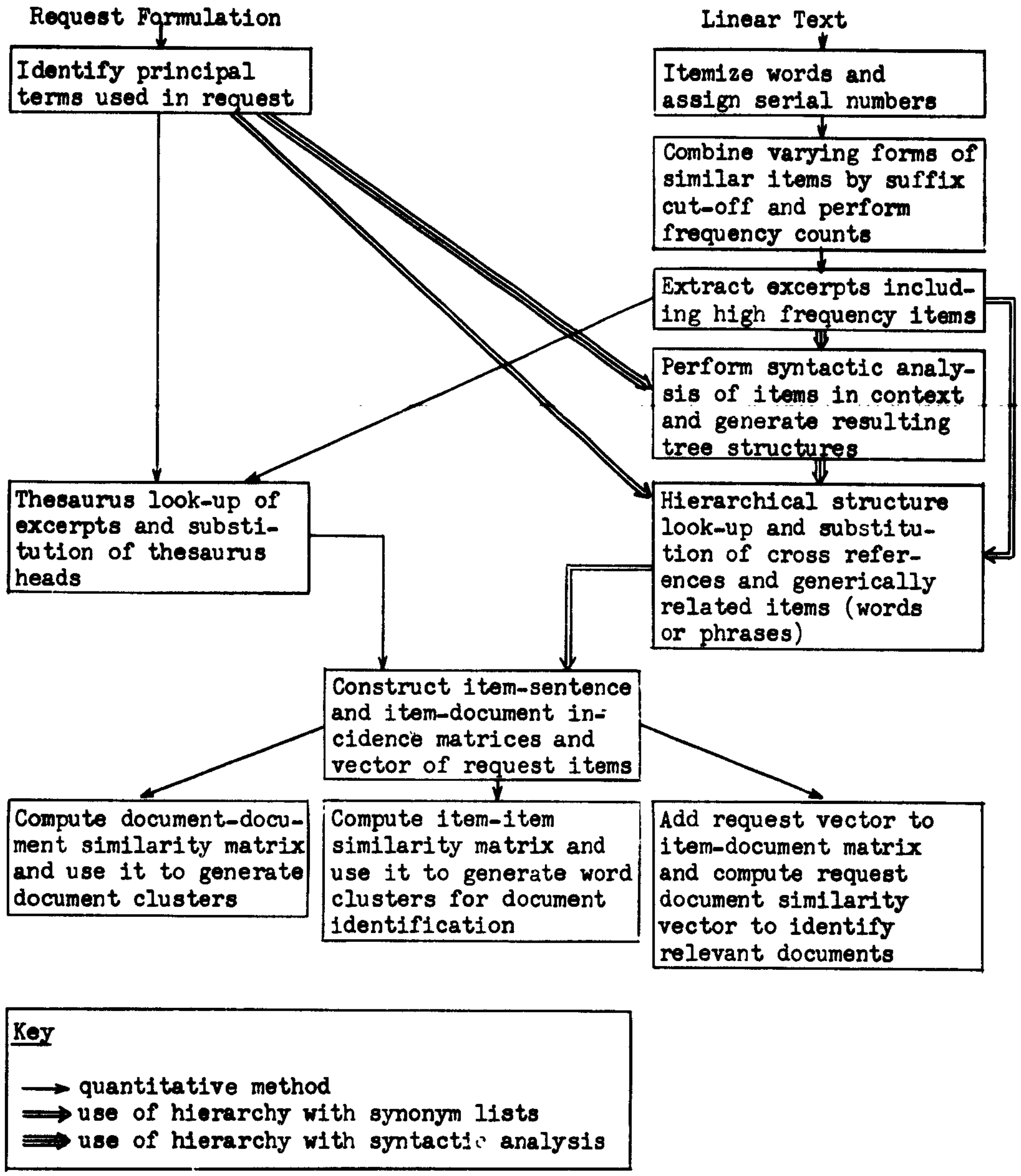
Figure 6

The complete procedure is summarized by following the simple arrows between the boxes of Fig. 7. The processing specified for the search requests is seen to be parallel to that used for the documents themselves. Furthermore, the machine program which identifies word clusters by row correlation of the word-sentence incidence matrix, can be used unchanged for the identification of document clusters, and also for the identification of relevant documents by substituting the document-word matrix for the word-sentence matrix.

B. The Hierarchical Model with Synonym Lists

The quantitative model is based primarily on the words which occur in the individual documents: documents are identified by sets of high-frequency words; they are grouped or clustered by using similarities between these sets of high-frequency words; and they are retrieved by comparing words extracted from search requests with high-frequency words extracted from documents.

In contrast, now consider a system which includes semantic associations as represented by the structure of Fig. 1(a). Each node in the hierarchical structure represents a <u>subject category</u>, and a given word occurring in a text may therefore be associated directly with one or more nodes in the hierarchy if it identifies such a subject category. More commonly, a word occurring in a text will be associated with one or more of the related word lists which are attached to the nodes of the hierarchy.



Simplified Procedures for the Generation of Document Identifications, Document Clusters and Relevant Documents

Figure 7

Given a high-frequency word occurring in a text, the structure of Fig. 1(a) can be used to identify the following types of semantic associations:

(a) the subject category or categories with which the given word is immediately associated;

(b) the subject category or categories which are generically superior, that is, located on a level immediately above, and linked to the nodes identified in part (a);

(c) the subject category or categories which are generically inferior, that is, located on a level immediately below, and linked to the nodes identified under (a);

(d) the first-order cross references, that is, all subject categories directly cross-referenced by the nodes identified in part (a);

(e) the set of all related words or phrases associated with the nodes identified under (a)

It is clear that the procedure for document identification, clustering, and retrieval can be improved considerably over those which are possible with a strictly quantitative model. Consider first the document identification: the word frequency lists identifying the various documents can be replaced with subject category lists by a look-up procedure in the hierarchical structure, and the corresponding subject category indicators can be arranged in frequency order to identify the documents. The procedure is described by means of an example.

Figure 8 contains a list of high-frequency words and phrases for a sample document. When each of the high-frequency words is looked-up in a hierarchical structure of the type shown in Fig. 1(a), and the corresponding category list is produced as shown in Fig. 9, it may be noticed that a total of 15 different subject categories are identified. These 15 categories, in turn, lead to 7 main subject headings by substitution of generically superior terms from the hierarchy: sociology, medicine, science, military science, technology, philosophy, and agriculture. The subject categories and the corresponding codes shown in Fig. 9 are taken from the Library of Congress classification schedules.

Heart Attack
Treatment
Cholesterol Level
Blood
Young Men
Suggest
Female Hormones
Female Sex Hormones
Sex Hormones
Effect of Sex Hormones
Reticulo-Endothelial System (RES)
Stimulation of RES
Animal
Medical Chemistry
Physics

High Frequency Words or Phrases in Sample Document†

Figure 8

† See H.P. Luhn, "Auto-Encoding of Documents for Information Retrieval Systems," Modern Trends in Documentation, M. Boaz (editor), Pergamon Press, 1959.

H (Sociology)	HV	Young Men (social pathology)
	HQ	Youth (family, marriage, women)
	HF	Suggestion Systems (commerce)
	HQ	Female
	HQ	Men
R (Medicine)	RC	Heart Attack (medical practice)
	RC	Blood Diseases (medical practice)
	RC	Sex Hormones (endocrinology)
	RS	Medical Chemistry (pharmacy)
	RB	Blood Chemistry (pathology)
	RM	Stimulants (therapeutics)
	R	Laboratory Animal (research)
Q (Science)	QP	Cholesterol (physiological chemistry)
	QP	Blood (physiology)
	QP	Blood Chemistry (physiology)
	QP	Sex Hormones (physiology)
	QP	Laboratory Animal (comp. physiology)
	QP	Cholesterol (chemistry)
	QH	Physical Research (physics)
	QP	Reticulo-Endothelial System (physiology)
	QL	Laboratory Animal (zoology)
U (Military Science)	UG	Attack (military engineering)
T (Technology)	TA	Leveling (engineering)
B (Philosophy, Religion)	BJ	Young Men (ethics)
S (Agriculture)	SF	Laboratory Animal (animal culture)

Subject Categories Obtained for Sample Document

Figure 9

The subject heading "military engineering," for example, is obtained by cross reference from the high-frequency word "attack," similarly, "agriculture" is obtained by cross reference from "laboratory animal." Four of the seven major subject headings can be eliminated because they are identified with insufficient frequency. The remaining major subject headings are "science," "medicine," and "sociology," respectively, and the principal subject categories in decreasing frequency order are

physiology,
medical practice,
family and marriage,

and so on.

The list of subject categories produced by the look-up procedures in the hierarchical structure can be used not only for purposes of document identification, but also for document clustering and retrieval. Specifically, lists of high-frequency words or phrases together with the relevant subject categories are obtained as before from the hierarchical structure for a given document collection. These phrases are then listed in a document phrase matrix $\underline{G}$, similar in nature to the document-term matrix $\underline{F}$ of Fig. 6. However, whereas each applicable term is listed only once and is thus represented by a single matrix column in $\underline{F}$, all semantic contexts are provided for each phrase in $\underline{G}$ by listing the phrases as many times as there are applicable subject categories. A given phrase may therefore be represented in $\underline{G}$ by several matrix columns. Thus if matrix element $\underline{G}^i_{jk} = n$, it is implied that a given phrase P_{jk} occurs n times in document D_i; the first subscript attached to the phrase refers to the phrase number, and the second to the subject category of the phrase. Figure 10 shows a typical document-phrase matrix $\underline{G}$ in which the phrases have been ordered in such a way that all items belonging to one and the same subject category appear in adjacent matrix columns.

$$
\begin{array}{c|cccccc}
\text{Phrase} \diagdown \text{Document} & \overbrace{P_{1A} \quad P_{2A} \quad \cdots}^{\text{Category A}} & & & \overbrace{P_{1B} \quad P_{3B} \quad \cdots \quad P_{n2}}^{\text{Category B}} & & \\
\hline
\text{Document} \left\{ \begin{array}{c} D_1 \\ D_2 \\ \cdot \\ \cdot \\ \cdot \\ D_m \end{array} \right. & \multicolumn{6}{c}{\left(\begin{array}{cccccc} \underline{G}^1_{1A} & \underline{G}^1_{2A} & \cdots & \underline{G}^1_{1B} & \underline{G}^1_{3B} \cdots & \underline{G}^1_{n2} \\ \underline{G}^2_{1A} & \underline{G}^2_{2A} & \cdots & \underline{G}^2_{1B} & \underline{G}^2_{3B} \cdots & \underline{G}^2_{n2} \\ & & & & & \\ & & & & & \\ & & & & & \\ \underline{G}^m_{1A} & \underline{G}^m_{2A} & \cdots & \underline{G}^m_{1B} & G^m_{3B} \cdots & \underline{G}^m_{n2} \end{array} \right) = G}
\end{array}
$$

Document-Phrase Matrix with Specified Subject Categories

Figure 10

Document clusters can now be generated by using the procedures previously described for the quantitative model, except that matrix $\underline{G}$ is substituted for F. Specifically, a document-document similarity matrix is obtained by pairwise comparison of rows of $\underline{G}$, and clusters of documents are defined as a function of the document similarity coefficients as before. The procedure for document retrieval is also parallel to that previously described in that an extra row specifying the search request is added to the phrase-document matrix $\underline{G}$. Row $\underline{G}^{m+1}$ is then compared against all other document rows to obtain an n-dimensional vector of similarity coefficients, and documents with sufficiently large coefficients are considered to be relevant to the given request.

The request vector $\underline{G}^{m+1}$ is constructed by substituting for the words used in a given search request the category indicators and first-order cross references found in the hierarchical structure, and by labeling the appropriate matrix elements in row $\underline{G}^{m+1}$. The value of the individual elements may be either 0 or 1, or alternatively a weight w may be assigned as previously explained. The row correlation program which is used to identify the relevant documents, as well as the word and document clusters is the same as that used for the quantitative model.

The complete procedure is again summarized in the chart of Fig. 7 where the double arrows are to be substituted for the simple arrows wherever an alternate path is provided.

C. The Hierarchical Model with Tree Structure

The preceding model, even though providing for the addition of semantic associations by introducing new words related analytically or generically to the ones originally used in search requests and documents, is still primarily based on the word as a unit of information. In particular, two documents are assumed to be related, if the words originally contained in them, or their respective substitutes, can be matched. The same is true for the matching of requests with documents. Phrases, rather than words, can of course be used, as illustrated by the document-phrase matrix of Fig. 10. However, in the hierarchical look-up process used up to now it is assumed that two phrases can be matched if they contain one, or two,... or n words which are identical (except for the permitted semantic substitutions). The syntactic structure of the phrase itself has not so far been taken into account; this makes it possible to match distinct concepts such as for example, "school children," "children in school," and "school for children."

The internal structure of the phrases can be utilized, and a more accurate matching process can be obtained by providing syntactically analyzed phrases and requiring matches of the individual words as well as of their syntactic structure. This is accomplished by using the model of Fig. 1(b) in which the sets of related phrases or criterion trees are listed in syntactic tree form. The principal modifications introduced in the over-all process by the structural analysis are shown in Fig. 7 by means of boxes interconnected by triple arrows.

The process starts as before by a word frequency count and a normalization procedure designed to combine the varying forms of words with similar stems. Excerpts containing a large number of high-frequency words are then extracted from the original documents. Instead of extracting only individual words, it is convenient to use larger units, such as clauses or complete sentences. A syntactic analysis is then performed of both the search requests and of the extracted parts of the documents, and the analyzed output is represented in tree form.[1,2,3] Instead of physically manipulating the two-dimensional tree form, it is convenient to attach to each word of the text a syntactic code which exhibits the syntactic function of the given word and also the syntactic dependency structure. The principal words of the sentence, such as main verb, subject, object, and so on, are assigned short strings of code characters, and the remaining words which are syntactically dependent on them are assigned longer strings. Thus, the length of the character string attached to a given word signifies "depth" of syntactic dependence and therefore depth of location in the corresponding tree structure.[3] A sample excerpt is shown in column 1 of Fig. 11 and the corresponding syntactic character strings are given in column 3 of Fig. 11.

1	2	3	4	5
Excerpt from Original Text	Subject Category	Coded Syntactic Tree Structure	Matching Code String	Matching Criterion Tree
Many		1S		
of		1SPQ		
the		1SPØA		
new		1SPØA		
citizens	[person]	1SPØ		
began		1V		
to		1VPR		
perceive	[consider]	1VPV		
the		1VPØA		
schools	[education]	1VPØ	-V :	
as		1VPVPR	-Ø : = { -Q:	[education]
a		1VPVPØA	-PR: { -Q:	(n.)
means	[means]	1VPVPØ	-PØ:	
by		1VPVPØ7VPR		
which		1VPVPØ7VPØ		
their		1VPVPØ7SA		
children	[person]	1VPVPØ7S		
might	[desirability]	1VPVPØ7VX		
have		1VPVPØ7V		
a		1VPVPØ7ØA		
fuller		1VPVPØ7ØA		
life	[world]	1VPVPØ7Ø		
.		1.		
Hence		1VD		
,		1,		
in		1VPR		
addition		1VPØ		
to		1VPØPR		
educating	[education]	1VPØPØG	- :	[education]
for		1VPØPØGPR		for
intelligent		1VPØPØGPØA	-PR:	(n.)
citizenship	[world]	1VPØPØGPØ	-PØ:	
⋮				

Sample Tree Matching Output

Figure 11

Following the syntactic analysis, the words and associated code strings extracted from documents and search requests are compared with the criterion trees included in the hierarchical structure, and the semantically associated information, including cross references and generically related items, is extracted from the hierarchy whenever a match is found. The operation required to find matches between word strings on the one hand, and criterion strings on the other, is however not as simple as the word matching procedure used before. Indeed, only a small number of general criterion trees are provided, each of which may be matched with a multiplicity of actual word strings. This is achieved by associating general terms, such as subject categories, with the nodes of the criterion trees, and by leaving the syntactic structure of the criterion trees largely unspecified. "Variable" characters are therefore introduced in the syntactic code strings associated with the criterion trees, and each "variable" character can replace a specified set of "fixed" syntactic characters of the type shown in Fig. 11. The dashes used in the code strings of the criterion trees shown in Fig. 4 are examples of such variable characters.

In order to find a match between a given word string extracted from a document and its associated sequence of fixed syntactic codes, and a given criterion string and its associated sequence of fixed or variable code characters, the following operations are therefore necessary.

Each word in the word string must be looked up in the hierarchical structure, and replaced by the corresponding subject heading (this operation is equivalent to the thesaurus look-up described in connection with the quantitative model).
Each variable code character associated with each term in the criterion tree must be replaced by a sequence of fixed characters.

A match is then obtained between a given word of a word string and a given term of a criterion string if the associated subject headings as well as the associated fixed character strings are identified. Furthermore, a complete criterion tree will match a complete sentence extracted from a text if there exists a sequence of fixed characters such that each term in the criterion tree matches at least one word in the sentence in the same linear order.

Given any sentence, it is necessary to test in order all the criterion trees which contain any of the relevant subject categories. Various strategies are possible for matching sentences (word strings) and criterion trees. The most immediate method consists in taking the first term in the first criterion tree and comparing it in turn with each subject category associated with the words in the sentence until a match is found. The next term is then taken and matched against all the subsequent words in the same sentence, and so on, until either the terms in the criterion tree or the subject categories associated with the sentence are exhausted. If the sentence is exhausted before the criterion tree, no match can possibly exist, and the next criterion tree must be processed. If the criterion tree is exhausted before the sentence, a match may or may not exist. The procedure will in general permit the matching of long sentences with short criterion trees, since many of the words in a sentence are disregarded in the process. Thus, it is always possible to disregard subordinate clauses, adverbial or prepositional phrases, or any other sentence parts which may not be included in any given criterion tree.†

The strategy used to assign sequences of fixed code characters to the variable characters associated with the criterion trees is similar to the previously described method for matching word and criterion strings. The variable characters are processed from left to right and are initially assigned the shortest possible fixed character strings. This process is continued until either the given character string is exhausted or it is determined that the complete assignment of fixed character strings to all variable characters is impossible. In the latter case a new assignment is tried by using successively longer fixed strings to replace the variable ones. As an example, the fixed character string "1SPØA" matches the string "-S*A", by replacing the variable characters "-" and "*" respectively by the fixed characters "1" and "PØ".

Two matches are found when the word string illustrated in Fig. 11 is compared with the criterion trees of Fig. 4. The subject headings which replace some of the words in the sentence are shown in column 2, and

† A different matching procedure which uses the graph theoretic properties of the syntactic trees and finds the subgraphs of a given graph instead of proceeding on a node-by-node basis has been programmed by E. H. Sussenguth.[7]

the matching code strings and criterion trees are shown in columns 4 and 5 of Fig. 11 respectively.

Following the look-up procedure in the hierarchical structure and the comparison of sentences with criterion trees, the item-sentence and item-document incidence matrices are constructed as before. Each item is now a criterion tree instead of a single word, and a given sentence or document is characterized by the matching criterion trees contained in it. The remainder of the procedure outlined in Fig. 7 is also followed unchanged in that the item-item similarity matrix is used to generate item clusters; the document-document correlation matrix similarly generates document clusters, and the item-document matrix augmented by the request vector determines documents which are relevant to a given search request.

The complete process can of course be kept more or less flexible by assigning completely variable code strings to the terms of the criterion trees, or alternatively by not permitting any variable code characters at all. Similarly the criteria used to match word strings with criterion strings may be made more or less restrictive.

4. Summary

Three models have been described for automatic document retrieval systems. The first one is based largely on the words used in the documents and search requests. The second adds semantic associations by introducing a hierarchical structure consisting of generic inclusion relations, cross references, and lists of related words. Finally, the last model also exhibits syntactic associations through the introduction of syntactic tree structures.

The programs needed for document clustering and retrieval become increasingly complex as semantic associations and syntactic structure are taken into account. The following programs are specifically required:

(a) itemization of linear text;
(b) normalization of word forms by suffix cutoff and condensation of words with identical items;
(c) word frequency count;
(d) extraction of words, phrases, or sentences including high-frequency words;
(e) thesaurus look-up process to replace a given item by the corresponding subject heading;
(f) construction of item-sentence and item-document incidence matrices;
(g) row-correlation process to obtain word and document clusters, and to compute a relevance index for documents.

The hierarchical models with synonym lists and criterion trees are based in addition on the following programs:

(h) a hierarchical look-up process designed to identify generically superior or inferior items, cross-referenced items, and lists of related items;
(i) a syntactic analysis procedure which furnishes the syntactic tree structure of each sentence;
(j) a tree matching process which compares a fixed tree structure representing a word string with a variable structure representing the criterion trees.

A useful comparison of the three models is difficult to perform, since the usual criteria of efficiency including time of execution and internal computer storage requirements are obviously not sufficient by themselves. Some measure of goodness or accuracy of retrieval is also needed. If speed and storage requirements alone were of importance, the quantitative model would clearly be most efficient. Comparative tests with actual document collections are needed to determine whether the more complicated hierarchical look-up and syntactic tree matching processes provide a sufficient improvement in retrieval accuracy to warrant the cost in both time and equipment required by the hierarchical models.

Acknowledgment

This study was supported in part by the Air Force Cambridge Research Laboratories and in part by Sylvania Electric Products, Inc.

REFERENCES

1. Plath, W., "Automatic Sentence Diagramming," First International Conference on Machine Translation and Applied Language Analysis, National Physical Laboratory, Teddington (September 1961).
2. Salton, G., "The Manipulation of Trees in Information Retrieval," Communications of the ACM, Vol. 5, No. 2 (February 1962).
3. Kuno, S., "Sentence Structure Diagramming," Mathematical Linguistics and Automatic Translation, Reports to the National Science Foundation by the Computation Laboratory of Harvard University, Cambridge, Massachusetts, NSF-9, Sec. I, part 6 (to appear).
4. See for example, Luhn, H. P., "Auto-Encoding of Documents for Information Retrieval Systems," in Modern Trends in Documentation, M. Boaz (editor), Pergamon Press (1959).

5. Salton, G., "Some Experiments in the Generation of Word and Document Associations," Fall Joint Computer Conference, Philadelphia (December 1962).

6. See also, Needham, R.M. and Parker Rhodes, A.F., "The Theory of Clumps," Cambridge Language Research Unit, Reports ML 138-139 (1960).

7. Sussenguth, E.H. Jr., "Graph Isomophism and Subgraph Testing Procedures," unpublished.

II. AUTOMATIC REFERENCE ANALYSIS

Margaret Thompson

1. Introduction

With the development of versatile character recognition equipment, it becomes increasingly possible to use as basic machine input unedited running text as it appears in books, magazines, or journals. Bibliographic references or citations constitute an integral part of most technical articles, and methods must be available for processing such bibliographic citations automatically. In particular, it becomes necessary to identify similar, or equivalent citations, and to rearrange a reference list into some specified order. The difficulties arising in this connection are largely due to the extreme variability in format and to the lack of standardisation which prevails in the publication of citations.

The present program takes bibliographic citations and automatically arranges them into a standard format in such a way that the various parts of the citation are unambiguously identified. These standardized citations can later be processed by sorting and matching procedures to identify similar citations and to effect various rearrangements.

2. General Categories of a Bibliographic Reference

All references can be manually separated into nine or fewer general categories. The present program handles this separation automatically with a minimum of manual preparation of the input data. The reference categories to be recognized are as follows:

(a) author(s), including initial(s), Jr., III, etc.,
(b) title of paper or book,
(c) title of journal,
(d) volume number,
(e) issue number,
(f) page number,
(g) name of publisher,
(h) city of publisher,
(i) year of publication.

All given input information is classified under one of these headings; however, one or more of these categories may be missing from a legitimate reference.

3. Automatic Category Selection

The first part of the program reads in the complete data from a single reference. Fields are pieces of data information ending with some punctuation mark. Categories contain one or more fields ending with some key punctuation mark. Categories are identified and formed in the order described in Part 2. From the complete reference source consecutive fields are examined to determine if the data fit into the description of the category being formed. It may happen that a part of a category is read and further fields are necessary to complete the category.

In the author category, certain defined formats are likely to begin the citation. The most common formats are tested first such as one or two initials preceding the authors last name, or the authors last name followed by one or two initials. The more subtle, less frequently occurring formats are then tested; in particular in a series of last names of authors, the word "AND" is searched for immediately before the final author.

A typical author category formation might be: first field contains an initial, "I."; the second field contains the authors last name, "NAME"; the third field contains the title of a paper, "Computers......Constants." Fields one and two are collected, put in a standardized format, and entered in the Author Table as "NAME, I." Field three did not contain "Jr." or another author. Therefore this is the first data field considered in formulating the next category. If more data are necessary for the second category, additional fields are read from the complete reference source. This procedure continues until all nine categories are formed or it is determined that one or more of the categories are vacuous. If a category identifier is found before the testing occurs, the pertinent data are identified and saved, until that particular category is being formed.

In certain cases, special words are also used to help identify the category to which a field belongs. For example, "vol" identifies the data with the volume number category.

If a blank word is read or all nine category tables include at least one entry, the reference has been fully processed. The program recycles to process the next reference in the same manner. See Appendix A for more detailed flow diagrams of entire program.

4. Conventions

Two types of restrictions are required: input and logical.

A. Input

To distinguish unmarked numerical fields and to overcome the limitations of the character set available on the IBM 026 keypunch machine, the following conventions are imposed on the input data:

(a) dollar sign ($) is inserted before any boldface. This sign indicates that all data up to the next punctuation mark are entered in the Volume Table;

(b) asterisk (*) replaces colons and semicolons. This will distinguish an otherwise unmarked (absence of special words, "vol" etc.) volume number and issue number from an unmarked page number. If there is a comma in the body of a title, the title category usually ends with a colon or semicolon. The replacement of the colon or semicolon by an asterisk will permit complete title selection in this situation;

(c) a series of two or more minus signs (---) replaces a dash. Only the 11-punch minus sign will be recognized;

(d) a single (') or double ('') apostrophe replaces a quote;

(e) Roman numerals are converted to arabic numbers.

B. Logical

The known logical limitations within the program itself are as follows:

(a) there must be an author and/or a paper (book) title for each reference processed;

(b) editors are treated as authors;

(c) translators, when given, are ignored. The original author(s) only are processed;

(d) occasionally a cited journal article is referred to only by page number, rather than by the actual title of the article. This journal title is listed in the "Title of Paper (Book)" Table. No entry is made into the "Title of Journal" Table. The journal title is substituted for the missing paper (book) title;

(e) a leading long dash beginning a reference indicates that this citation has the same author as the citation immediately preceding. For this reason, the first reference to be processed cannot begin with a dash.

5. Input Data

The following list represents some typical input examples as they appear after manual pre-editing. The titles have been shortened where a series of dots occurs.

1. Cook, C. E., "Modification ... Forms," Proc. 1958 Natl. Electronics Conf., pp. 108-1067.
2. Hildebrand, F. B. * Introduction to Numerical Analysis. McGraw-Hill, New York, 1956.
3. Cecil Hastings, Jr., Approximations ... Computers. Princeton University Press, 1955.
4. S. Schechter (Ed.), Nuclear ... Newsletter, New York University.
5. Cesaro, Ernesto * Einleitung ... Rechning; translated by G. Kawalewski. Teubner, Leipsig, 1922.
6. Memo from R. B. Reddy to A. G. Carlton, The Computer ... Constants, JHU Applied Physics Laboratory, April 6, 1954.
7. Copi, I. M., Elgot, C. C. and Wright, J. B., Realization ... Events. J. Assoc. Comput. Mach. 5 (1958), 181-196.
8. Courant, Freidricks, Lewy, G. B., Uber ... Physic, Math. Ann. 100, 32 (1928).
9. British Association for the Advancement of Science, Mathematical Tables, Cambridge Univ. Press, 1952.
10. ------, Generation ... Computers. Math. Tables ... Aids Comput. 11 (1957), 255-257.

Typical Input

TABLE 1

11. Encyclopaedia Britannica. 24 vols. 1944.

12. M. A. Acsel, ''The Effect ... Priorities,'' Opns. Res. 8, 730-733 (1960).

13. P. M. Morse and H. Feshback, Methods ... Physics, Part 1 and 2, McGraw-Hill, New York, 1953.

14. V. Nather and W. Sangren, ''Abstracts – N R Codes,'' Communications ... Machinery, 2*1 (January, 1959).

15. Aitken, A. C., Determinants and Matrices. Edinburgh * Oliver and Boyd, 1948.

16-17. P. A. M. Dirac, Proc. Roy. Soc., A 133, 60 (1931); Phys. Rev. 74, 817 (1948).

18-19. Rosenblatt, F., (a) The Perception ... Systems. Cornell Aero. Lab., Inc., Report No. VG-1196-G-1 (1958). (b) Two Theorems ... Perception. Cornell Aero. Lab., Inc., Report No. VG-1196-G-2 (1958).

20-21. P. A. M. Dirac, Proc. Roy. Soc., A 133, 60 (1931) * see also F. London, Superfluids, Wiley, New York, 1950, Vol. 1, p. 152.

22. Bickley, W. G., See Temple and Bickley

23. Samuelson, P. A., 'Iterative ... Roots,' Journ. of Math. and Phys. 28 (1949), 259-301.

24. -----, 'A Simultaneous ... Equation,' ibid. 242 (1950).

25. Letting Bills Pay Themselves, Business Week, October 27, 1956.

TABLE 1 (continued)

6. Processed Table Listings

The input examples given in Part 5 are listed in Table 2 as they appear as the result of the standardization procedure.

Author(s)	Title of Paper (Book)	Title of Journal	Vol. No.	Issue No.	Page No.	Publisher	City	Year
1. Cook, C. E.	Modification ... Forms	Proc. 1958			108-1067			
2. Hildebrand, F. B.	Introduction ... Analysis					McGraw-Hill	New York	1956
3. Hastings, C., Jr.	Approximations ... Computers					Princeton University Press		1955
4. Schechter, S. (Ed.)		Nuclear Codes Newsletter				New York University		
5. Cesaro, E.	Einleitung ... Rechning					Teubner	Leipzig	1922
6. Reddy, R. B.	Computer ... Constants					JHU Applied Physics Laboratory		1954, April, 6
7. Copi, I. M. Elgot, C. C. Wright, J. B.	Realization ... Events	J. Assoc. Comput. Mach.	5		181-196			1958

Category Tables After Processing

TABLE 2

Author(s)	Title of Paper (Book)	Title of Journal	Vol. No.	Issue No.	Page No.	Publisher	City	Year
8. Courant Friedricks Lewy	Über ... Physik	Math. Annals 100	32					1928
9. British Association for the Advancement of Science	Mathematical Tables					Cambridge University Press		1952
10. British Association for the Advancement of Science	Generation ... Computers	Math. Tables ... Aids Comput.	11		255-257			1957
11.	Encyclopaedia Britannica		24 vols.					1944
12. Acsel, M. A.	Effects ... Priorities, The	Opns. Res.	8		730-733			1960
13. Morse, P. M. Feshback, H.	Methods ... Physics		Part 1 and 2			McGraw-Hill	New York	1953
14. Mather, V. Sangren, W.	Abstracts - N R Codes	Communications ... Machinery	2	1				1959 January
15. Aitken, A. C.	Determinants and Matrices					Oliver and Boyd	Edinburgh	1948

TABLE 2 (continued)

Author(s)	Title of Paper (Book)	Title of Journal	Vol. No.	Issue No.	Page No.	Publisher	City	Year
16. Dirac, P. A. M.		Proc. Roy. Soc.	A 133	60				1931
17. Dirac, P. A. M.		Phys. Rev.	74	817				1948
18. Rosenblatt, F.	Perception ... Systems, The		Report No. VG-1196-G-1			Cornell Aero. Lab., Inc.		1958
19. Rosenblatt, F.	Two Theorems ... Perception		Report No. VG-1196-G-2			Cornell Aero. Lab., Inc.		1958
20. Dirac, P. A. M.		Proc. Roy. Soc.	A 133	60				1931
21. London, F.	Superfluids		1		1152	Wiley	New York	1950
22.								
23. Samuelson, P. A.	Iterative ... Roots	Journ. of Math. and Phys.	28		259-301			1949
24. Samuelson, P. A.	Simultaneous ... Equation, A	Journ. of Math. and Phys.	242					1950
25.	Letting Bills Pay Themselves	Business Week						1956, October, 27

TABLE 2 (continued)

A preliminary output is planned to test accurate category separation after processing of each reference. Each category of a reference will be listed on a separate line in the order described in Part 2. All category listings will be indented except the author, and spacing within a reference listing indicates that no information was given or processed for this particular category.

Input: Cook, C. E., "Modification ... Forms," Proc. 1958 Natl. Electronics Conf., pp. 108-1067.

Preliminary Output:

(a) Cook, C. E.

(b) Modification ... Forms

(c) Proc. 1958 Natl. Electronics Conf.

(d)

(e)

(f) 108-1067

(g)

(h)

Final Output: See Part 6, Table 2 and Part 8, Table 3

7. Explanation of Table Formats

A. Nine Categories

Authors

(a) Author's last name occurs first, followed by initials.

All examples.[†]

(b) Jr., III, etc. are included.

Example 3.

(c) Editors are treated as authors.

Example 4.

(d) Translators are ignored.

Example 5.

(e) Private communications, memos, etc., the originator (from) is considered the author and person(s) addressed (to) are ignored.

Example 6.

(f) If more than one author is cited, all authors are listed and the rest of the reference information is listed after last author. "AND" occurs before last author.

Example 7.

Exception no "AND". Only last names of multiple authors are given and the title is greater than one word.

Example 8.

(g) If special words (Staff, Association) indicating multiple authors occur, this field is considered the author.

Example 9.

[†] Examples refer to the reference numbers in Table 1 and Table 2.

(h) An initial long dash refers to previous author and all author information from previous reference is repeated. Examples 10, 24.

(i) A reference has no given author if the special words "Tables," "Encyclopaedia," "Dictionary," or "U.S." appear in the first field. Example 11.

<u>Title of Paper (Book)</u>

Book titles (Example 2) and titles of papers appearing in journals (Example 1) are listed in this category.

<u>Title of Journals</u>

This category lists all titles of journals (Example 2) and has no entries from books (Example 2).

<u>Volume Number</u>

(a) All information after a boldface indicator (#) up to the next punctuation mark is considered the volume number.

(b) All integers preceding a colon indicator (*) are considered the volume number.

(c) An occurrence of one of the following special words with or without prefix or suffix integers will indicate a volume number entry: vol., v., part, pt., diary, report, rept., Report no., paper, technical note, tech. note, T. N., thesis, doctoral thesis. Examples 11, 13.

Issue Number

(a) If a numerical field occurs after the volume number or colon indicator (*) and is not the date or page number, it is entered in the Issue Number Table.

(b) An occurrence of one of the following special words with or without prefix or suffix integers will indicate an issue number entry: edition, ed., series, number, no.

Page Number

(a) An occurrence of one of following special words with or without prefix or suffix integers will indicate a page number entry: p., pp., section, chapter(s), chapt., monograph.

(b) Any number that has not been processed up to now, is not the date and has a dash, is considered the page number.

Example 7.

Publisher and City

Publisher and city occur only for books.

(a) Any field of nonnumeric characters (excluding months and N.D. no date), not processed up to now, is considered the publisher and/or city of publication. Most references have the publisher first, followed by the city of publication.

Example 2.

(b) Occasionally the city appears before the publisher, therefore a small table of such cities will be searched prior to calling this field the publisher.

Example 15.

(c) An occurrence of one of the following special words with or without prefix or suffix data will indicate a publisher and city entry: laboratory, lab.

Year

(a) If "N.D." (no date) is present there is no date entry.

(b) Any four character integer not processed up to now, beginning with 18-- or 19--, is considered the year of publication.

Example 2.

The month and day of month is included in the year entry, if present.

Example 14.

B. Multiple Citations

Multiple citations occur when a single reference contains two or more cited articles or books by the same author or each cited work may have a different author. Multiple citations appear in two general forms: the same author and multiple authors. There are two specific formats for each of these forms. These four formats, if present, must be identified at the beginning of each category to be processed after the author.

<u>Same Author</u> - two or more citations

(a) Repetition of boldface.

Examples 16, 17.

(b) Occurrence of the form "(a)_______ (b)________".

If either of these two conditions occurs the same author is again entered into the author table and the information following the repeated punctuation is processed.

Examples 18, 19.

<u>Multiple Authors</u> - multiple citations

(a) "see also" The processing of a new reference begins if this special word is found at the beginning of a new category.

Example 20, 21.

(b) "ibid." - in the same place

"loc. cit." - in the same location

"op. cit." - in the same work

The occurrence of these special words indicates the same book or journal source as the previous citation and this information is repeated.

Example 24.

C. Descriptive Titles

If the first field of the citation (up to first punctuation mark) contains four or more words, after tests for initials and all special words likely to occur in the author field have failed, a descriptive title, having no given author, is assumed. This information is entered in the Title of Paper or Book Table. Example 25.

8. Output Format

A. Normalisation

Normalisation is necessary to standardise the output from this part of the program before entry to a SORT program. The following standardisations have been adopted. See Parts 5 and 6, Tables 1 and 2.

(a) Authors are given with last name first, followed by any initials.

(b) The title of a book or paper has leading "The" or "A" at the end, otherwise same as original source.

(c) The original form of the journal title is kept even if abbreviated. This may have to be modified as some journals are abbreviated in more than one way and this may interfere with the later matching process.

(d) The words vol. or v. will not be used in the Volume Number Table if they occur before the integer.

(e) All special words for the issue number are included in the Issue Number Table.

(f) The words p. or pp. will not be used in the Page Number Table.

(g) The publisher and city are now processed as they exist. This may need modifications as abbreviations may occur.

(h) The year is entered first into the Year Table, followed by month and day, if present.

B. Tables

(a) Unit

For each reference processed, a maximum of one data entry is made to each of the nine category tables with one exception. Each author of a multiple author reference is entered separately in the Author table. Otherwise, there is a maximum of nine data entries per reference. The next reference is then processed and an addition may or may not be made to each of the nine tables. After all references are processed, there are nine separate tables in memory of varying length. Each of these tables may be processed as a unit (e.g., alphabetical listing of all authors from Author table).

(b) Modification

To make it possible to associate all parts of a single reference (given a title, get the author(s), etc.), some identification has to be attached to each table entry. Data entry to any category table generated from a single citation would carry the same identification number. A sequential numbering or chaining increased by one for each new citation processed, must be handled during the category formation of each reference. The chain links the separated information from the same source. Note that only multiple authors have more than one data entry per table from the same reference (same identification number). After all references are processed, the tables are collected and an output tape is made containing the data with identification.

Table 3 shows how the category tables including identification will appear in memory after all references have been processed.

Table 4 shows the final output tape. The number of files on the tape equals the number of category tables to which entries have been made during processing of all citations. The first record of each file contains the table name. The first word of a record contains the identification number. In this way, each separated category can be worked with independently (e.g., all authors) or, using the entire output, parts of a single reference may be extracted.

9. **Summary**

This program, when completed, should identify and normalize all references into the categories described earlier, and set these up as tables readily available for use by a SORT program. The automatic analysis presupposes the absence of almost all pre-editing and manual determination of information content of the input data. A SORT program can later be used to extract from these prepared tables those items which obey some given search criterion. For example, authors may be listed alphabetically, multiple authors and citations may be extracted, and many other rearrangements may be effected automatically.

	Author	Title Paper, Book	Journal Title	Vol.	Issue	Page	Publisher	City	Year
	1‡	1	1	3		1	2		2
†1)	Cook, C. E.	Modification.. ....Forms	Proc. 1958 Nat. Electronics Conf.	5		108-1067	Princeton University Press		1955
	2	2	3			3			3
3)	Hastings, C. Jr.	Approximations .....Computers	J. Assoc. Compt. Mach.			181-196			1958
	3	3							
7)	Copi, I. M.	Realization... ....Events							
	3								
	Elgot, C. C.								
	3								
	Wright, J. B.								

Category Tables After Processing

TABLE 3

†Example numbers refer to reference numbers in Tables 1 and 2.

‡Identification numbers.

No.	Authors	1st Record	
1†	Cook, C. E.	record	File
2	Hastings, C. Jr.		
3	Copi, I. M.		
3	Elgot, C. C.		
3	Wright, J. B.		

No.	Title, Paper Book	1st record	
1	Modification ....Forms	record	File
2	Approximations .....Computers		
3	Realization .....Events		

No.	Journal Title
1	Proc. 1958 Electronics Conf.
3	J. Assoc. Comput. Mach.

No.	Volume
3	5

No.	Page
1	108-1067
3	181-196

No.	Publisher
2	Princeton University Press

No.	Year
2	1955
	1958

Category Tables After Chaining

TABLE 4

† Identification numbers.

APPENDIX A

Flow Chart: Identification and Standardization of Nine Reference Categories.

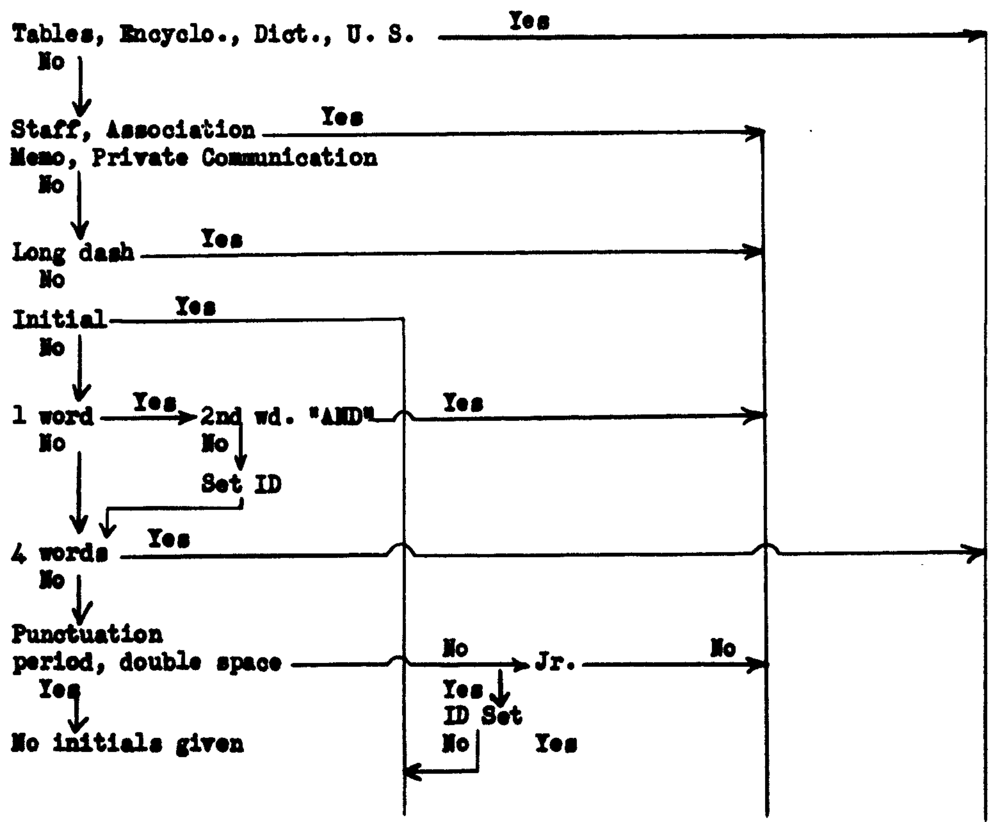
Main Program: Reads in data of next reference.

Controls sequence of 9 main subroutines.

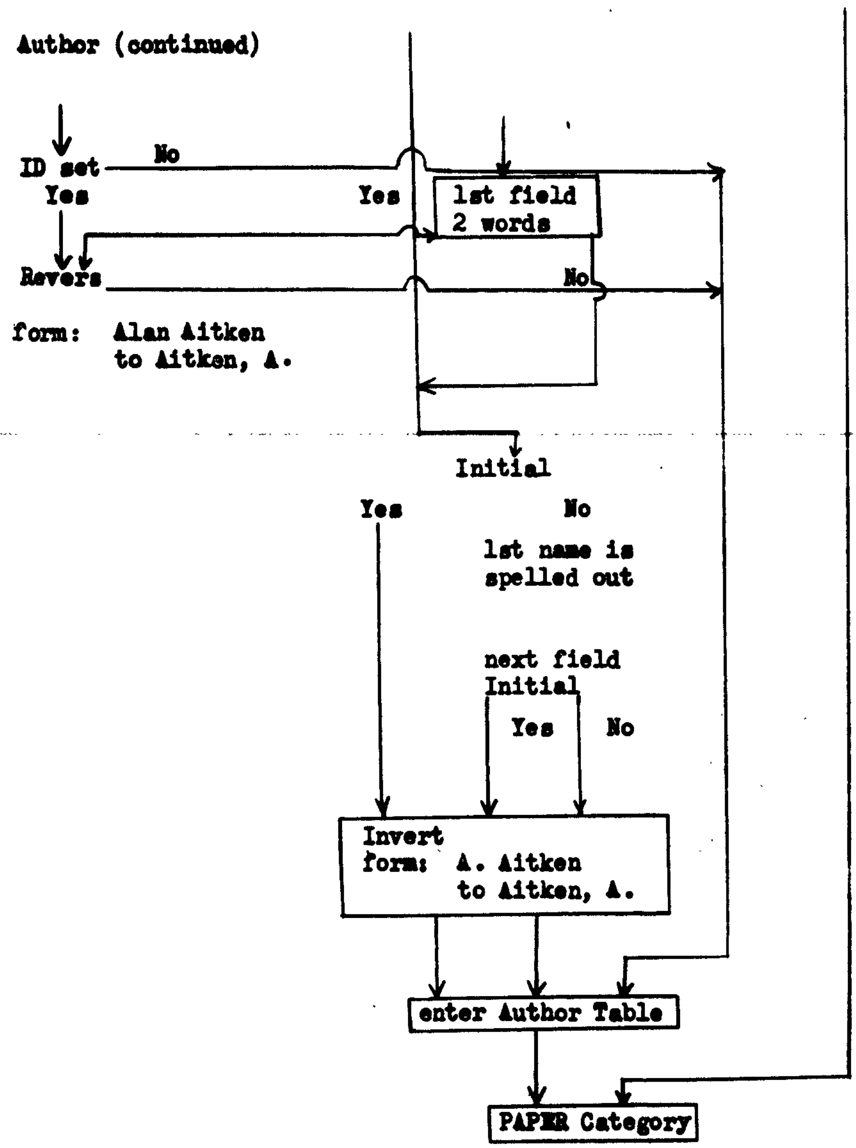
More references to process —No→ END

Author ←— Yes

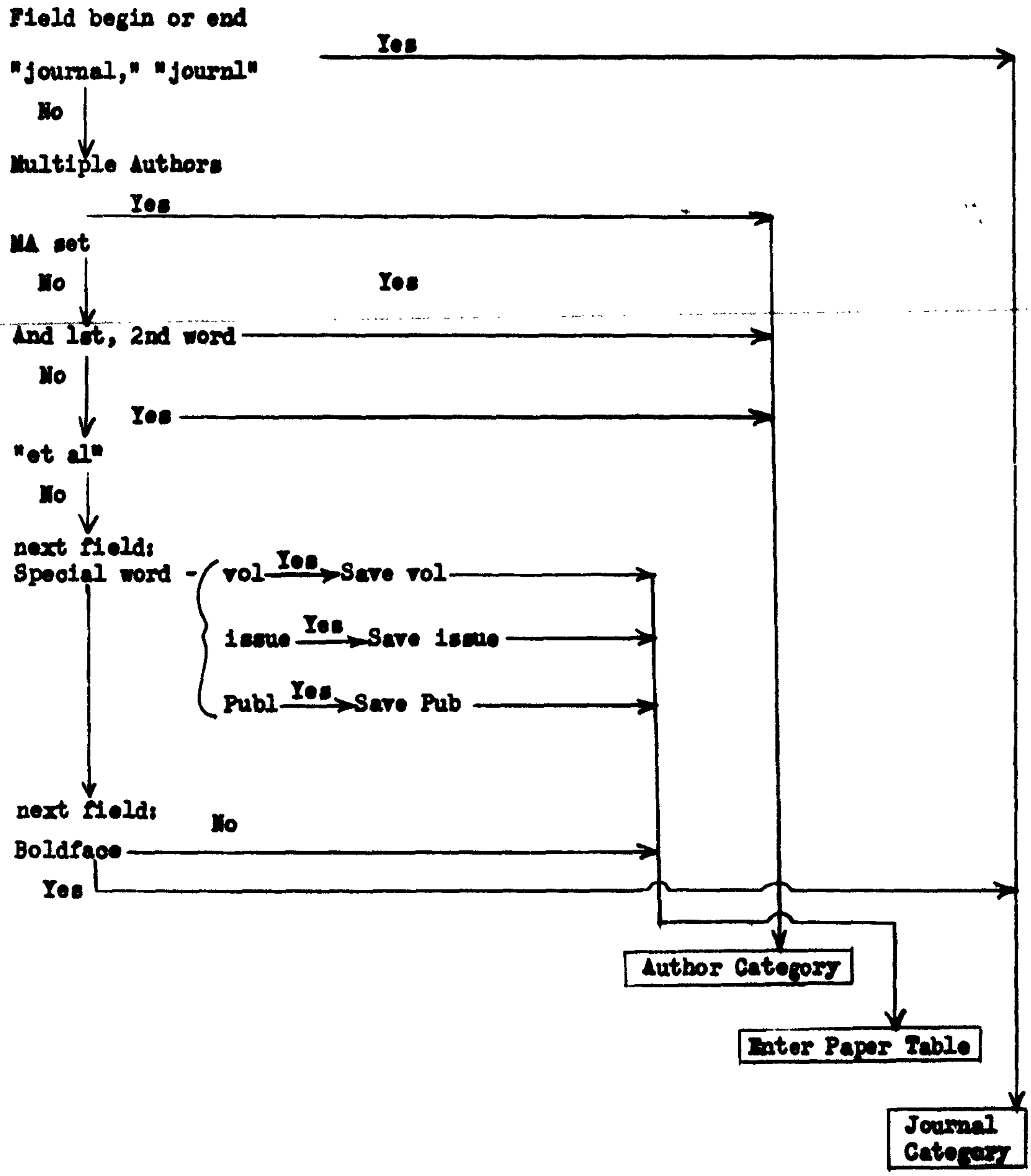
Author



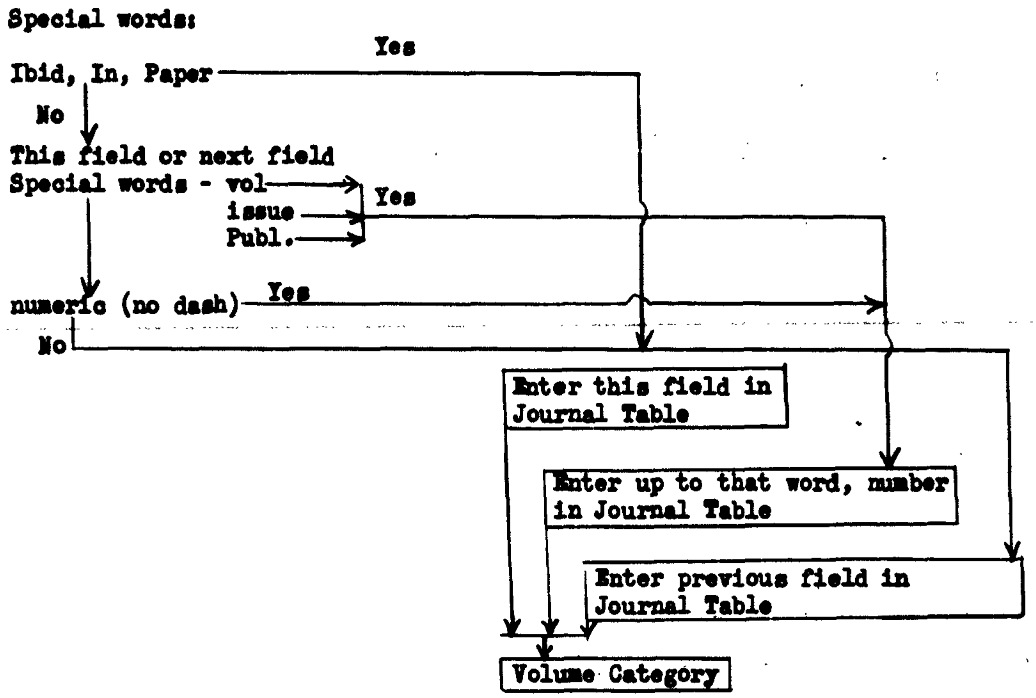
Author (continued)



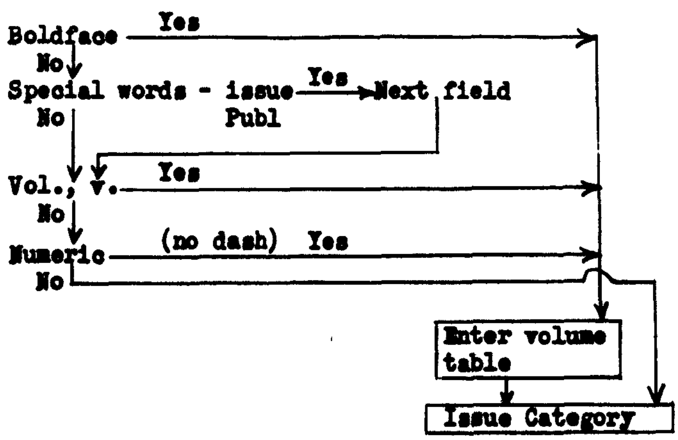
Paper


Field begin or end
"journal," "journl"
Yes
No
Multiple Authors
Yes
MA set
No
Yes
And 1st, 2nd word
No
Yes
"et al"
No
next field:
Special word -
vol
Yes
Save vol
issue
Yes
Save issue
Publ
Yes
Save Pub
next field:
No
Boldface
Yes
Author Category
Enter Paper Table
Journal
Category

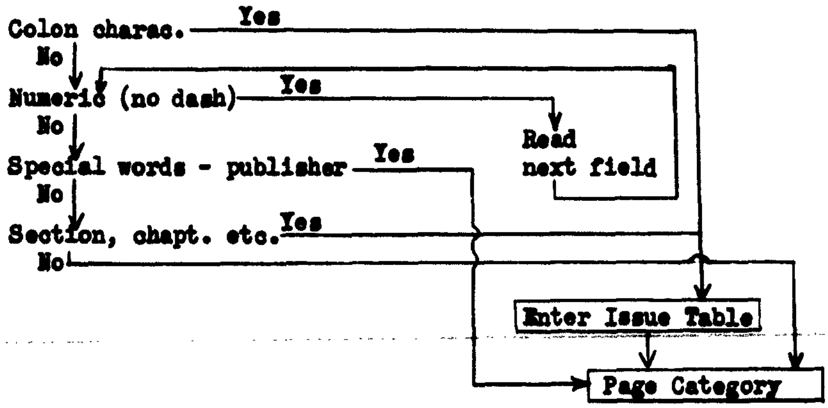
Journal



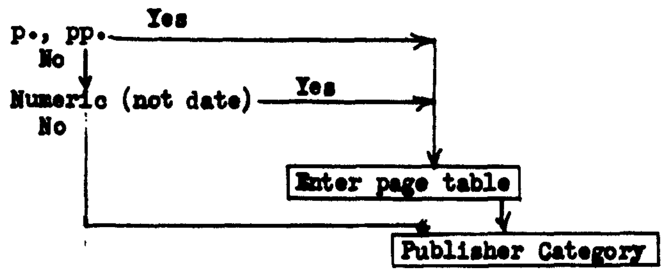
Volume



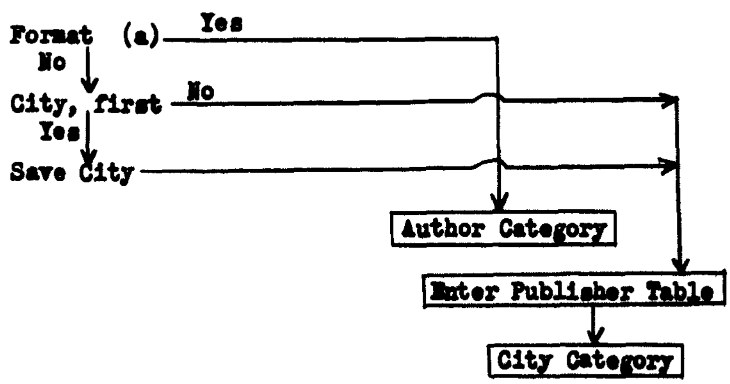
Issue



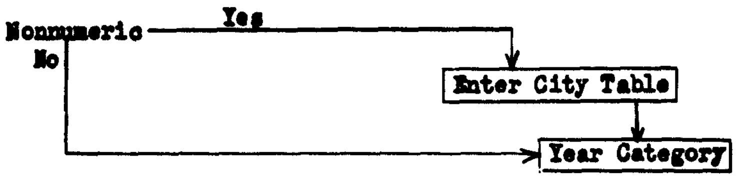
Page



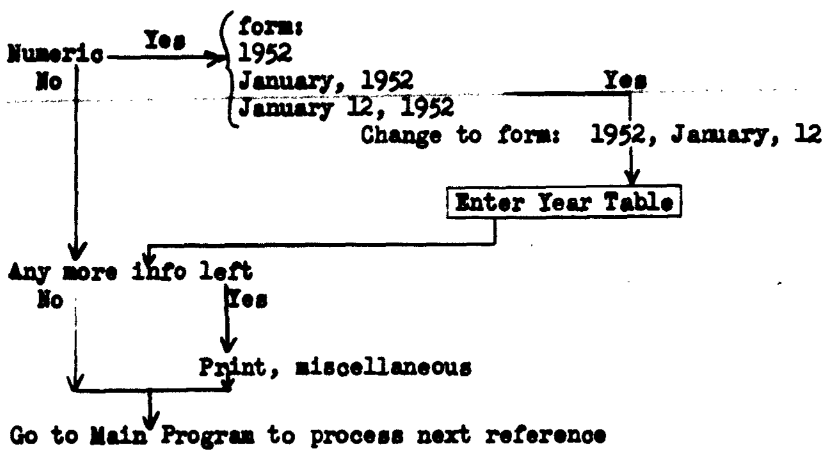
Publisher



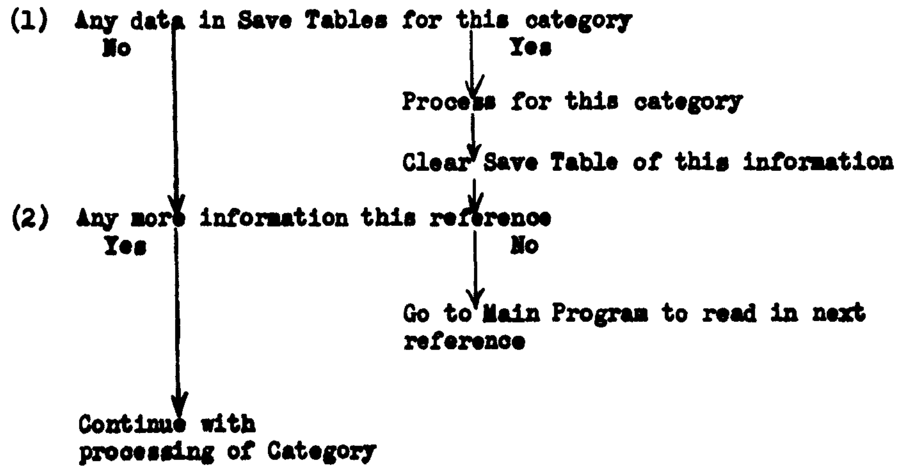
City



Year



Also at beginning of each of the nine categories two tests are always made:



III. THE USE OF TREE STRUCTURES FOR PROCESSING FILES

Edward H. Sussenguth, Jr.

ABSTRACT

In data processing problems, files are frequently used which must both be searched and altered. Binary search techniques are efficient for searching large files, but the associated file organization is not readily adapted to the file alterations. Conversely, a chained file allocation permits efficient alteration but cannot be searched efficiently. A file organized into a tree-like structure is discussed, and it is shown that such a file may both be searched and altered with times proportional to $s \log_s N$, where N is the number of file items and s is a parameter of the tree. It is also shown that optimizing the value of s leads to a search time which is only 25% slower than the binary search. The tree organization employs two data chains and may be considered to be a compromise between the organization for the binary search and the chained file. The relation of the tree organization to multidimensional indexing and to the trie structure is also discussed. An example of an automatic dictionary for language translation is used to illustrate the principles involved.

1. Introduction

In many data processing applications large files of information must be searched to extract some pertinent data and new data must be added to the file. There are many ways to perform these manipulations depending upon the structure of the file and the characteristics of the computer. When it is necessary to both search and alter the file, a sorting procedure is frequently employed in conjunction with the search technique to keep the file updated. Another attack is to avoid time-consuming sorting by allocating the file in the computer memory so that alteration is efficient; the searching of such a file is usually difficult, however. Several strategies for such problems are reviewed and analyzed below. The body of the paper, however, is concerned with a method of allocating (and implicitly sorting) the items of a file so that the file may both be searched and altered efficiently.

Before examining the details of the proposed techniques, a sample problem will be used to demonstrate the relative efficiency of the procedure when compared with other sorting and search methods. Specifically it is desired to design an efficient system to produce a listing of all distinct items from a given list of items.[†] Clearly this is a problem of the type mentioned above in which it is necessary to search and frequently alter the constructed file. An example of this problem is the tabulation of symbolic addresses and literals in assembly and compiler programs. Another illustration is the frequency counting of words in a text, a common procedure in the information retrieval field.

Let M be the total number of items in the given list, and let N be the number of distinct items in the list. If the main list is sorted and

---

[†]This problem is considered in more detail in Ref. 1.

the duplicate items identified and removed, approximately $M \log_2 M$ operations (i.e., comparisons and/or transfers) are required. (The IBM Fortran Assembly Program (FAP) uses this system to form its symbol table.[2]) Instead of sorting, the main file may be examined item by item and a file of distinct items constructed. If the constructed file is maintained in some preassigned order (to reduce the time required to test if a given item has occurred already), an approximate upper bound on the number of operations is $MN/3$. (FAP uses this system to form its table of literals.[2]) If the constructed file is not kept in order (thereby increasing the search time, but decreasing the time required to add an item to it), an approximate upper bound on the number of operations is $M\ (\log_2 N + \frac{N}{12})$. Finally, if the tree structure proposed below is used to construct the list of distinct items, the upper bound is approximately $\frac{1}{4}(M + NM^{\log_N 4})(\log_2 N)$ operations.

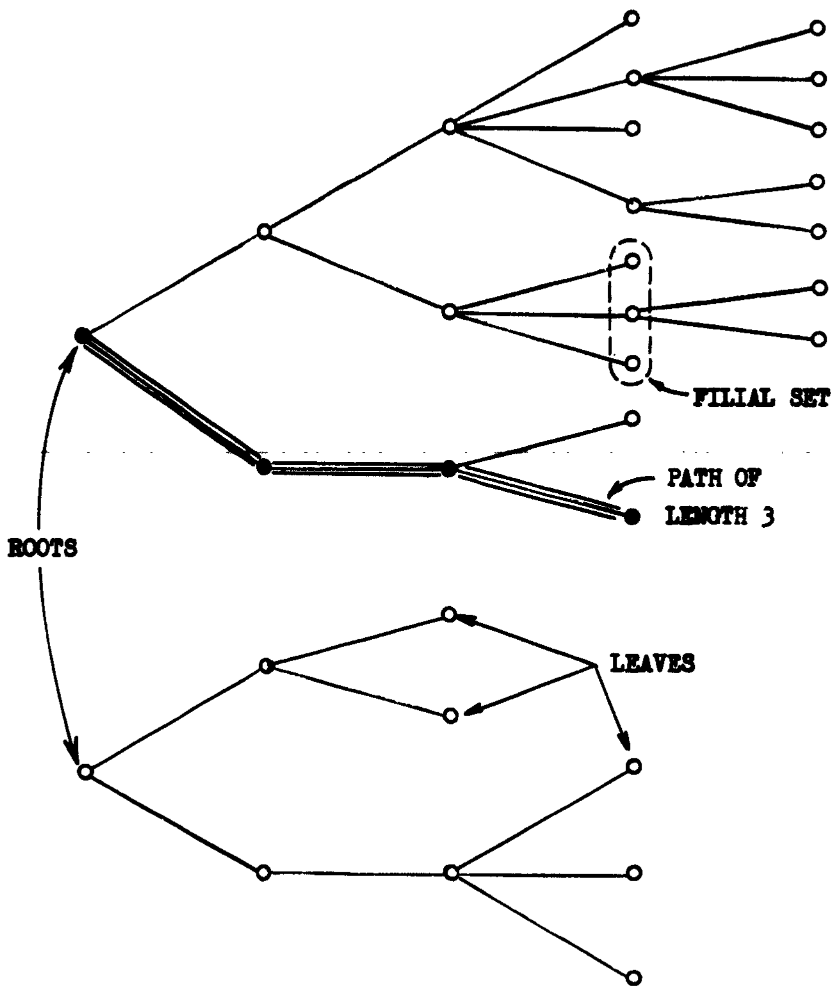
Thus, the tree procedure is significantly more efficient when there are relatively few distinct items. For example if there are 100 distinct items in a file of 1000 items, the number of operations for the four procedures are in the ratio (5:16:7:1). Hence, if the tree procedure is not too complex (so that one tree "operation" is comparable to a sorting "operation"), it merits consideration for that class of problems involving files which are both searched and altered.

2. Definitions

The underlying principle of most search techniques is to partition the main file into several small subfiles and to select one subfile for further scrutiny. As the partitioning and selection process may be illustrated and explained in terms of tree structures, it is convenient to collect all definitions of terms associated with trees. Most of these definitions have been adopted from Iverson.[3] Several basic definitions are illustrated in Fig. 1.

A graph comprises a set of nodes and a set of unilateral associations specified between pairs of nodes. If node i is associated with node j, the association is called a branch from initial node i to terminal node j. A path is a sequence of branches such that the terminal node of each branch coincides with the initial node of the succeeding branch. Node j is reachable from node i if there is a path from node i to node j. The number of branches in a path is the length of the path. A circuit is a path in which the initial node coincides with the terminal node.

A tree is a graph which contains no circuits and has at most one branch entering each node. A root of a tree is a node which has no branches entering it, and a leaf is a node which has no branches leaving it. A root is said to lie on the first level of the tree, and a node which lies at the end of a path of length j-1 from a root is on the jth level. The set of nodes which lie at the end of a path of length one from node x are said to be governed by node x and comprise the nodes of the subtree rooted at node x. A chain is a tree which has at most one branch leaving each node.



A Pictorial Representation of a Tree, Illustrating Some of the Terminology

Figure 1

3. Binary and Serial Searches

If a file of N items is stored in a random access memory with the items arranged so that their keys are in ascending order, a binary search may be used to locate an item in a time approximately proportional to $\log_2 N$. The binary search begins by testing first the item which is at the midpoint of the file. A comparison determines whether it is the desired item and, if it is not, the comparison specifies in which half of the file the desired item lies. This half is then bisected and, if necessary, the quarter of the file containing the desired item is determined. The bisection process continues until the item is located.

The binary search procedure is conveniently depicted by a tree in which each node represents a file item. The node on the first tree level corresponds to the item at the midpoint of the file; the two items on the second level correspond to the items at the one-quarter and three-quarter points of the file; etc. Except in the last one or two levels, two branches emanate from each node; which of these branches is followed is determined by the comparison of the desired item with the item associated with the node in question. Selecting one branch obviously eliminates half of the remaining candidate items.

Example: To clarify some of the principles introduced, an example of an automatic dictionary will be used. The key for such a file is an English word and the information value of the key is its equivalent in a foreign language. For purposes of specific illustration the English-Germain section of "The New Cassell's German Dictionary"[4] is used.

Figure 2 shows the memory map of a file of 38 English words arranged in alphabetical order and stored in locations 301 to 338 of a random access memory. The binary search for the word "wallah" proceeds as follows: Comparison with the item "waist" at the midpoint of the file indicates that "wallah" is in the lower half of the file because "wallah" is below "waist" in the alphabetical order. The item in the center of the lower half is "wallop;" comparison with it indicates "wallah" is in the upper half of this subfile (i.e., in the third quarter of the main file). Another comparison, this time with "walk" indicates that "wallah" is in the lower half of the third quarter. The fourth comparison then locates the desired item.

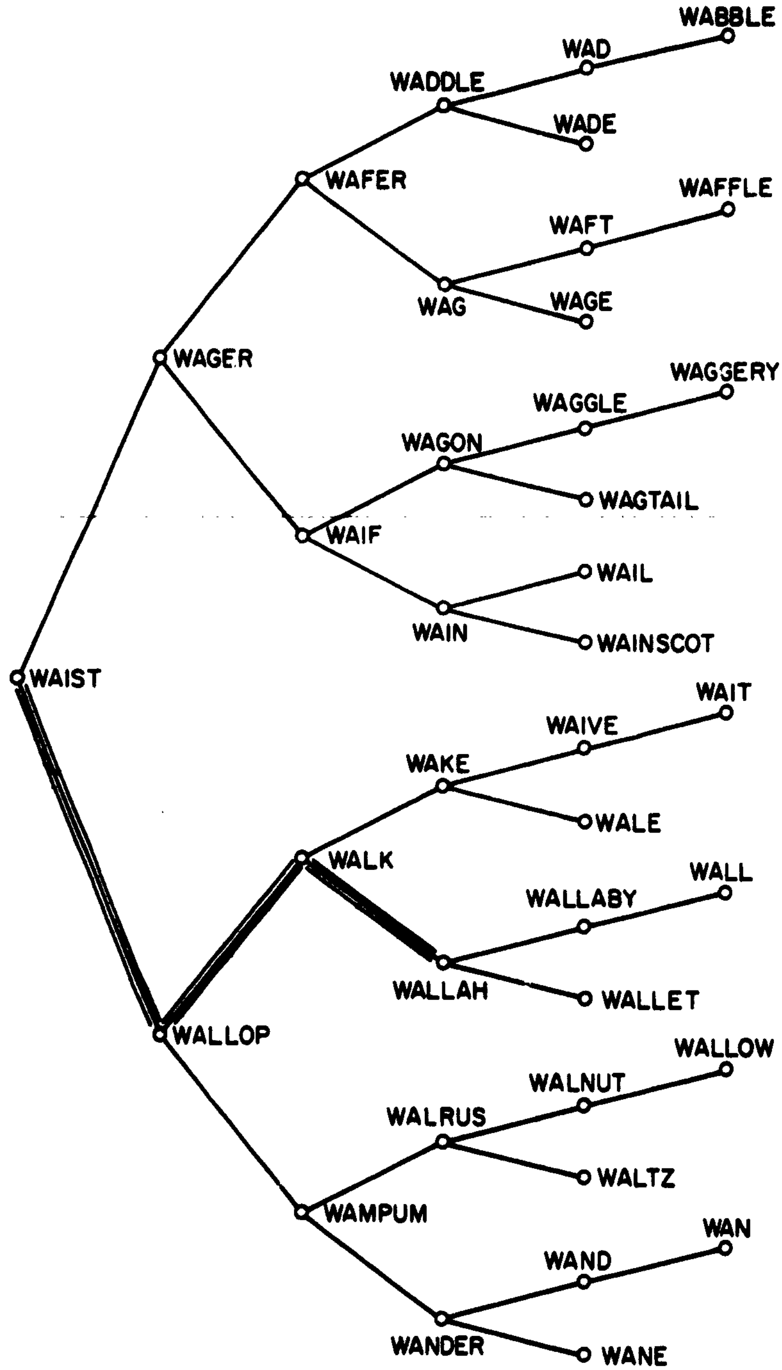
The tree representation of this binary search is shown in Fig. 3.

The binary search requires that the items be arranged in increasing order in consecutive locations of a random access memory. Although the expected search time for this arrangement ($\log_2 N$) is relatively small, the time to alter the file by adding (or deleting) items is proportional to N because many items must be moved to make room for the new item. The time to alter a file may be drastically reduced by chaining the items together instead of storing them in consecutive memory locations. With each item in the chained structure is stored the location of another item of the file. Thus, the addition of a new item is simply accomplished, because the chain may be broken at any convenient point and then relinked with the new item inserted. The time to search such a file is, however, proportional to N (whether

301 WABBLE = WACKELN
302 WAD = BUNDEL
303 WADDLE = WATSCHELN
304 WADE = WATEN
305 WAFER = WAFFEL
306 WAFFLE = WAFFEL
307 WAFT = FORTWEHEN
308 WAG = WEDELN
309 WAGE = LOHN
310 WAGER = WETTE
311 WAGGERY = SPASS
312 WAGGLE = WACKELN
313 WAGON = LASTWAGEN
314 WAGTAIL = STELZE
315 WAIF = VERWAHRLOSTES KIND
316 WAIL = JAMMERN
317 WAIN = WAGEN
318 WAINSCOT = TAFELUNG
319 WAIST = TAILLE
320 WAIT = WARTEN
321 WAIVE = AUFGEBEN
322 WAKE = WACHEN
323 WALE = STRIEME
324 WALK = GEHEN
325 WALL = MAUER
326 WALLABY = KLEINES KANGURUH
327 WALLAH = BURSCHE
328 WALLET = BRIEFTASCHE
329 WALLOP = WUCHT
330 WALLOW = SICH WALZEN
331 WALNUT = WALNUSS
332 WALRUS = WALROSS
333 WALTZ = WALZER
334 WAMPUM = MUSCHELGELD
335 WAN = BLEICH
336 WAND = RUTE
337 WANDER = WANDERN
338 WANE = ABNEHMEN

A Listing of 38 Words with the Route of the Binary Search for the Word "WALLAH"

Figure 2



The Tree Which is Implied by a Binary Search of the File of Fig. 2

Figure 3

or not the order is maintained), because only one other item is accessible from any given item. Hence the search must proceed serially, item by item. The tree representation of this serial search reduces to the trivial case of a chain. Also trivial is the new subfile partitioned at each step; it is merely the previous file less one item.

Example: Figure 4 is a representation of that portion of random access memory containing the file of the 38 words of Fig. 2 in a chained allocation. The words are not arranged in any order with respect to the memory locations, but the alphabetic order is maintained by the chain. To retrieve the word "wallop," first the word "wabble" at the start of the chain (its location 312 is prespecified) is tested. As it is not the desired item, the next word of the chain, "wad," is tested; the location of "wad," 323, is given as part of the data of "wabble." The chaining link of the word "wad" indicates location 313 is to be tested next. Thus the items along the chain are tested until the desired item is found. Figure 5 shows the tree representation of this serial search.

To add the word "waiter" to the chained allocation, the data for "waiter" is stored in an available location (339), the chain broken after "wait," and the new item inserted. Thus after inserting "waiter," location 317 contains

WAIT = WARTEN 339

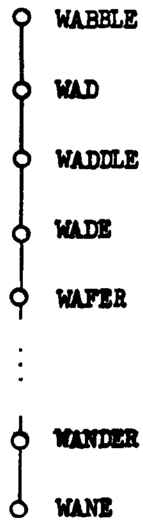
and location 339 contains

WAITER = KELLNER 323.

301	WADE	316
302	WALRUS	319
303	WAG	332
304	WAN	329
305	WAINSCOT	318
306	WAGTAIL	314
307	WAMPUM	304
308	WALLOP	333
309	WALLET	308
310	WAGGLE	324
311	WANE	*
312	WABBLE	338
313	WADDLE	301
314	WAIF	331
315	WALNUT	302
316	WAFER	320
317	WAIT	323
318	WAIST	317
319	WALTZ	307
320	WAFFLE	330
321	WALL	337
322	WAGGERY	310
323	WAIVE	326
324	WAGON	306
325	WANDER	311
326	WAKE	336
327	WALK	321
328	WALLAH	309
329	WAND	325
330	WAFT	303
331	WAIL	334
332	WAGE	335
333	WALLOW	315
334	WAIN	305
335	WAGER	322
336	WALE	327
337	WALLABY	328
338	WAD	313

A Memory Map of the Chained Allocation of the Words of Fig. 2

Figure 4



The Tree Implied by a Serial Search

Figure 5

To add "waiter" to the ordered listing (Fig. 2), however, requires moving all words from "waive" to "wane" down one location in memory before inserting "waiter" in its proper location at 321.

Summarizing, it is seen that the ordered arrangement with a binary search is efficient for a file which is frequently searched and infrequently altered, and the chained arrangement efficient for a file which is frequently altered but infrequently searched. If it is necessary both to search and to alter the file, neither arrangement is attractive and another may be preferable.

The tree allocation, described in the following sections, is a compromise arrangement which utilizes the effective partitioning of the file found in the binary search and which chains items together for simplicity of organization and alteration. It will be shown that the tree allocation may be both searched and altered with time proportional to $s \log_s N$, where s is a parameter associated with the tree structure. It is useful both when the file is completely stored in random access memory and when the bulk of it is in a disc or drum memory.

4. The Tree Allocation

In a tree allocation the partitioning of the file is accomplished by breaking the key into several disjoint parts. Each part or element of the key is made to correspond to a level of the tree. The first tree level lists all possible values of the first element of the key. With each of these elements is associated a list of those second elements which may be used in combination with that first element. A complete list of all possible second elements is not necessary for some of them will never be used.

<u>Example</u>: A natural way to break an English word into several parts is to partition it into its component letters. Then the first tree level will have 26 nodes, one for each letter of the alphabet. With each letter is associated a list of letters which may be used with it as the start of an English word. Thus with the letter "w" on the first level is associated a list of the following letters on the second level: a, e, h, i, o, r, and y. The letter "k," for example, is not included in this list as no English word starts with the pair "wk."

The two-level tree may partition the file into enough subfiles so that any one of them may be conveniently searched serially. Then with each second-level tree node, i.e., with each pair of elements, is associated the block of items governed by the partition implied by the pair. However, if the number of elements in each subfile is still so large that an efficient search is not possible, the tree may be extended to include more levels. In this case, a list of those key elements which may be used with the pairs corresponding to the second-level nodes is associated with each second-level tree node. Clearly the tree may be extended in this manner through as many levels as desired, and one part of the tree may contain more levels than some other part. By varying the number of levels in different parts of the tree it is possible to make the number of items in each subfile nearly uniform, if this is desired. Indeed it is possible to extend the tree levels so that each subfile consists of only one item. The remainder of this section describes a procedure by which the number of levels and the subfile sizes may be chosen so as to minimize the expected search time.

__Example:__ There are about 14,000 English words listed in the English-German section of Cassell's dictionary. The partitioning imposed by a one-level tree is shown in Fig. 6, where it is seen that "w" governs 355 items. Partitioning on the first two letters breaks the "w" portion of the file into the subfiles of the following sizes (see Fig. 7):

wa	75	wh	70	wo	45
we	55	wi	78	wr	31
				wy	2.

A	755	N	270
B	775	O	355
C	1335	P	1170
D	815	Q	100
E	515	R	840
F	680	S	1695
G	395	T	785
H	480	U	615
I	535	V	200
J	100	W	355
K	80	X	15
L	540	Y	50
M	675	Z	30

Distribution of Initial Letters of a Sample of 14,000 English Words

Figure 6

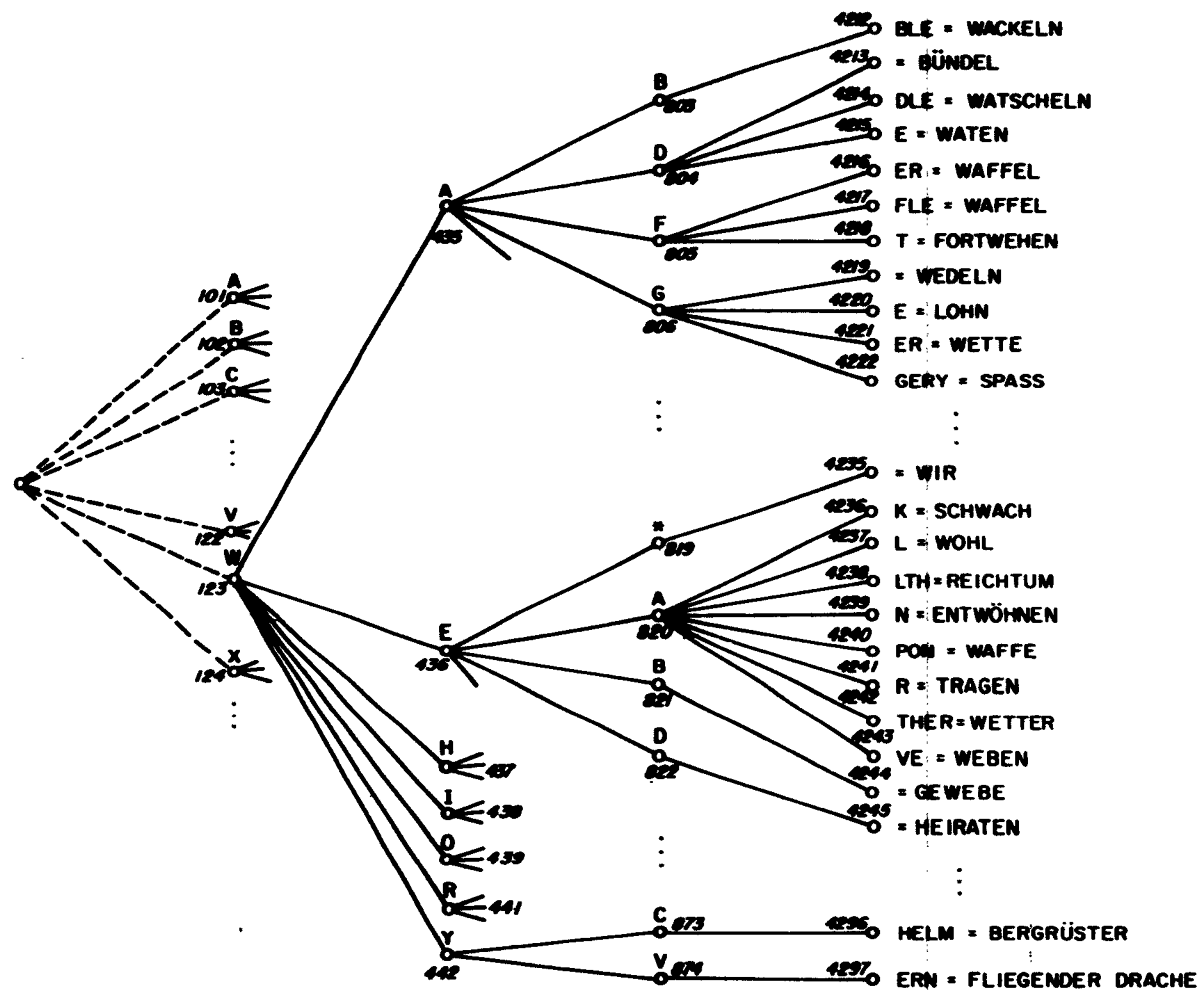
The subfile associated with "wy" is certainly small enough to be conveniently searched on an item-by-item basis. However, the other subfiles associated with "w" would require an item-by-item search which is considerably longer and perhaps should be partitioned on the third letter. Figure 8 shows part of a four-level tree for this file.

The search for a given item in a tree allocation is conducted by scanning the set of nodes on the first level until the element which matches the first element of the key is located. Then one proceeds to the set of elements associated with that node on the next level; that is, to the filial set of that node. This set is scanned until the second key element is matched. Its filial set is located and scanned similarly.

Third letter \ Second letter	A	E	H	I	O	R	Y
*		1					
A		13	4		1	7	
B	1	1			1		
C				4			1
D	3	4		4			
E		7	15	1	1	8	
F	3	1		1			
G	9			3	1		
H							
I	7	3	37			9	
J							
K	2						
L	11	12		13	5		
M	1			1	4		
N	7	4		18	5		
O			13		7	4	
P	1	1		1	1		
Q							
R	16	3		1	13		
S	5	1		7			
T	4	2		19	1		
U					3	1	
V	1	1		3	1		1
W					1		
X	2						
Y	1	1	1			2	
Z				2			
Totals	75	55	70	78	45	31	2
Number of entries	16	15	5	14	14	6	2

Distribution of Second and Third Letters When Initial letter is W

Figure 7



A Four Level Tree

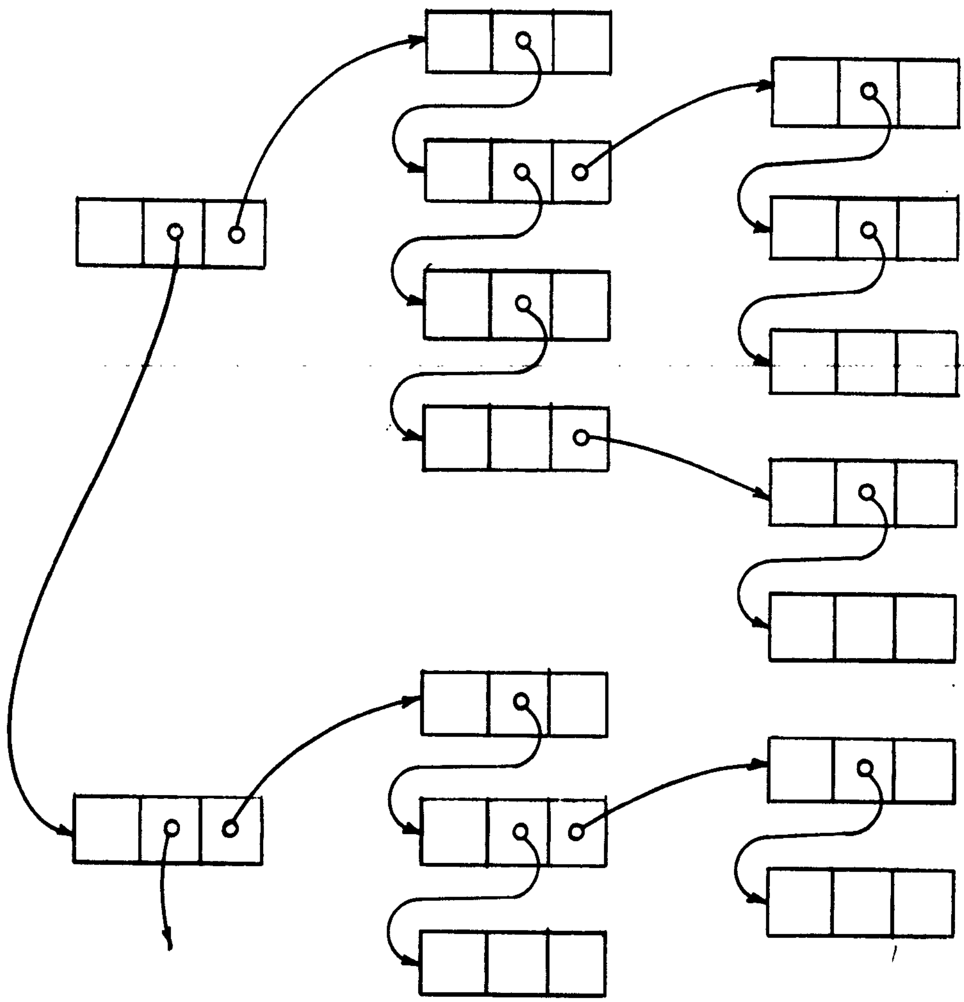
Figure 8

This process continues, building a path through the tree to the final block of key elements. This block is just the filial set of the last node of the path; it differs from the other filial sets in two ways. First, it has associated with each of its nodes, not one element of the key, but all of the elements of the key not already associated with the nodes of the path leading to it. Second, each of the nodes of the final filial set does not have a filial set of its own, but rather indicates the information value of the key which the node represents.

There are several ways in which a node may be joined to its filial set and the nodes within a filial set may be kept together. A very convenient technique is to chain each node to its filial set and to chain the nodes within each filial set together. This arrangement is called a doubly-chained tree.† Using this method, each tree node is represented by one computer word. The computer word is divided into three fields: the first indicates the key element value of the node, the second contains the address of another node in the filial set of which the given node is a member, and the third contains the address (of the first node) of its filial set.# (See Fig. 9.)

†Iverson[3] describes a similar arrangement called a filial-heir chain representation. Johnson[5] also discusses this arrangement. The tree is also closely related to list structures.[9]

#A singly-chained tree is also possible. The nodes of a filial set are stored in consecutive memory locations, instead of being chained together. Then each computer word contains a value field and only one address field.



Schematic Representation of a Doubly-Chained Tree

Figure 9

<u>Example</u>: Figure 10 shows the actual contents of memory for the doubly-chained allocation of the tree in Fig. 8.

The filial set is scanned by following the chain of addresses in the second field of the computer word and comparing the given key element with the key element values of the nodes of the filial set. When a match is found, the next tree level is reached by branching to the address given in the third field. Thus if there are s nodes in the filial set, the expected number of chaining links required to find a match is $\frac{1}{2}(s + 1)$: one link to reach the filial set and $\frac{1}{2}(s - 1)$ to search it.[*]

It is easy to add an item to the doubly-chained tree allocation. The tree is entered assuming the item is in the file, and the path for the key found. At some point a key element will not be found in an existing filial set. This new key element is then added to that filial set by breaking and relinking the filial set chain. The double chaining feature permits the use of any available memory locations for the new tree nodes. Moreover, this feature allows an item to be added to the file in roughly the same time that is required to locate an item.[**]

In many applications the file is so large that it will not fit into fast random access memory but is stored on slower media such as discs

[*] It is assumed all nodes in the filial set have the same probability of being selected. If the probabilities differ, the nodes should be tested in order of decreasing probability for greatest efficiency. An easy way to do this is to arrange the nodes in the order of the number of items they govern.

[**] In a singly-chained tree the new tree nodes must be added at a location adjoining the locations reserved for their filial set. To do this may entail relocating the entire filial set.

101	A	102	128
102	B	103	224
103	C	104	236

122	V	123	428
123	W	124	435
124	X	125	443

435	A	436	803
436	E	437	819
437	H	438	834
438	I	439	839
439	O	440	853
440	R	441	867
441	Y	-	873

803	B	804	4212
804	D	805	4213
805	F	806	4216
806	G	807	4219

819	*	820	4235
820	A	821	4236
821	B	822	4244
822	D	823	4245

873	C	874	4296
874	V	-	4297

4212	BLE = WACKELN
4213	= BÜNDEL
4214	DLE = WATSCHELN
4215	E = WATEN
4216	ER = WAFFEL
4217	FLE = WAFFEL
4218	T = FORTWEHEN
4219	= WEDELN
4220	E = LOHN
4221	ER = WETTE
4222	GERY = SPASS

4235	= WIR
4236	K = SCHWACH
4237	L = WOHL
4238	LTH = REICHTUM
4239	N = ENTWÖHNEN
4240	PON = WAFFE
4241	R = TRAGEN
4242	THER = WETTER
4243	VE = WEBEN
4244	= GEWEBE
4245	= HEIRATEN

4296	HELM = BERGRÜSTER
4297	ERN = FLIEGENDER DRACHE

The Storage Map of the Tree of Fig. 8

Figure 10

or drums. A file stored on a disc or drum is physically subdivided by the tracks of the disc or drum into subfiles, one subfile per track. In this case the tree structure may be thought of as a transformation from the key to the track address of its associated data. The subfile corresponding to this track may then be scanned for the desired item on an item-by-item basis or transferred to the random access memory for scanning by the binary search.

5. Minimization of the Expected Search Time

Assuming the search time is proportional to the number of chaining links traversed, the expected search time may be calculated if all of the filial set sizes are known. However, it is inconvenient to use the set of all filial set sizes in macroscopic calculation. For computational purposes an average filial set size for each tree level is defined as:

$$s_i = \frac{\text{number of nodes on level } i}{\text{number of filial sets on level } i}. \qquad (1)$$

Since the average time to search a filial set on the ith level is $\frac{1}{2}(s_i + 1)$, the expected time to search an h-level tree is

$$\frac{1}{2}\sum_{i=1}^{h} (s_i + 1). \qquad (2)$$

Thus the expected search time, $\bar{t}$, for a file of N items allocated as an h-level tree with blocks of average size B is

$$\bar{t} = \frac{1}{2} \sum_{i=1}^{h} (s_i + 1) + t(B), \tag{3}$$

where t(B) is the expected time to search one block.

Equation (3) gives the expected search time in terms of the parameters s_i, h, and B. These parameters are related by the expression

$$B \prod_{i=1}^{h} s_i = N, \tag{4}$$

because $\prod_{i=1}^{h} s_i$ represents the number of nodes on the hth level,† which is just equal to the number of blocks. Frequently when designing an allocation and search system for a particular file the user is free to vary one or more of s_i, h, or B under the constraint of Equation (4) to achieve an efficient system. Several such situations are discussed below.

CASE A: In many data processing problems the file fits within the random access memory, and the data processing requirements are such that the key elements may be manipulated at will as long as the proper response is received from a given query. The keys may then be considered to consist of a single string of binary digits rather than several disjoint elements each of which consists of several binary digits. For example, the

---

†This assumes there are no leaves on levels 1,2,...,h-1.

key "CAT" is considered as the binary string "010011010001110011" rather than the set of distinct elements "C," "A," and "T," or equivalently, the distinct elements "010011," "010001," and "110011." With keys of this format, the binary digits may be grouped to give the most efficient search system; that is the s_i, h, and B may be selected without constraint (other than (4)) to minimise $\bar{t}$. It is shown in the appendix that the minimum $\bar{t}$ is achieved when:

(1) all paths from a root to a leaf have the same length, h;

(2) all filial sets have the same number of members, s, that is $s_1 = s_2 = \cdots = s_h = s$;

(3) the number of elements in a block, B, is the same as the common filial set size, that is $B = s$; and

(4) the common filial set size s is $N^{1/h+1}$.

The results (1) through (4) fix B as equal to s but related s, h, and N only by the relation $s^{h+1} = N$, so that either s or h may be selected arbitrarily. Another analysis, also in the appendix, shows that the s, which satisfies this constraint and also minimises $\bar{t}$, is 3.6 nodes per filial set.

CASE B: Another class of data processing problems has characteristics similar to those of Case A, except that the file is too large to fit within the random access memory and is stored instead on a drum or disc memory. Here the block size B is normally determined by the number of items, T, which can be accommodated by one track, and not by considerations which minimise the over-all search time. For these cases the tree acts as a transformation from the set of keys to the set of track addresses, and $\bar{t}$ may be minimized by varying the s_i and h. It is shown in the Appendix that when B is fixed at T the minimum $\bar{t}$ is achieved when

(1) all paths from a root to a leaf have length h;

(2) all filial sets have the same number of members, s; and

(3) the common filial set size s is $(\frac{N}{T})^{1/h}$.

As in Case A, absolute values for s and h are not fixed by these results, and it can be shown that the optimum s is also 3.6 nodes per set.

CASE C: For some applications it may be inconvenient or unnatural to consider the keys as single strings of binary digits; rather the keys must be considered to consist of several distinct elements. If the number of elements per key is a constant, h, for all items of the file, or if it is desirable to form a tree using h levels, the analyses of Cases A and B apply, assuming h to be fixed rather than variable. Thus, the minimum search time is achieved when the filial set sizes are all equal to the optimum value of either $N^{1/h+1}$ or $(\frac{N}{T})^{1/h}$, according as the file fits within random access memory or the file is stored on a disc memory.

The optimum tree allocation for these three cases requires that all paths within a tree be of the same length. Frequently, however, it is profitable to vary the path lengths within the tree, stopping the branching in each path at some optimum path length. For the case when the main storage is a disc, the proper path lengths are easily determined because each leaf must govern T item. Thus, if a particular node governs T or fewer items, the branching along that particular path may be stopped. If, however, the node governs more than T items, the branching must be continued for at least one more level.

If the file fits within the random access memory, the optimum path length determination is based on the number of items, g, which a particular node governs and the number of nodes, s, in its filial set. A detailed analysis of this optimization is presented in the appendix; the main result is that in most cases the optimum length of a path is achieved when the branching is discontinued at the first node which governs fewer than six items.

In Cases A and B in which it was possible to manipulate the keys to determine a minimum expected search time, it was found that the minimum was achieved when all filial sets had the same number of members, s, and the optimum s was 3.6. Substituting into Equation (3) it can be shown that (see Appendix)

$$\bar{t} = \frac{s+1}{2} \log_s M \tag{5}$$

where M is either N or $\frac{N}{T}$ according as the file is stored in the random access memory or on a disc memory. Letting s take on its optimum value of 3.6, one finds

$$\bar{t} = 2.3 \log_{3.6} M = 1.24 \log_2 M. \tag{6}$$

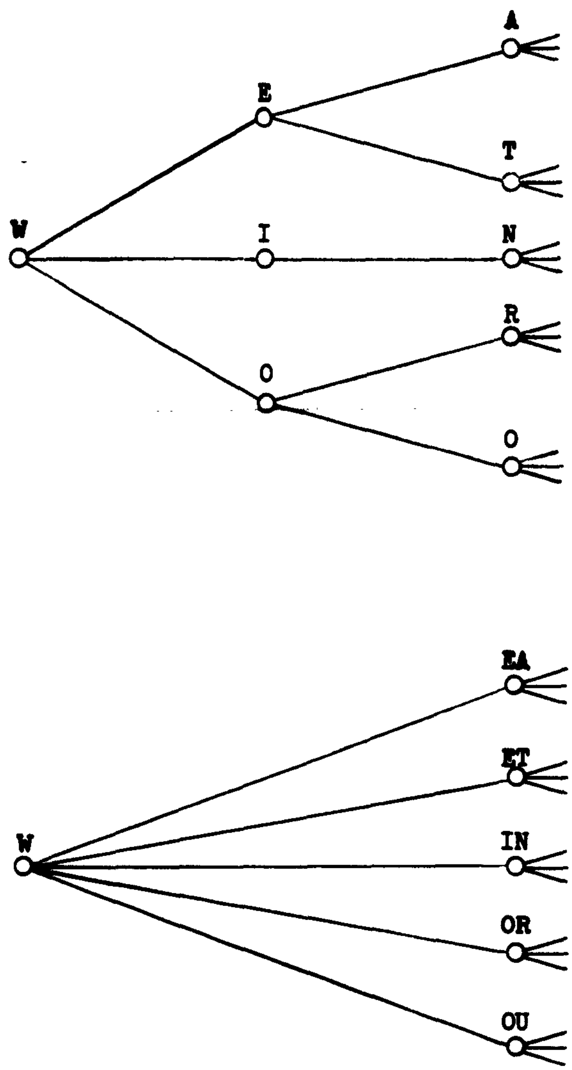
That is, the expected search time in the optimum case is only 24% slower than the binary search time. Moreover, as was indicated in the previous section, the time to add or delete an item to the file is approximately the same as the search time, a considerable improvement over the file allocated for binary searching.

It is not always necessary to manipulate the keys so that the constructed elements are completely different from the natural elements, although in general any natural partitioning of the keys (e.g., partitioning English words into their component letters) will not lead to filial sets of the optimum size. If the sizes, s_i, are smaller than the optimum, two or more tree levels may be combined to a single level with more nodes per filial set by considering two or more key elements to be a single element. Conversely, if an s_i is much larger than the optimum, one tree level may be split into several levels by factoring the corresponding key element. The simplest way to accomplish the factorization is to take the binary representation of the element in several pieces of a few bits each, e.g., consider a six-bit element as two pieces of three bits each or three pieces of two bits. Factoring in this way should make the filial sets relatively constant in size, not only from level to level, but also within the same level.

<u>Example</u>: Figure 11 shows a tree in which two levels have been combined into a single level by considering the second and third letters to be a single key element.

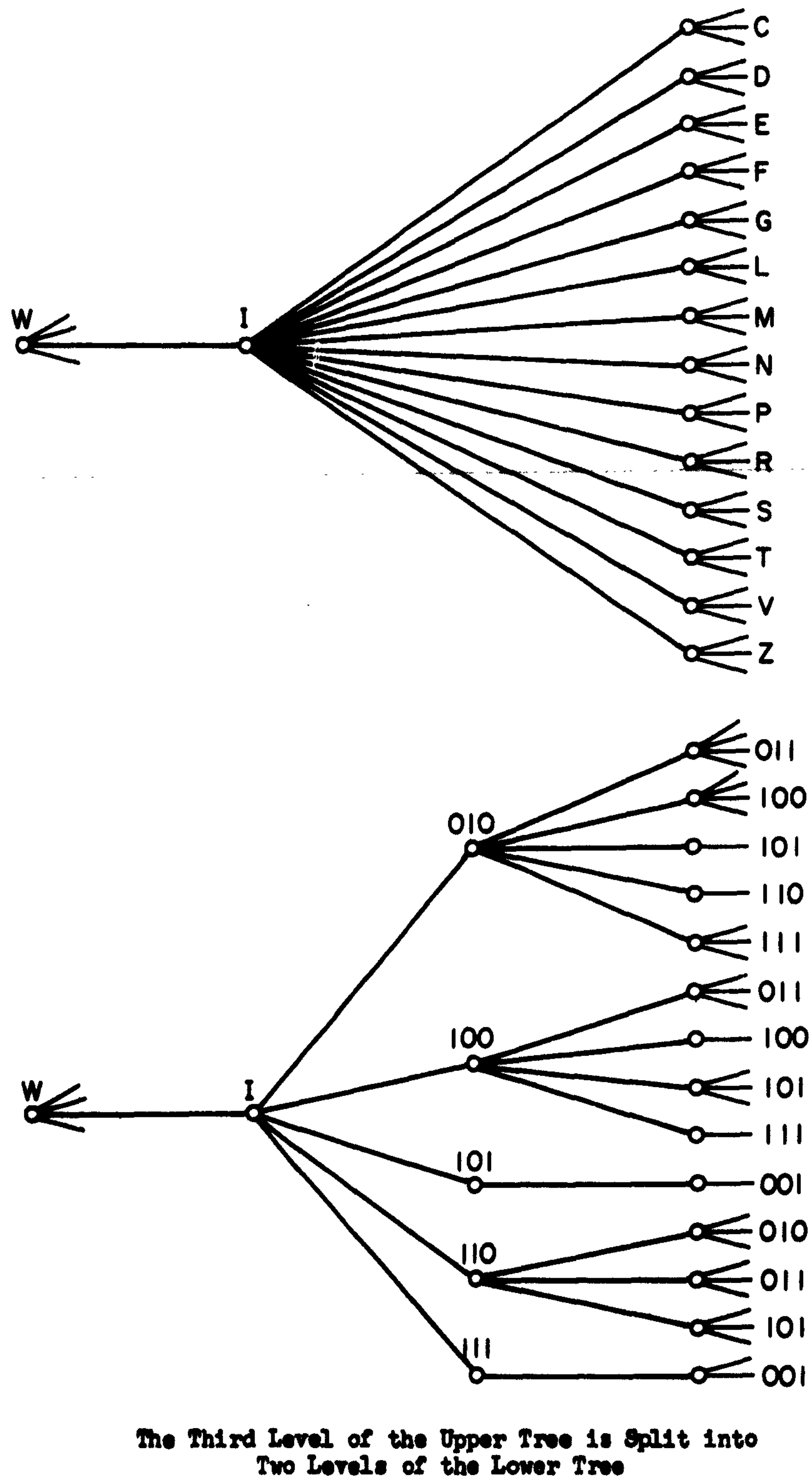
Figure 12 shows how one level may be split into two levels by factoring the binary representation of the key elements.

The efficient search time of the tree structure noted above is achieved by using a relatively small filial set size. Small filial sets



The Second and Third Levels of the Upper Tree are Combined into a Single Level of the Lower Tree

Figure 11



The Third Level of the Upper Tree is Split into Two Levels of the Lower Tree

Figure 12

in turn imply relatively larger amounts of random access storage locations to accommodate the tree. Specifically, the total number of nodes in a tree of h levels, assuming there are leaves only on the hth level, is

$$W = \sum_{j=1}^{h} \prod_{i=1}^{j} s_i. \qquad (7)$$

If it is assumed that s_i are all equal, $s^h = N$, and (7) becomes

$$W = \sum_{j=1}^{h} s^j = s\frac{s^h - 1}{s - 1} \approx \frac{s}{s - 1}N. \qquad (8)$$

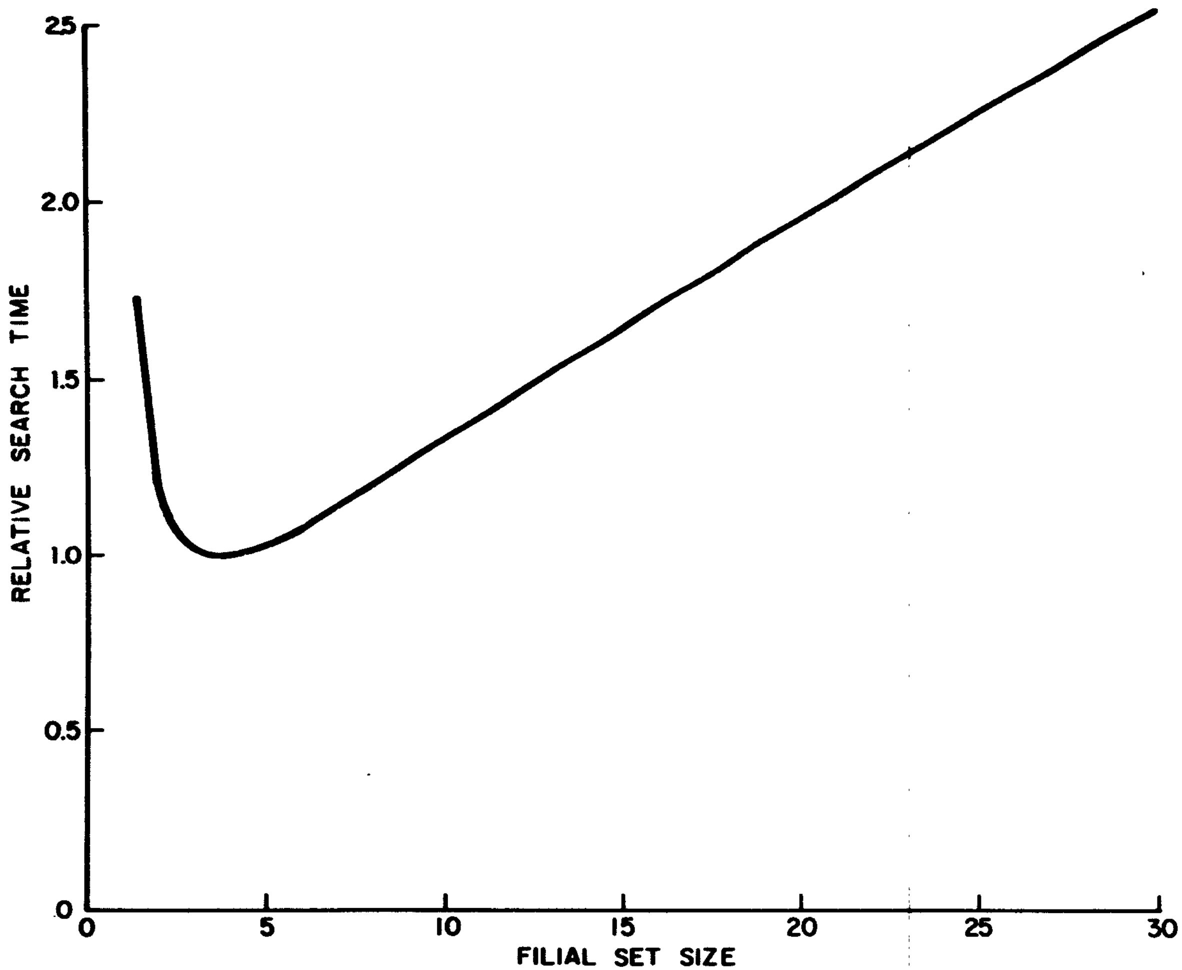
Equation (8) shows that as s increases, the number of storage locations required decreases.

Thus an efficient search time is achieved with relatively small s, but fewer storage locations are needed with relatively large s. To achieve a balance between these conflicting criteria, the measure of efficiency may be taken to be the product of search time and storage capacity. This measure is directly related to the cost of the system for it reflects both the amount of equipment and the length of time the equipment is in use. Using (5) and (8) one finds that the cost C is

$$C = \frac{s}{2}\left(\frac{s + 1}{s - 1}\right)(s^h - 1). \qquad (9)$$

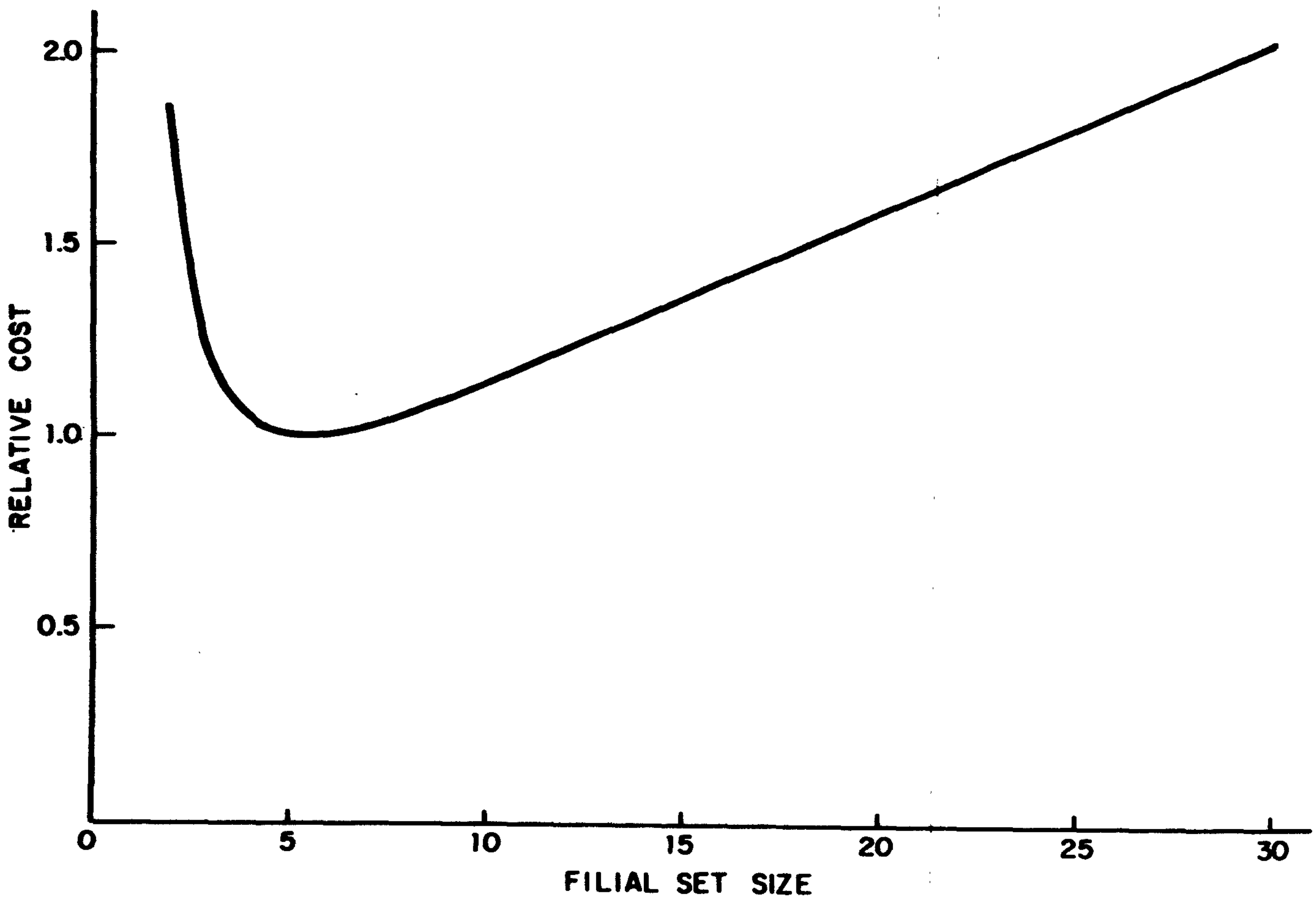
The cost C achieves its minimum value when s = 5.3.

The curves plotted in Figs. 13, 14, and 15 show, respectively, the expected search time as a function of s (Case A or B), the cost as a function of s (Case A or B), and the search time as a function of s



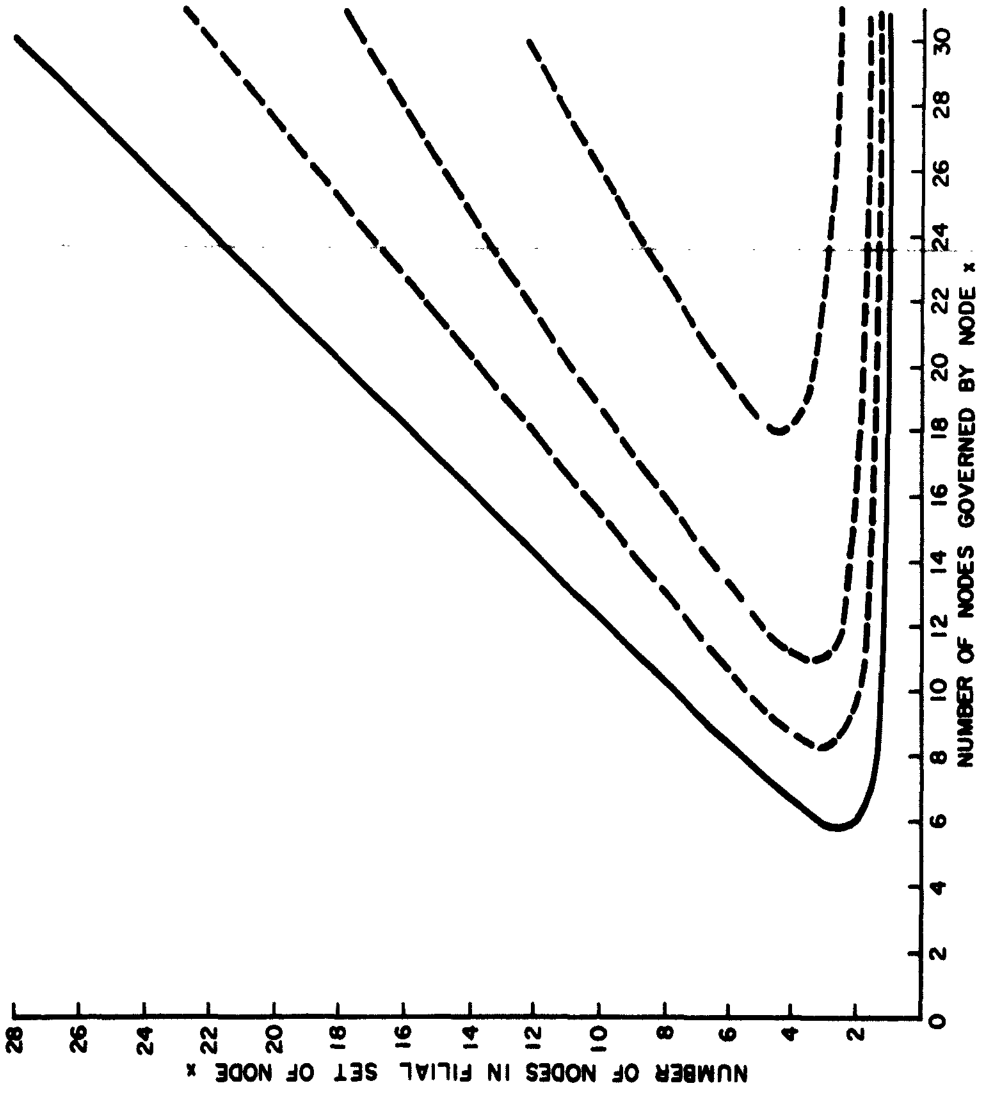
Relative Search Time as a Function of Average Filial Set Size

Figure 13

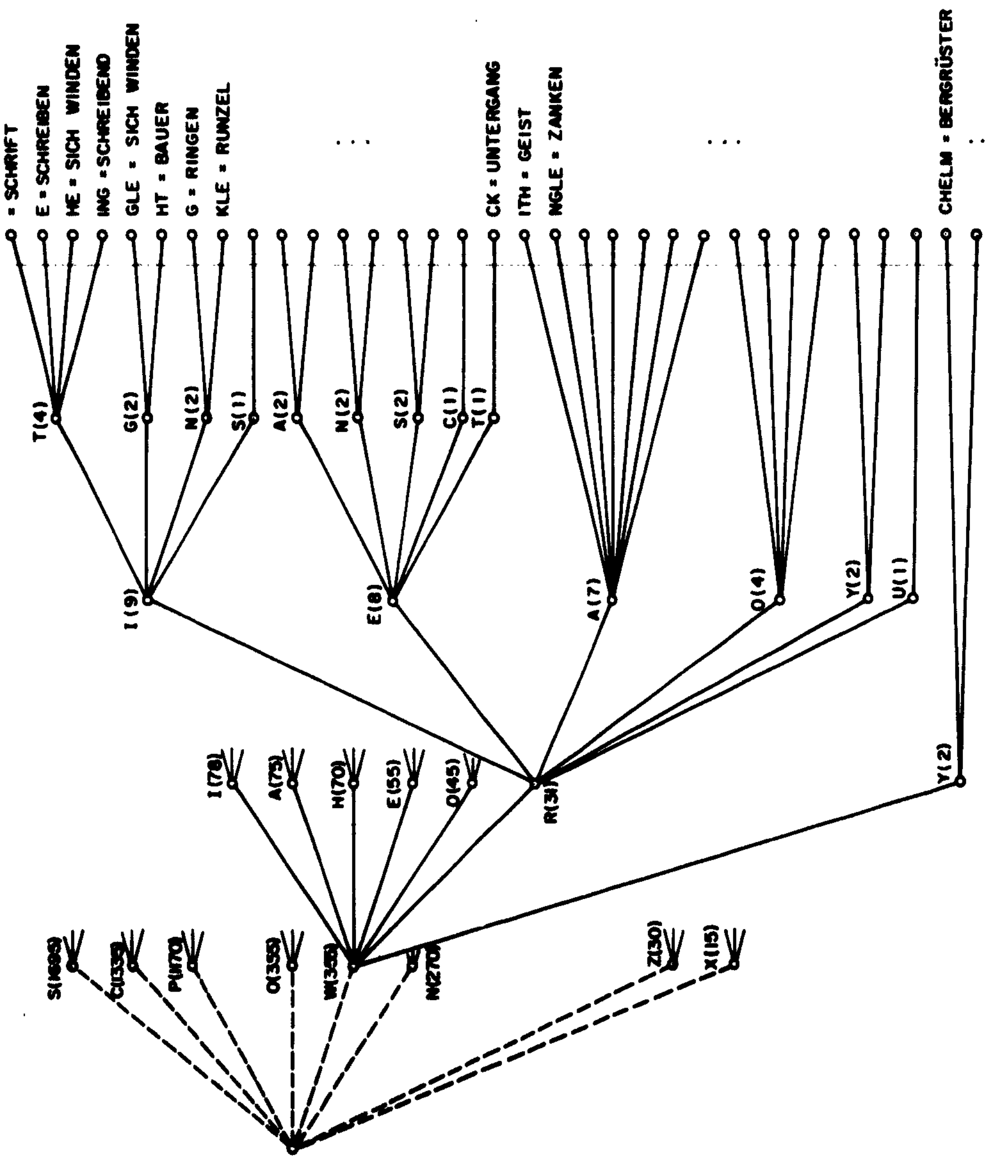


Relative Cost as a Function of Average Filial Set Size

Figure 14



Relative Search Time as a Function of Filial Set Size and Number of Governed Nodes



Part of Optimum Tree Using a Natural Partition

Figure 16

and g (Case C). All curves are normalized so that the minimum values are unity. In each case there is a shallow minimum, indicating that it is not too expensive in terms of time or cost if the average filial set size varies from the optimum value. Thus, for example, a speed decrease of less than 20% from optimum is observed if the actual filial set size is between 2 and 8 nodes, and, similarly the cost increases less than 20% from optimum if the actual size is between 3 and 12.

In summary, therefore, if it is possible either to select or to manipulate the sizes of the filial sets, they should be chosen to be in the range of 4 to 8 nodes per set for most efficient operation. In this case each path from root to item will have the same length. If it is not possible to choose the filial set size, the most efficient operation will be achieved when the path lengths vary, and the optimum path length is determined by terminating the path at any node which governs six or fewer items.

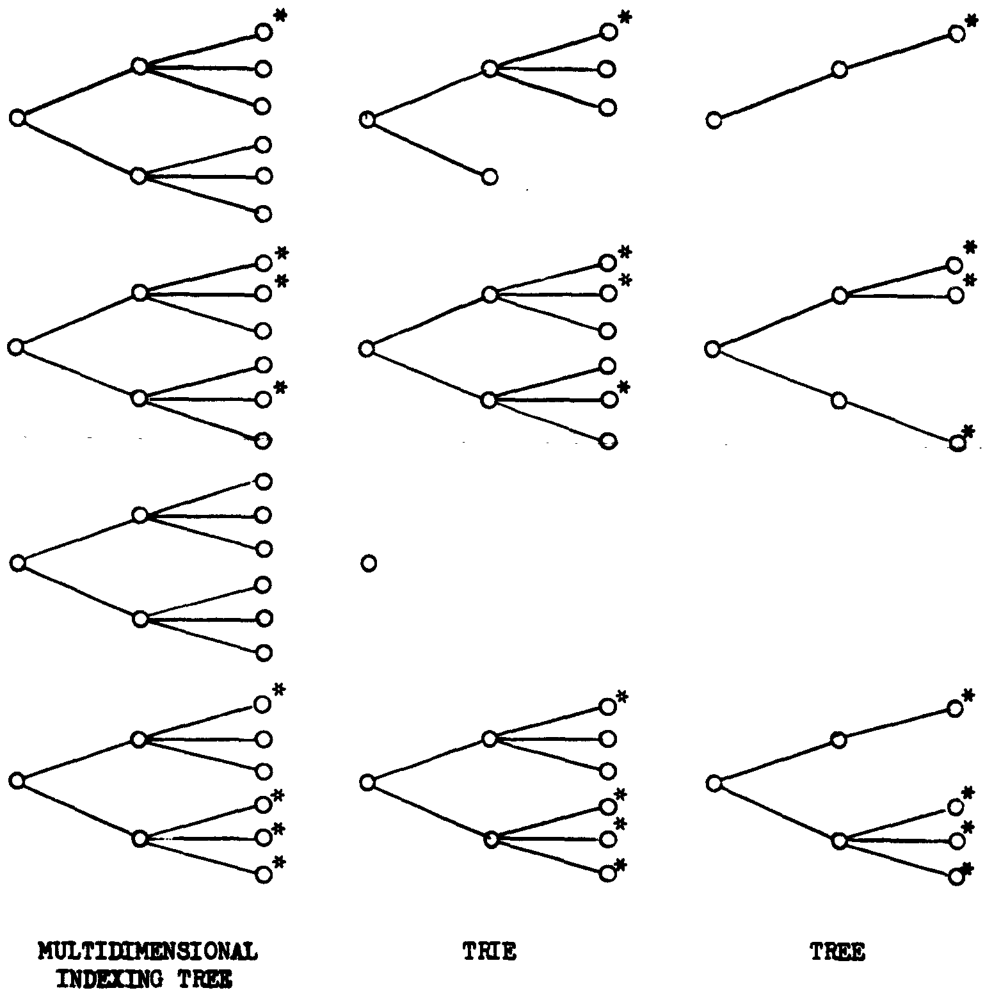
6. Multidimensional Indexing, Tries, and Trees

Multidimensional indexing techniques[6] provide a means for partitioning a file into subfiles which is simpler than the tree structure and also permits more rapid entry to a subfile than the tree. An h-dimensional indexing arrangement is essentially an h-dimensional array of addresses. Element $(i_1 i_2, \ldots, i_h)$ of the array indicates the address of the subfile composed of those items whose first h key elements are key element i_1, key element $i_2, \ldots,$ key element i_h. Such an array requires $n_1 \times n_2 \times \cdots \times n_h$ computer words of storage (where n_j is the number of elements in the jth key position). In most data processing applications, many of the entries are not used and, hence, are wasted. Moreover the size of the subfiles is not uniform, thereby increasing the expected search time.

An h-dimensional array may be loosely thought of as an h-level tree with the jth dimension corresponding to the jth tree level. Thus, the h-dimensional indexing techniques is roughly equivalent to an h-level tree. In this tree each node possesses a complete filial set; that is, all filial sets on the jth level have n_j elements. This is equivalent to assuming that each key element may be followed by every key element of the next level, no elision taking place because of nonoccurrence of element pairs. See Fig. 17.

<u>Example</u>: A three-dimensional indexing technique for the English dictionary requires 26 x 27 x 27 = 18,954 locations. The size of the subfiles varies greatly: empty for many letter triples; one item for "aja," 113 for "int." The equivalent tree has 26 nodes on the first level and 27 nodes in every filial set of the tree. Thus nodes such as "q" (which needs only one node in its sib set) and "wk" and "wsq" (which need no filial sets) all have filial sets of 27 elements.

An obvious modification to this tree structure is to eliminate the filial sets which are never used. That is, if one or more nodes of a filial set are used as parts of items, that complete filial set is retained in the tree; if none of the nodes are used, that filial set is removed. See Fig. 17. Notice that there still are nodes remaining in the tree which are never used; if these nodes are also removed, the



Comparison of a Multidimensional Indexing Tree, a Trie, and a Tree

Figure 17

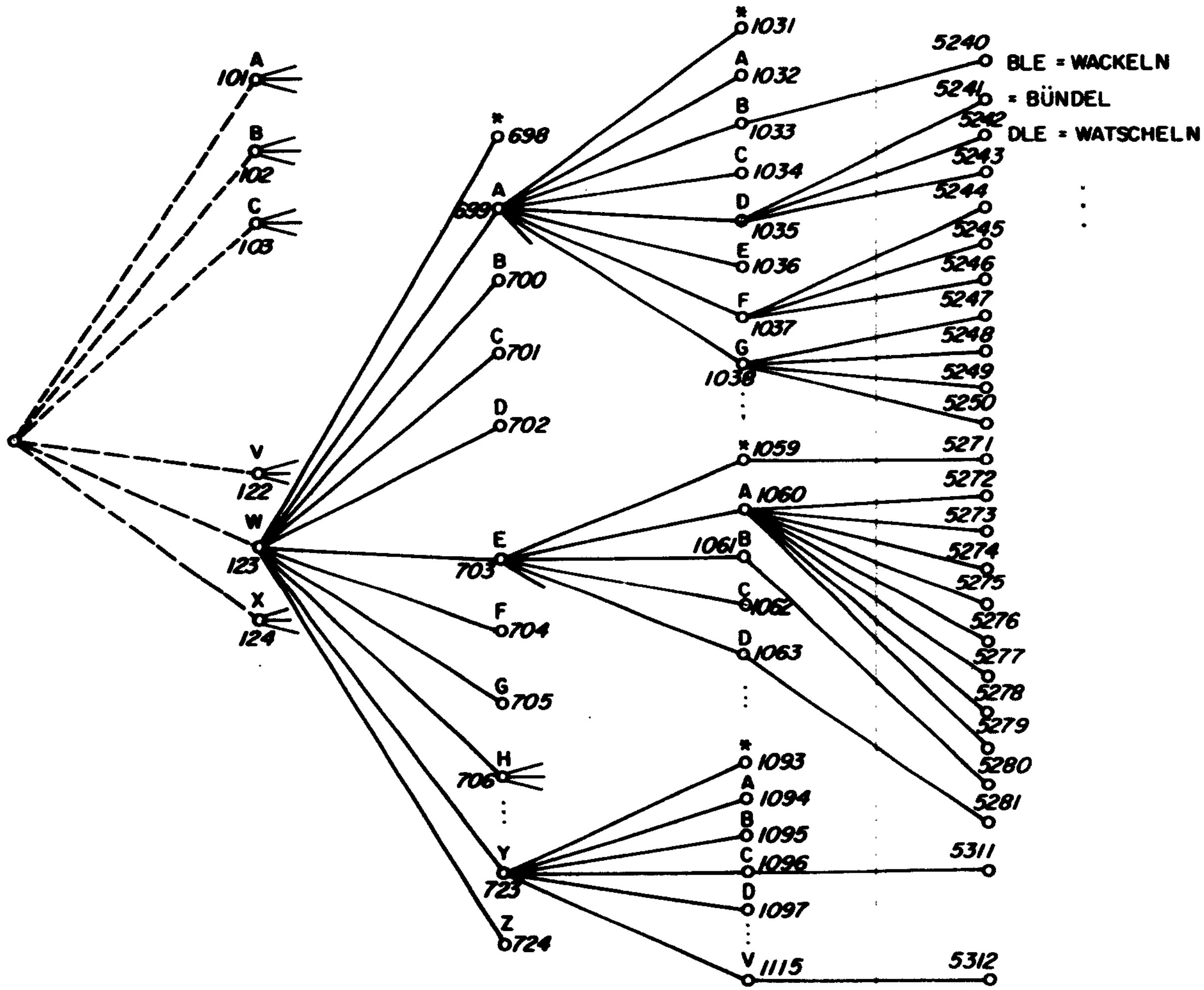
structure reduces to the tree discussed in the preceding parts of this paper. See Fig. 17. In some cases, however, the retention of these unused nodes is useful, because all filial sets in the tree are complete and may be conveniently entered by simple indexing, eliminating the node-by-node scanning of the set required for incomplete filial sets.

This modification which removes unused filial sets is essentially the tree organisation described by Fredkin[7] and dubbed a trie (retrieval). The University of California at Berkeley[8] uses a trie (although they do not call it a trie, or even a tree) for their language translation dictionary of 20,000 words.

The representation of a trie in a computer is similar to that of a tree. As before, one computer word is made to correspond to one tree node, and the nodes of one filial set are represented by continguous computer words. The computer word need contain only one field, the chaining address to link the node to (the first word of) its filial set. The key element value of the node need not be stored as its value is implicit in the position of the node within the filial set.

Example: Figure 18 shows the four-level trie which corresponds to the tree of Fig. 8. Note the filial set of the node "w" contains the complete alphabet, but only the nodes "wa," "we," "wh," etc. have filial sets themselves. These filial sets also contain the complete alphabet. Figure 19 shows the memory map for the trie of Fig. 18.

Retrieval from a trie structure requires essentially only one indexing manipulation for each level of the trie; thus the expected search time is proportional only to the average path length, h, rather



A Four Level Trie

Figure 18

101	128
102	155
103	182
⋮	
122	671
123	698
124	725
⋮	
698	---
699	1032
700	---
701	---
702	---
703	1059
704	---
705	---
706	---
⋮	
723	1093
724	---
⋮	
1031	---
1032	---
1033	5240
1034	---
1035	5241
1036	---
1037	5244
1038	5247
⋮	
1059	5271
1060	5272
1061	5280
1062	---
1063	5281

1093	---
1094	---
1095	---
1096	5311
1097	---
⋮	
1114	---
1115	5312
1116	---
⋮	
5240	BLE = WACKELN
5241	= BUNDEL
5242	DLE = WATSCHELN
5243	

The Storage Map of the Trie of Fig. 18

Figure 19

than $\frac{h}{2}(s + 1)$ as in the tree structure. This speed advantage is compensated for by requiring more storage locations, however. To estimate the number of locations, let n be the (average) number of elements possible in each key element position and s be the (average) number of elements utilized in each position. The trie requires n nodes on the first level and n nodes in each filial set of every other level. On the average, s of the nodes in each filial set will have filial sets themselves. Thus, the total number of nodes in the trie is:

$$n + sn + s^2n + \dots + s^{h-1}n = n\,\frac{s^h - 1}{s - 1}. \qquad (10)$$

A comparison of Equation (8) with (10) shows that the storage requirements of the tree and the trie are in the ratio s/n.[†]

$(s/n)^h$ may be considered to be the density of key words which actually occur in usage in the set of all possible key words. For most practical data processing applications (e.g., automatic dictionary, personnel, file, inventory records, etc.) it does not seem that the density is very high.

<u>Example</u>: For the dictionary n = 26 for the first level and n = 27 for all other levels. s = 26 for the first level, also. Vowels on the first level have filial sets with s near 26, consonants have s about 8 or 10; thus an average s for the second level is about 15. For the third level, it appears that s is about 10, and for

---

[†]Since the trie requires a shorter word length than the tree (one address field vs. two (possibly one) address fields and one value field), the ratio 2s/n might be more appropriate.

subsequent levels s appears to be less than 5. (Detailed analysis of these data has not been made.) Thus s/n for the dictionary considered is about 1/2 considering the first four or five levels.

If s/n is near unity, the trie organization is preferable to the tree as its search is much faster and its representation requires about the same storage space. In this case, however, the total number of storage locations required is greater than n^h; h-dimensional indexing requires n^h locations and its expected search time is less than that of the trie. Thus if $s \approx n$, it appears that a multidimensional-indexing organization is preferred to either a tree or a trie.

If s is significantly less than n, the speed advantage of the trie is more than compensated for by the excess storage requirement, unless the random access memory can accommodate this excess. If s is quite small, it was suggested that two or more levels be combined to yield an s closer to the optimum value. Such a technique would be disastrous with the trie organization because the number of possible key elements in such a combination grows very rapidly (e.g., n^2 for two levels), reducing the over-all density enormously.

Thus the trie may be considered to be an organization intermediate to the tree and to multidimensional indexing, and it may be considered to be a special case of each. If the density is almost unity, the multidimensional indexing techniques are superior; if the density is far from unity, the tree organization is superior; somewhere in between the trie may be used advantageously.

7. Summary

The tree organisation of the keys of a large file has been reviewed. Such a structure was found to be useful either when the entire file is stored within a random access memory or when the bulk of the file is held on a disc or drum memory.

The most important characteristic of the tree organisation is that it can be both searched and altered efficiently. The expected branch and add-item times were shown to be proportional to $s \log_s n$, where s is the average filial set size and n is either the number of items in the file for random access memories or the number of tracks for a disc memory. The optimum s was found to be between 3 and 8, and, using the optimum s, the expected search time was calculated to be only 25% slower than the binary search.

Acknowledgment

This investigation was begun under the supervision of L. R. Johnson of the IBM Research Center. He and K. E. Iverson, also of IBM Research, suggested the basic tree search technique.

REFERENCES

1. Sussenguth, E. H., "A Study of Procedures for Frequency Counting of Words in Running Text," Report ISR-4, The Computation Laboratory of Harvard University (to be published).

2. IBM Corporation, "Fortran Assembly Program (FAP) for the IBM 709/7090," IBM Form J28-6098-1 (July 1961).

3. Iverson, K. E., A Programming Language, Wiley (1962).

4. Betteridge, H. T. (ed.), The New Cassell's German Dictionary, Funk and Wagnalls, New York (1958).

5. Johnson, L. R., "On Operand Structure, Representation, Storage, and Search," Research Report No. 603, IBM Corporation (1962).

6. Hellerman, H., "Addressing Multidimensional Arrays," CACM (April 1962).

7. Fredkin, E., "Trie Memory," CACM (September 1960).

8. Lamb, S. M., Jacobsen, W. H., "A High-Speed Large-Capacity Dictionary System," Mech. Transl. (November 1961).

9. Newell, A. (ed.), Information Processing Language - V Manual, Prentice-Hall (1961).

APPENDIX

DERIVATION OF THE RESULTS FOR THE OPTIMUM EXPECTED SEARCH TIME

Equation (1) gives the definition of the average filial set size for tree level i. Since the number of filial sets on level i is just the number of nonleaf nodes on level i - 1, an equivalent definition, which is sometimes more convenient is

$$s_i = \frac{\text{number of nodes on level } i}{\text{number of nonleaf nodes on level } i - 1} \qquad \text{(A1)}$$

(and s_1 = number of nodes on level 1).

For Case A it is assumed that the blocks are searched on an item-by-item basis so that $t(B) = \frac{1}{2}(B + 1)$. Thus Equation (3) becomes

$$\bar{t} = \frac{1}{2}\sum_{i=1}^{h} (s_i + 1) + \frac{1}{2}\frac{N}{\prod_{i=1}^{h} s_i} + 1. \qquad \text{(A2)}$$

From the symmetry of Equation (A2) in $s_1, s_2, \ldots, s_h$ it is clear that the expected search time is independent of the order in which the levels are taken and that the minimum search time is achieved when the s_i are all equal. The latter assertion is proved by setting the derivative of (A2) with respect to s_k equal to zero. This yields

$$s_k \prod_{i=1}^{h} s_i = N. \qquad \text{(A3)}$$

By letting $k = 1,2,\ldots,h$ one finds

$$s_1^2 s_2 \cdots s_h = s_1 s_2^2 \cdots s_h = \cdots = s_1 s_2 \cdots s_h^2 = N$$

so that

$$s_1 = s_2 = \cdots = s_h = N^{1/h+1}. \qquad \text{(A4)}$$

This means that the time spent searching the block at the end of the tree should be the same as the time spent searching a filial set.

That is, the block size should be $N^{1/h+1}$, so that the entire structure may be viewed as a tree of h + 1 levels. Setting

$$L = h + 1 \tag{A5}$$

(A2) and (A4) becomes

$$\bar{t} = \frac{L}{2}(s + 1) \tag{A6}$$

and

$$s = N^{1/L}. \tag{A7}$$

If the file is stored on a drum or disc as in Case B and not in random access memory, the block size, B, is normally determined by the number of items, T, which can be accommodated by one track and not by considerations which minimise the over-all search time. The tree acts as the transformation between the set of keys and the set of track addresses.

In this case B = T, and the tree has h levels, where h is the largest k which satisfies:

$$\prod_{i=1}^{k} s_i \leq \frac{N}{T}. \tag{A8}$$

Since B is fixed by an external constraint, it is safe to assume $\bar{t}(B)$ is also fixed, so that the problem of minimising the over-all expected search time is reduced to choosing the s_i so that

$$\frac{1}{2}\sum_{i=1}^{h} (s_i + 1) \tag{A9}$$

is minimised, subject to constraint (A8).

Again it is clear that the minimum is achieved when

$$s_1 = s_2 = \cdots = s_h = s.$$

Then

$$\bar{t} = \frac{h}{2}(s + 1) \tag{A10}$$

and

$$s = \left(\frac{N}{T}\right)^{1/h}. \tag{A11}$$

Equations (A10) and (A11) have exactly the same form as (A6) and (A7). From either pair one obtains

$$\bar{t} = \frac{s + 1}{2} \log_2 M. \tag{A12}$$

It is possible to determine a numerical value for the optimum filial set size from either pair of equations. Minimizing

$$\bar{t} = \frac{h}{2}(s + 1) \tag{A13}$$

subject to the constraint

$$s = M^{1/h}, \tag{A14}$$

leads to the requirement that

$$s + 1 = s \ln s$$

which has a solution at $s = 3.6$. Figure 13 displays (A13) subject to (A14) normalized to $\bar{t}$ opt = 1.

In Case C when the file fits within random access memory, the optimum path lengths are determined by the following argument.

A path leading to a block of items is considered to have its optimum length if the expected search time for items in the block is increased when the path length is changed. To determine the optimum length for a path, assume a node x of the tree governs g items and has s nodes in its filial set. Then the nodes of the filial set each govern an average of $\frac{g}{s}$ items. If the branching is discontinued at node x, the expected search time from x is

$$t_d = \frac{1}{2}(g + 1). \qquad \text{(A15)}$$

If, however, the branching is continued for one more level, the expected search time from x is

$$t_c = \frac{1}{2}(s + \frac{g}{s}) + 1. \qquad \text{(A16)}$$

Thus, the expected search time from x will decrease if the branching is continued when $t_d > t_c$.

$t_d > t_c$ when

$$s^2 + (1 - g)s + g < 0. \qquad \text{(A17)}$$

The zeros of (A19) are at

$$\sigma_1, \sigma_2 = \frac{g - 1 \pm \sqrt{g^2 - 6g + 1}}{2} \qquad \text{(A18)}$$

Thus if $g^2 - 6g + 1 \geq 0$, there is a range of (real) s for which adding one more level does decrease the expected search time. Conversely, if $g^2 - 6g + 1 < 0$, there is no such value. The zeros of $g^2 - 6g + 1$ are at $3 \pm 2\sqrt{2}$.

Therefore, the expected search time will be decreased if branching is continued from any node which has

$$g \geq 6 \quad (A19)$$

and

$$\sigma_1 \leq s \leq \sigma_2 \quad (A20)$$

Figure 15 displays (A17). For any node with its (g,s) lying in the pie-shaped region to the right of the solid curve of Fig. 15, (A19) and (A20) are satisfied; that is, the search time can be decreased by continuing the branching from that node. The dashed curves indicate the relative improvement in the search time between the cases of continuing and discontinuing the branching from the node.

<u>Example</u>: See Figs. 6, 7, and 16. Conditions (A19) and (A20) are certainly satisfied for all nodes on the first level of the tree for Cassell's Dictionary. They are also satisfied for all the nodes on the second level of the "w" subtree except the "wy" branch; the "wy" node should not branch but indicate the location of the list "wychelm," "wyvern." Similarly (A19) and (A20) are satisfied for nodes "wre" and "wri," but not for the other nodes of the sib set of "wr": node "wra" satisfies (A19) but not (A20), the others do not satisfy (A20).

Notice that the argument leading to (A19) and (A20) involves adding one level at a time. In rare cases it might be advisable to add a level which increases the expected search time if subsequently another level is added which decreases the over-all time.

<u>Example</u>: The first level node "q" of the tree of English words has $s = 1$. Hence adding the second level, to form "qu," increases the search time. However, adding a third level ($s = 4$: "qua," "que," "qui," and "quo") substantially decreases the search time.

IV. AN INTERPRETIVE PROGRAM FOR CORRELATING LOGICAL MATRICES

Michael Lesk

Several experiments in automatic content analysis have been conducted in which properties of a document collection were conveniently expressed by logical matrices. For example, a subject index may be represented by a matrix in which each row corresponds to an index term, and each column to a document. The documents associated with each term are then indicated by 1's in the proper columns. Bibliographic citations may also be represented in this form; specifically, logical matrices were used recently to compute an index of relations between citations and document content.[1] The present report describes a computer program designed to manipulate logical matrices in accordance with the specifications outlined in ref. 1. The following operations are needed: matrix transposition, Boolean multiplication, and correlation of logical matrices. Two types of correlations may be produced: row correlations, which compare rows of the same matrix, and cross correlations, which compare two different matrices.[2]

The program is written for the IBM 7090 computer, which has binary logic instructions permitting efficient storage and processing of matrices. The number and size of the logical matrices to be processed is limited only by the space available in 7090 core (there is a normal limit of 25 matrices, but this can be raised easily as described in the appendix). Numerical matrices of row correlations may not be placed in core storage, but are written on intermediate tapes. Logical matrices are referred to at all times by their names, consisting of any five BCD characters (except five blanks). Each matrix is assigned a unique name when it is read into storage or generated by the program.

The input cards to the program consist of instructions and (where needed) descriptions of matrices. The instruction cards include a pseudo-operation code beginning in card column 1, extending as far as necessary, and terminating with a blank column. The names of any matrices or tapes to be used in the instructions are written following this blank column. Operation codes recognized by the program include the following:

ASYMLOGIN NAME1 - Asymmetric logical matrix input. This instruction causes the program to generate a matrix, named NAME1, as specified by the cards following the operation card. "Asymmetric" implies that the rows and columns are not referred to by the same names, and has nothing to do with the symmetry properties of the matrix elements. Immediately after the ASYMLOGIN card a list of the names of every row of the matrix, called the "row identifiers," follows. Each name consists of five characters (not all blank) and is followed by a blank. This permits twelve identifiers to be punched in one card, columns 1-72. As many cards as needed are used, the end of the list being signaled by six consecutive blank columns (on a new card, if need be). The next set of cards after this sentinel contains the "column identifier" list in precisely the same format (with the same ending sentinel), but specifying the names of the columns rather than the rows. The cards which describe the actual matrix follow the column identifier list.

Each element of the matrix is specified by its row and column identifiers. The cards describing the matrix have blanks in columns 1-6, a row identifier in columns 7-11, and a list of column identifiers (with a single blank column after each identifier) in columns 13-72. The program assigns a value of 1 to the matrix element specified by the row identifier and each column identifier on that card. Up to ten elements may be specified on one card. If a row has less than eleven 1's in it, it is described by a single card with the row identifier in columns 7-11, followed by the column identifier for each column in that row at which there is an element with the value one. A row with more than ten 1's in it will require more cards to specify it; a row identifier may occur on any number of cards. If a row has no 1's at all, no card is needed for that row. The order of the column identifiers on a card, or of the cards that specify the matrix, is immaterial, but all matrix specification cards must follow all cards giving the identifier lists. The end of the specification cards is marked by a card containing something other than six blanks in the first six columns (this card will be the next pseudo-operation). A card in the matrix specification section with a row identifier not given in the list of row identifiers which preceded the matrix specification will cause an error notation on the output copy. Aside from this notation, such a card will be ignored. If a card contains a column identifier that was not given in the column identifier lists, an error notation will be made and that identifier ignored. The program will try to interpret the remainder of the card, however. Within the lists of row and column identifiers, each identifier must be given only once. If any identifier is repeated in an identifier list, the program will detect the second occurrence, make an error notation, and ignore the repetition.

SYMLOGIN NAME1 - Symmetric logical matrix input. This pseudo-operation reads in a matrix whose rows and columns have the same names. Only one identifier list, which serves for both rows and columns, is given; otherwise, the operation is identical with ASYMLOGIN.

LOGWRITE NAME1 T - Write logical matrix on tape. There are two intermediate tapes, denoted P and Q. To write a logical matrix on one of them, this instruction is used with either the character P or the character Q replacing T in the instruction. NAME1 is written on the tape as a two record file.

LOGREAD T NAME1 - Read logical matrix from tape. A matrix written by a LOGWRITE instruction is read from tape (again, T must be either P or Q), and named NAME1. If a matrix written by a LOGWRITE is not in position on the specified tape, there will be an error printout, and the operation will not be executed. NAME1 may be omitted from the instruction; if no name is given, the matrix will retain the name it had when it was written on the tape.

FREE NAME1 NAME2 NAME3 ... - Free storage areas used by NAME1, etc. (operation terminates when six blank columns are encountered on the instruction card). This card tells the program that the storage being occupied by NAME1, etc. is now available for use by other matrices. No further references may be made to NAME1, etc. An alternative method of freeing a matrix is to insert an F after its name in any pseudo-operation that refers to the matrix. This causes the matrix to be freed after the pseudo-operation is performed. Thus, LOGWRITE NAME1F P is equivalent to LOGWRITE NAME1 P followed by FREE NAME1; either will cause NAME1 to be written on tape P, and then freed so that the core storage it occupied becomes available for further operations. To decide when matrices must be freed, the following formula is used: let R be the number of rows of a matrix, C the number of columns, and let N equal C/36 (add one if any remainder). Then the number of words of core occupied by the matrix is 6 + C + R(N + 1). CR/36 + C + R is a good approximation. Whenever the total space used by matrices would exceed 29,000 locations, matrices must be freed or an error printout will result.

TRANSPOSE NAME1 NAME2 - Transpose logical matrix. NAME1 is transposed and the result called NAME2. Note that during the execution of this instruction, the storage space required is equal to twice that for NAME1 plus the storage for NAME2. If the instruction is of the form TRANSPOSE NAME1F NAME2, however, the space needed is only that for NAME1 and NAME2.

MULT NAME1 NAME2 NAME3 - Multiply logical matrices (Boolean multiply). Matrix NAME3 is defined as follows: its row identifier list is the row identifier list of NAME1, its column identifier list is the column identifier list of NAME2, and its elements are defined by normal multiplication of matrices, with "multiply" replaced by "logical and" and "add" replaced by "logical or" throughout the definition of matrix multiplication. That is, if we call <u>A</u> the matrix named NAME1, <u>B</u> the matrix NAME2, and $\underline{C}$ the product matrix NAME3, then $\underline{C}_j^i = 1$ if and only if there exists a k such that $\underline{A}_k^i$ and $\underline{B}_j^k$ are both 1. The column identifier list of NAME1 must be the same as the row identifier list of NAME2 for this pseudo-operation.

MULTX NAME1 NAME2 NAME3 - Boolean matrix multiplication by transpose. NAME3 is defined to have the row identifier list of NAME1, a column identifier list which is the same as the row identifier list of NAME2; element $\underline{C}_j^i = 1$ if and only if there is a k such that $\underline{A}_k^i$ and $\underline{B}_k^j$ are both 1. Thus NAME3 is the product of NAME1 by the transpose of NAME2. It should be noted that MULT is performed by executing a TRANSPOSE followed by a MULTX, so that MULTX is faster than MULT.

ROWTAPE NAME1 T - Row correlate NAME1, write the results on the output tape and on T. Row correlations are comparisons between rows of a matrix. The correlation factor, r, between rows $\underline{X}^i$ and $\underline{X}^j$ of a matrix $\underline{X}$, is defined

$$r = \frac{\sum_{k=1}^{m} \underline{X}_k^i \underline{X}_k^j}{\sqrt{\sum_{k=1}^{m} (\underline{X}_k^i)^2 \cdot \cdot \sum_{k=1}^{m} (\underline{X}_k^j)^2}},$$

where matrix $\underline{X}$ has m rows. ROWTAPE computes the correlation for each pair of rows in NAME1 (unless a SELECT pseudo-operation has been executed; see that operation), and writes these correlations on the output tape and the tape specified by T (either P or Q). The output appears in a triple column format with 56 lines per page. Each output item consists of two row identifiers and the correlation factor, thus:

ROW13 ROW14 .4937.

The number of 1's in each row of the matrix is also written on the output tape and is called the row sum.

ROWTAPESUPP NAME1 T - Row correlate, write on intermediate tape. This operation is the same as ROWTAPE except that the row correlations are not written on the output tape. The row sums are written on the output tape as before, however.

ROWPRINT NAME1 - Row correlate, print (off-line). Same operation as ROWTAPE, except that nothing is written on an intermediate tape.

LOGCROSS NAME1 NAME2 - Cross correlate NAME1, NAME2. Cross correlations are a measure of similarity between sets of row correlations derived from two distinct logical matrices. For every row in a logical matrix we can form a set of row correlations of the row; this set is a vector $\underline{R}^i$, where the elements $\underline{R}^i_j$ are the row correlations of row i with row j. If two matrices have the same row identifier lists, we may define a corresponding $\underline{S}^i$ for the same row in the other matrix. Now, the cross correlation between row i in these two matrices is defined as

$$\underline{X}_i = \frac{\sum_k \underline{R}^i_k \underline{S}^i_k}{\sqrt{\sum_k (R^i_k)^2 \cdot \sum_k (S^i_k)^2}} .$$

LOGCROSS computes these cross correlations from the logical matrices by first computing the row correlations for a row in each matrix, and then the cross correlation for this row. The row correlations, although computed as intermediate data, are not available as output. An "over-all" cross correlation is also produced; this is the quotient

$$\frac{\sum_i \sum_k \underline{R}_k^i \underline{S}_k^i}{\sqrt{\sum_i \sum_k (\underline{R}_k^i)^2 \cdot \sum_i \sum_k (\underline{S}_k^i)^2}} .$$

LOGCROSS writes on the output tape the row sums of each matrix, the cross correlation for each row, and the over-all cross correlation.

TAPECROSS - Cross correlate two matrices on tape. If there is a numerical matrix of row correlations on tape P, and another on tape Q (written by ROWTAPE and ROWTAPESUPP operations), and both matrices have the same row identifiers, TAPECROSS will compute the cross correlations for each row and the over-all cross correlations, and write them on the output tape. If tapes P and Q do not both contain numerical matrices with the same row identifiers, error printouts result.

SELECT NAME1 - Selects row identifiers. If two matrices have different row identifier lists, they cannot be processed directly by LOGCROSS and TAPECROSS. If portions of the identifier lists coincide, however, and it is wished to correlate some or all of the common rows, the SELECT pseudo-operation may be used. It must be followed by an identifier list in ASYMLOGIN format. The program will flag all row identifiers in NAME1 **not** given in the list supplied after the SELECT card, and all such flagged identifiers will be ignored during computation of row correlations. Thus if the unflagged portions of the identifier lists of two matrices are the same, they may be cross correlated, although the flagged portions of the identifier lists are different.

BACKSPACE T_1 m T_2 n - Backspace tape. Tape T_1 (either P or Q) is moved backwards m files (one logical matrix or one set of row correlations is one file), and tape T_2 is moved backwards n files. If either m or n is blank, it is interpreted as 1. If T_2 is blank, only one tape is moved. If $T_2 = T_1$, the second specification overrides the first.

DATE iiiiiiiiiiiiiii - The fifteen characters following DATE are copied onto the page heading.

END - Stops the run. This should be the last card of any deck of instructions. If it is inadvertently omitted, the program will stop when an end-of-file appears on the input tape.

As assembled, this program will run under the FORTRAN monitor system on any 7090 with 32K storage and sufficient tape units (two channels, five tapes on each). The program reads BCD unblocked records from A2 and writes BCD unblocked records on A3 as output. The intermediate tapes used are A5 (P) and B5 (Q). Output is to be printed PC; nothing is printed past position 72 on the page.

In a standard FMS system, the following cards are needed:

(1) job ID card for FAS sign-on record; check individual system requirements;

(2) *XEQ (*in column 1; X, E, Q in columns 7-9);

(3) binary program deck, in normal relocatable column binary format;

(4) *DATA (*in column 1; D, A, T, A in columns 7-10);

(5) instructions and other data, terminated by an END card.

Scratch tapes must be mounted on tapes P and Q (A5, B5).

(Note: To date it has never been necessary to use the pseudo-operations TAPECROSS and SELECT. As a result, these two instructions cannot be guaranteed to operate correctly.)

APPENDIX A

PROGRAM PARAMETERS

Some program parameters may be changed without excessive difficulty. Four particularly important cards in the FAP deck are (all near the beginning):

Cols. 1-6	Cols. 8-15	Cols. 16-72
LNADTB	EQU	50
MAXST	EQU	29000
P	TAPENO	A5B
Q	TAPENO	B5B

LNADTB is twice the absolute maximum number of matrices that may be kept in 7090 core; it is presently set for 25 matrices. MAXST is the main storage area, and determines the size of the storage area for matrices and other temporary storage (such as I-O buffers; none of these other storage areas is likely to be large). At the moment, the distribution of core storage is

~~Inaccessible lower and upper core areas:~~	~~306 locations~~
Main program, lower storage:	3,178 locations
Subroutine EXIT, lower storage:	18 locations
Subroutine SQRT, lower storage:	44 locations
Address table (LNADTB), lower storage:	50 locations
Main storage area, upper storage:	29,000 locations
Unoccupied storage:	172 locations
Total 7090 core storage:	32,768

In changing the input and output tapes, it must be remembered that tape reading and writing is <u>not</u> done through (IOU) and reassembly will be required if the system configuration is changed from the version in the FORTRAN manual. The input tape is defined by a card R TAPENO A2; the output tape is defined when FD9OUT is called to write a line of output ((STH) is not used). To change the output tape, use the symbolic reference table in the assembly listing to find calls to OUT and OUTNC, and the SHARE writeup of FD9OUT (with the comments in the listing) to determine how to alter the calling sequence to change the output tape.

APPENDIX B

SAMPLE PROGRAM

The sample program investigates the relation between citations and content as outlined in ref. 1. It uses two citation correlation matrices, CITNG and CNG2, and computes their cross correlations with the term correlation matrix TDCMP. There are two input matrices, CITED and TTCMP. CITED represents a citation index; if we call this matrix $\underline{A}$, then $\underline{A}^i_j = 1$ if document j cites document i. TTCMP represents a subject index; calling TTCMP $\underline{B}$, $\underline{B}^i_j = 1$ if term i applies to document j. The program generates from these matrices the needed matrices CITNG, CNG 2, and TDCMP. It then computes the row correlations for CNG 2 and the cross correlations for both CITNG and CNG 2 with TDCMP.

The program, complete with all descriptions of matrices, is given in the attached listing; the instructions are reproduced here:

DATE OCT. 18, 1962	This instruction causes OCT. 18, 1962 to appear on the top of each page of output.
SYMLOGIN CITED	Reads in CITED. The next six cards give the identifier list; matrix specifications follow. Note that only one identifier list is given.
TRANSPOSE CITED CITNG	CITNG, the matrix of citations produced by transposing CITED, represents the actual bibliographies of the documents; each row

	corresponds to the references pertaining to one document.
MULTX CITNG CITEDF CNG 2	This instruction produces CNG 2, the second order citation matrix. In CNG 2, A_j^i = 1 if there is a document k such that i cites k and k cites j. By reference to the definition of MULTX, it will be seen that CNG 2 is precisely the matrix produced. CITED is no longer needed and is freed.
ROWPRINT CNG 2	The row correlations of CNG 2 are computed and written on the output tape. They are not stored internally or on intermediate tapes and will not be available for future processing.
ASYMLOGIN TTCMP	TTCMP, the subject index, is read in from the input tape. Note that two identifier lists are given; there are fifty-six terms and sixty-two documents, each with a distinct set of names.
TRANSPOSE TTCMPF TDCMP	TTCMP is transposed to get a matrix with the same row identifier list as CNG 2 and CITNG. TDCMP, the transposed matrix,

	represents a set of descriptors; calling it $\underline{A}$, $\underline{A}_j^i = 1$ if term j applies to document i. TTCMP is freed.
LOGCROSS TDCMP CITNG	Cross correlations are produced for TDCMP and CITNG. This is done by computing the row correlations for each row of each matrix internally, and cross correlating the row correlation vectors, row by row. A cross correlation for each row and an over-all cross correlation are produced. Row correlations are not written on tape for printing.
LOGCROSS CNG 2 TDCMP	Cross correlations are computed for TDCMP and CNG 2, as above.
END	Stops the run.

```
DATE OCT. 18, 1962
SYMLOGIN CITED
BO319 BO407 BO408 CH504 CO208 CO305 CO317 CO402 CO403 FO409 FO415 FO496
FO506 FO603 FR205 FR303 FR304 FR309 FR405 FR406 FR612 GB604 GE302 GI209
GI491 GI495 GR202 GR204 GR313 IS311 IS312 IS411 JO201 JO310 JO314 JO316
JO413 JO607 JO609 JO611 KU605 LE606 LY414 LY602 MA301 MG404 MG493 MJ203
MJ306 MJ318 MJ494 MJ505 OE492 PL315 PL601 RO207 SA401 SH206 VC307 VS308
WA410 WA412
          BO319 CO317 FO415 IS311 PL315 VC307
          BO407 GB604 KU605 PL601
          BO408 BO407 PL601
          CH504
          CO208 BO319 CO317 FO415 FR205 JO609 LE606 SH206 VS308
          CO305 VS308
          CO317 FR405 FR612
          CO402
          CO403 JO609 LY602
          FO409 IS411 JO413 WA410 WA412
          FO415 IS411 JO609 LE606
          FO496 JO609
          FO506 FO415 JO609 IS311
          FO603 PL601
          FR205 BO319 CO403 FR304 IS311 JO310 JO607 SA401 VS308
          FR303 CO317 VC307 VS308
          FR304 CO403 FO409 VS308
          FR309 JO310 JO413 PL315 VS308
          FR405 CO403
          FR406 CO403
          FR612
          GB604
          GE302 CO317 JO316 VS308
          GI209 BO319 JO314 MA301 SA401 VS308
          GI491 CH504 FO496 GI495 JO201 JO609 MJ203
          GI495 FO496 JO201 JO609
          GR202 PL315
          GR204
          GR313 CO317 FO409 FR304 IS411 WA410 WA412
          IS311 JO609
          IS312 FR304 JO316 JO609
          IS411 WA412
          JO201 BO319 BO407 BO408 CO317 CO403 FR205 FR309 GR202 GR204 JO310
          JO201 JO314 JO316 JO413 JO607 JO611 SA401 VS308
          JO310 CO317 FR309 FR405 GB604 MA301 PL315
          JO314 JO611 MA301 MG404 SA401
          JO316 CO403 IS411 JO413 JO607 WA412
          JO413 JO607
          JO607
          JO609
          JO611
          KU605
          LE606
          LY414 FO603 JO609 LY602
          LY602 FO603 JO609
          MA301 JO314 SA401
          MG404 CO403 JO609
          MG493 FO496 GI495 GI209 JO609 MJ203 MJ494 MJ505 SH206
          MJ203 JO413 JO609 LY602 SH206
          MJ306 CO317 CO403 IS311 JO609 PL315
```

```
        MJ318 J0609 C0403 PL315
        MJ494 J0609 MJ505
        MJ505 C0208 F0409 FR405 GI209 J0310 J0609 LY602 MJ203 MG404 SH206
        MJ505 VC307 VS308
        OE492
        PL315 C0403 FR309 FR405 J0310
        PL601 F0603
        R0207 C0305 C0402 C0317 FR303 LY602 VS308
        SA401
        SH206 C0317 C0403 FR205 FR303 FR309 GE302 IS411 J0310 J0314 J0316
        SH206 PL315 VC307 VS308
        VC307 C0317
        VS308 C0305 FR303
        WA410 WA412
        WA412 F0603 IS411 J0611
TRANSPOSE CITED CITNG
MULTX CITNG CITEDF CNG 2
ROWPRINT CNG 2
ASYMLOGIN TTCMP
ADJEC BIBLI CIRCT CUMPL CMPOR CMPDV CONTX CDRUN CRECT EXTRC LOKUP DICOR
EDITG DICTE FEEDB FILOR FREQU GRAGR GRCOD HOMOG HADIC INFLE INFLR INPEQ
INPPR JAPAN MASHD MUCOR NOUNS NUMLS POEDT PRSAN PREPO PRODS PROER PRONS
RFVBS SERCH SEGMT SEMAN SEDIA SELGT SORTG STOAC STRAN SUFAN SYNAR SYNAE
TEXTS TIMNG TRNSL TRLIT UPDAT VERBS WDCLS WDSTM
B0319 B0407 B0408 CH504 C0208 C0305 C031- C0402 C0403 F0409 F0415 F0496
F0506 F0603 FR205 FR303 FR304 FR309 FR405 FR406 FR612 GB604 GE302 GI209
GI491 GI495 GR202 GR204 GR313 IS311 IS312 IS411 J0201 J0310 J0314 J0316
J0413 J0607 J0609 J0611 KU605 LE606 LY414 LY602 MA301 MG404 MG493 MJ203
MJ306 MJ318 MJ494 MJ505 OE492 PL315 PL601 R0207 SA401 SH206 VC307 VS308
WA410 WA412
        ADJEC FR405 FR612
        BIBLI C0402 R0207
        CIRCT CH504
        COMPL GI495 J0609 MJ494
        CMPOR OE492
        CMPDV CH504 OE492
        CONTX B0319 C0317 FR309 FR405 J0314 MJ203 VC307
        CDRUN GI491 GR204 J0201 J0316 J0413 J0607 VS308
        CRECT C0208 F0415 FR205 FR304 GR204 IS312 J0314 J0609 LE606 VS308
        EXTRC GE302 IS311
        LOKUP B0407 J0201 J0609 SA401
        DICOR GI491 MJ494 MJ505
        EDITG J0609 J0611
        DICTE B0407
        FEEDB GB604 GI209 J0314 MA301
        FILOR GI491
        FREQU C0305 FR205 FR303 VS308
        GRAGR C0317 FR405 FR612
        GRCOD B0407 C0208 C0403 F0409 F0496 F0506 FR303 FR309 GR313 IS411
        GRCOD J0310 J0413 LY414 LY602 MG404 MJ203 MJ306 MJ318 MJ494 MJ505
        GRCOD PL315 PL601 SH206 WA410 WA412
        HOMOG C0208 C0317 GE302 SH206 VC307
        HADIC C0208 C0403 F0415 FR205 FR304 GI209 GI491 GI495 IS312 IS411
        HADIC J0609 LE606 MG404 MJ494 MJ505 PL315 VS308
        INFLE F0409 GR313 IS411 WA410 WA412
        INFLR F0496 F0506 MG493 MJ505
        INPEQ GR202
        INPPR GR202 GR204
```

```
      JAPAN KU605
      MASHD SA401
      MUCOR J0611
      NOUNS C0317 FR405 FR612 VC307 WA410
      NUMLS MG404
      POEDT GI209 J0314 MA301
      PRSAN B0407 B0408 GB604
      PREPO MJ306
      PRODS B0319 B0407 B0408 C0208 C0317 F0409 F0415 F0496 F0506 FR205
      PRODS FR304 GE302 GI491 GI495 IS311 IS312 IS411 J0201 J0310 J0316
      PRODS J0413 J0607 J0609 LE606 SA401 SH206 VS308 WA412
      PROER J0310
      PRONS MJ318
      RFVBS LY414 LY602
      SERCH GI491 OE492
      SEGMT KU605 SA401
      SEMAN LY414
      SEDIA PL601
      SELGT C0305
      SORTG FR205 GI491 IS311 IS411 VS308
      STOAC OE492
      STRAN F0603 PL601
      SUFAN CH504 GI491 GR313 J0316 KU605 LY414 MJ203 SH206
      SYNAR B0319 F0409 F0603 FR205 FR304 FR406 FR612 IS411 J0310 LY414
      SYNAR PL601
      SYNAE B0407 B0408
      TEXTS C0402 R0207
      TIMNG GR204
      TRNSL F0603 GB604 GI209 J0314 KU605 LY602
      TRLIT CH504 GR202 KU605
      UPDAT C0208 F0415 GI495 IS312 J0609 LE606 PL315
      VERBS FR406 FR612
      WDCLS F0409 FR303 FR309 FR405 FR406 FR612 GR313 KU605 MG493 MJ203
      WDCLS MJ306 MJ318 MJ494 SH206 VC307
      WDSTM FR309 GE302 MJ203
TRANSPOSE TTCMPF TDCMP
LOGCROSS CITNG TDCMP
LOGCROSS CNG 2 TDCMP
END
```

V. A COMPARISON OF CITATION DATA FOR OPEN AND CLOSED DOCUMENT COLLECTIONS

Michael Lesk

In a study by Salton of the relation between bibliographic citations and document content,[1] the set of citing documents was assumed to be identical with the set of cited documents; that is, citations to and from documents outside the given document collection were disregarded. It was suggested, however, that the same methods might be used with open collections in which all citations would be allowed.[2] This section describes an experiment in which the same computations performed previously were repeated with an extended citation network including all citations to and from the collection, regardless of source. The code names previously used for matrices, index terms, and documents have been retained unchanged in the present description.

Three of the matrices were processed on the IBM 7090 computer in both the open and closed forms. These are the matrices CITED, CITNG, and CNG 2. In each case the larger number of documents leads to lower-row correlations in the open collection, but the relationship to content is not seriously impaired.

The change from CITED closed to CITED open resulted in a relatively small change, because a comparatively small amount of additional data was introduced. Although the number of citations rose from 165 to 295, only 21 new documents were introduced in the open collection. Furthermore, most of the new citations were from five works, which had long, exhaustive bibliographies.

The over-all cross correlation between CITED and TDCMP dropped from .405 in the closed collection to .389 in the open collection. Of the five documents with the highest cross correlations, three were common to both collections, as shown in the following table:

CITED Closed		CITED Open	
Document	Cross Correlation	Document	Cross Correlation
MA301	.753	MJ505	.689
MJ505	.735	SH206	.682
MJ306	.651	MJ203	.680
CO208	.633	MJ494	.645
SH206	.612	MJ306	.610

The row correlation data are similar in both collections, but the coefficients are somewhat smaller in the open collection. This is, of course, not true for the documents in report NSF-6, the most recent report in the collection, where the closed collection row correlations are entirely zero because of the total lack of citation data. The open collection does not remedy this defect entirely, however; newer documents still have fewer than the average number of citations, and some articles (such as the sole article on analysis of Japanese) have no citations at all. Only one document in report NSF-6 has a cross correlation with TDCMP which is as large as the over-all cross correlation. The lack of data for the last report can be attributed to the limited circulation of the reports, preventing most non-Harvard writers from citing them, and to the existence of only one additional report in the NSF series, whose authors do have access to the documents in the collection. Document collections taken from more accessible publications and printed earlier than the collection studied would probably not show a lack of data for the more recent documents.

The extent of the similarity between the open and closed collections in the CITED matrix may be demonstrated by considering the detailed row correlations of a typical document, C0208. The row correlation vectors for document C0208 are shown in Table 1. The cross correlation for this document in the closed collection was .633; in the open collection, .527. Almost all the row correlations are lower in the open collection, although the arithmetic difference between the correlations in the open and closed collections varies from document to document. Several articles, in fact, do not follow the normal trend, having higher row correlations with document C0208 in the open collection than in the closed collection. These are documents CH504, GI209, GI491, J0201, J0310, MJ494, SH206, and GI495, which are indicated in Table 1 by an asterisk. CH504, whose row correlation rose from .0000 (closed collection) to .3333 (open collection), is unimportant; the rise is due to a single citation that appeared in the open collection. The content of CH504 is completely unlike that of any other article, and presumably a sophisticated program evaluating citation data would recognize that the high-row correlation was caused by only one citation, and would ignore CH504 in studying the content of C0208.

The other articles that exhibited larger row correlations in the open collection were furnished with a substantial number of citations. It may therefore be of interest to study the relation of their content to that of C0208. C0208 deals with dictionary correction and updating;

Document	Closed	Open
BO319	.3162	.2981
CH504	.0000	.3333*
CO305	.3536	.3333
CO403	.2500	.1667
FO415	.4082	.3849
FO496	.3536	.2722
FO506	.4082	.4082
FR205	.2500	.2222
FR303	.4082	.3333
FR304	.2041	.1667
FR309	.1768	.1667
GE302	.4082	.3849
GI209	.3162	.3536*
GI491	.1443	.1849*
GI495	.2041	.2357*
GR313	.1443	.1179
IS311	.3536	.3333
IS312	.2041	.1667
J0201	.3430	.3727*
J0310	.1443	.1491*
LY414	.2041	.1925
LY602	.2500	.2357
MG404	.2500	.1361
MG493	.2500	.1721
MJ203	.3536	.2520
MJ306	.3162	.2222
MJ318	.2041	.1361
MJ494	.2500	.2520*
MJ505	.3062	.2910
RO207	.2887	.1491
SH206	.2942	.3234*
VC307	.3536	.3333

Row Correlations of C0208
(All row correlations not listed are .0000)

TABLE 1

thus, GI209 (postediting, translation algorithms), J0310 (recognition of phrases within sentences), and SH206 (word-by-word syntactic analysis) are articles that we should expect to have low-row correlations. J0201 (continuous dictionary run) should also exhibit a low-row correlation, while GI491, GI495, and MJ494 (dictionary compilation) might be considere partial successes for the open collection, since they exhibited larger row correlations in that collection. Unfortunately, such directly releva articles as FO415 had lower row correlations in the open collection than in the closed collection. Thus, as far as CO208 is concerned, the closed citation data give slightly better results, as indicated by the respectiv cross correlations.

Greater differences between open and closed collection data are shown by matrix CITNG. Here the total number of citations more than doubled (from 165 to 393) and the number of documents went up correspondingly (from 62 to 175) in going from the closed collection to the open collection. The over-all cross correlation fell from .366 (closed collection) to .239 (open collection). There is no regular decrease in either the row or cross correlations, which increase in the open collection for some documents and decrease for others. The five highest cross correlations are as follows:

CITNG Closed		CITNG Open	
Document	Cross Correlation	Document	Cross Correlation
CO317	.629	BO319	.614
JO413	.629	IS411	.601
VS308	.623	JO609	.526
JO609	.620	JO314	.411
IS411	.606	FO603	.408

Only two documents are common to both lists, and the cross correlations of the open collection are distinctly lower. The row correlation also change more than in CITED. BO408 has a .080 drop in cross correlations from the closed to open collections (.444 to .364). Its row correlation vectors are reproduced in Table 2. No uniform pattern can be found. The open collection does not give the impression, as it did for CITED, of being a scaled-down version of the closed collection. In the values of its coefficients, the open collection has a definite advantage in the low-row correlations of GR202 and GR204 (.2887 open, 1.0000 closed),

Document	Closed	Open
B0319	.5000	.2041
B0407	.7071	.5000
C0317	.3015	.1491
C0403	.0000	.2801
F0603	.0000	.1925
FR205	.5774	.1925
FR309	.5000	.1925
GB604	.0000	.4364
GR202	1.0000	.2887
GR204	1.0000	.2887
IS411	.0000	.1925
J0310	.4082	.1925
J0314	.5000	.2582
J0316	.5000	.2887
J0413	.4472	.4364
J0607	.5000	.4714
J0609	.0000	.1179
J0611	.5774	.2582
KU605	.0000	.3482
PI601	.0000	.2041
SA401	.4472	.1925
LY602	.0000	.1667
VS308	.2887	.1325
WA412	.0000	.2357

Row Correlations for B0408 CITNG
(All correlations not listed are .0000)

TABLE 2

which are totally unrelated in content to B0408. The higher-row correlation of GB604 in the open collection is also desirable, as both GB604 and B0408 deal with the same topic (predictive syntactic analysis). A distinct disadvantage of the data from the open collection is the introduction of many small row correlations with B0408 for documents which previously had a zero correlation, and are not closely related to B0408 (e.g., LY602, C0403). Such correlations lower the cross correlation because they do not correspond to high-row correlations in TDCMP, and also interfere with attempts to derive index terms because of their lack of relation to the content of the articles.

The open CITNG matrix was also used for attempts to assign index terms to documents from the citation data. The results were not, unfortunately, better than those provided by the closed matrix. The introduction of a more sophisticated index term assigning procedure also failed to raise accuracy above fifty percent. It should be noted, however, that the cross correlations are lower in the open collection than in the closed collection. Results of the attempt to assign index terms are shown in Table 3. The new method for deriving index terms is:

(1) Select the five highest row correlations for any given document. To each of the related documents producing these high row correlations, assign a weight equal to the row correlation with the given document divided by the number of index terms associated with the related document. For example, a document with a row correlation of .7071 and with five assigned index terms would be weighted .1414 = .7071/5.

(2) For each index term, compute the sum of the weights of each document of the above five that it is associated with in TDCMP, and associate this weight with the term.

(1) Document	(2) Index Terms Assigned Automatically from CITNG	(3) Number of Terms in (2) Correct (also assigned manually)	(4) Total Number of Terms Assigned Manually	(5) Cross Correlation
FR304	5	1	4	.3426
J0607	3	1	2	.2101
F0409	4	3	5	.3006
C0208	6	3	6	.3711
J0413	3	3	3	.2777
TOTAL	21	11	20	

Index Terms from CITNG Open

TABLE 3

(3) Select those index terms which have a weight greater than the weight of any single document. For example, suppose documents A, B, C, D, and E had weights (respectively) of .1, .15, .15, .08, and .2. A term assigned to documents A and D would not be selected (.1 + .08 = .18 is less than .2), while a term assigned to documents A and C would be selected (.15 + .1 = .25 is greater than .2).

The matrix CNG 2 has far more citations and far more documents than either CITNG or CITED. Cross correlations are much lower in the open collection than in the closed collection; the second highest cross correlation is only .201 in the open collection as opposed to .747 in the closed collection. The over-all cross correlation drops from .482 to .116. The changes in the row correlation vectors are not consistent with respect to content, but many new low-row correlations are introduced, making the study of the correlations more difficult and adversely affecting the determination of index terms. Irrelevancies may be introduced; the open CNG 2 collection contains citations to Mark Twain's "The Jumping Frog of Calveras County" and Lewis Carroll's essay on assigning prizes in tennis tournaments.

In summary, then, the open collection of documents is inferior to the closed collection in terms of content-citation relationships. This is probably due to the greater number of extraneous citations which obscure the relevant data. This effect is strongly evident in CNG 2. In other matrices, the inferiority of the open collection is not as strongly marked, and it is possible that it might be of more use in analyzing citation data. As was pointed out under CITNG, there is some improvement in the relation of the high row correlations to content, while the small row correlations are less indicative of content than they were in the closed collection. Thus a scheme of analysis concentrating on the large correlations might give better results using the open collection, but this seems unlikely. Work done to date confirms the impression of the cross correlations, that is, the superiority of the closed collection. A conclusion as to the relative merits of CITED and CITNG cannot be drawn from these data, because the small circulation of the document collection has prevented adequate citation of its members. The lack of data for the more recent documents in the matrix CITED prevents general comparisons with matrix CITNG, which has no corresponding restriction.

REFERENCES

1. Salton, G., "The Use of Citations as an Aid to Automatic Content Analysis," Information Storage and Retrieval, Report ISR-2, Section III, The Computation Laboratory of Harvard University (September 1962).

2. Ibid., p. 30.

VI. ATTEMPTS TO CLUSTER DOCUMENTS WITH CITATION DATA

Michael Lesk

Since citation data alone are probably not capable of producing detailed, accurate, and reliable index terms for documents, an attempt was made to determine whether citation data could be used for placing documents within general groupings, each group containing documents of similar content. The need to place individual documents into subgroups of a large collection may arise in two distinct situations:

(1) An already subdivided collection exists, and it is desired to place newly acquired documents into their proper groups;

(2) Groupings are to be produced from a collection which has not been previously divided.

Problem (1) will be considered first. Here we may assume, for any given document, that the groupings for all other documents are known. The citation data used to study this first problem consisted of the row correlation matrix CITED.[1,2] A manual division of the collection into nine groups, of three to nine documents each, was made. These groupings were made solely on the basis of content. Seventeen documents which did not fit into groups were omitted. The grouping is given in Appendix A.

An attempt was then made to place each document into its correct group, using the citation data for that document and the known grouping of all other documents. This was done by considering the three highest row correlations of the given document in CITED, and the related documents associated with these numbers. If two of these related documents belonged to the same group, the given document was assigned to that group. If the three related documents belonged to three different groups, the group containing the document with the highest row correlation was chosen as the cluster to which the given document was to be assigned. This procedure was applied to the forty-one documents which were previously grouped by hand and which had citations in CITED (open). Of these, the correct group was selected in all but eight cases. Two of these eight cases involved articles in Report NSF-6, for which CITED does not give sufficient data. Two more errors involved a single article which was cited more than any other article and may have suffered from irrelevant citations. The elimination of these three articles with either too few or too many citations leaves four errors in thirty-seven documents, or a success rate of 89.2%.

A direct division of the collection into groups was also studied, where only citation data were used to derive the clustering. The method employed was suggested by Mr. Edward Sussenguth. We may consider any two documents as linked by a strength proportional to the size of the row correlation between the two documents in a citation matrix. Let us define as major links those with row correlations exceeding 0.7. If a document has at least one citation link with a coefficient exceeding 0.7, we may define a group containing it as the set of all documents joined to the given document through a chain of links all greater than 0.7 in strength. This produces several clusters in a typical citation matrix, but leaves documents without links of 0.7 or greater ungrouped. These documents may be grouped by assigning them to that cluster which contains a document that is linked to the given document by a strength of 0.6 or greater, if any such cluster exists. If the given document has no links of that strength (including all row correlations more than 0.6), it may be viewed as an extraneous document, or links of strength 0.5 can be studied. Clearly, the constants 0.6 and 0.7 in this method are arbitrary and can be raised or lowered as demanded by the individual data.

The citation matrices CITED (open) and CNG 2 (closed) were clustered by this procedure. TDCMP was also clustered in the same way to compare it with the hand division made earlier. The rough results are shown in Appendices B, C, and D. Generally, the groupings do not accurately reproduce the hand division. TDCMP comes closest, as would be exptected. CNG 2 is the worst, despite the fact that the cross correlation of CNG 2 with TDCMP is higher than that of CITED with TDCMP. Roughly, of the nine groups in the hand division, six are retained more or less intact in the TDCMP clustering, about three in CITED, and one in CNG 2. For example, the group of documents on postediting and the "trial translator," which consists of GI2C9, MA301, and J0314, is retained in the TDCMP clustering (with the addition of GB604, which does not belong in this group), is still retained in the CITED clustering, but is split over three groups in the CNG 2 clustering. The continuous dictionary run articles, J0201, J0316, J0413, and J0607, are still clustered in the TDCMP grouping, but are split up in the CITED and CNG 2 groupings. Other rearrangements are also apparent from the clusterings given in the Appendices.

More sophisticated methods of clustering the collection from the citation data have been proposed, but it is doubtful that really substantial improvement is to be expected. For example, the links between BO408, GR202, and GR204 in CNG 2 are all of maximum strength (all row correlations 1.0000), and yet the documents should not be in the same group. It would seem that some form of additional information will be needed to succeed.

APPENDIX A

MANUAL CLUSTERING

(1) GI209 - MA301 - JO314 (postediting)

(2) JO201 - JO316 - JO413 - JO607 (continuous dictionary run)

(3) BO407 - BO408 - GB604 (predictive syntactic analysis)

(4) FO603 - WA410 - WA412 - FO409 - GR313 - IS411 (English inflection)

(5) FR304 - FR205 - VS308 (editing programs for syntactic study)

(6) GI491 - GI495 - MJ505 - CO403 - MJ306 - MJ318 - PL315 - MG404 - MJ203 (dictionary compilation)

(7) CO208 - FO415 - IS312 - JO609 - LE606 (dictionary correction)

(8) SH206 - FR309 - GE302 - FR406 - FR612 - FR405 - CO317 - FR303 (Russian syntactic analysis)

(9) MG493 - FO496 - FO506 - MJ494 (Russian inflection)

APPENDIX B

TDCMP CLUSTERING

(1) GI209 - MA301 - JO314 - GB604

(2) JO201 - JO316 - JO413 - JO607

(3) BO407 - BO408

(4) WA412 - WA410 - FO409 - IS411 - GR313 - FO496 - FO506 - SH206 - JO310

(5) CO208 - FO415 - IS312 - FR304 - FR205 - VS308 - LE606 - JO609 - GI495

(6) MG404 - CO403 - PL315 - MJ505 - MJ494

(7) MJ203 - MJ318 - MJ306 - FR303

(8) CO317 - FR405 - FR612 - FR406 - VC307

(9) RO207 - CO402

APPENDIX C

CITED CLUSTERING

(1) GI209 - MA301 - JO314

(2) BO407 - BO408 - JO607

(3) WA410 - IS411 - FO409 - JO316 - JO413 - FR612 - JO611 - FR405 - FR406

(4) CO403 - MJ203 - MJ318 - MG404 - MJ306 - VS308 - IS312 - FR304 - FR205

(5) FO496 - FO506 - MJ494 - MG493 - GI491 - GI495

(6) LY602 - LY414 - IS311

APPENDIX D

CNG 2 CLUSTERING

(1) MA301 - GB604

(2) J0314 - J0316 - B0319 - GE302 - FR205 - SA401

(3) GI209 - C0208 - SH206 - J0201 - F0496 - MJ505 - MJ203

(4) B0407 - B0408 - GR202 - GR204

(5) FR304 - IS311 - F0415

(6) FR405 - FR309 - FR612 - C0317 - FR303 - PL315 - C0403 - VC307 - VS308 - J0413 - C0305 - J0310

(7) LY602 - J0609

REFERENCES

1. Salton, G., "The Use of Citations as an Aid to Automatic Content Analysis," Information Storage and Retrieval, Report ISR-2, Section III, The Computation Laboratory of Harvard University (September 1962).

2. Lesk, M. E., "Comparison of Citation Data for Open and Closed Document Collections," Information Storage and Retrieval, Report ISR-3, Section IV, The Computation Laboratory of Harvard University (January 1963).

The Computation Laboratory of Harvard University, Cambridge, Massachusetts, INFORMATION STORAGE AND RETRIEVAL, SCIENTIFIC REPORT ISR-3, Gerard Salton, Project Director, April 1963, 144 pp. AFCRL-63-134 Unclassified report. This report contains a record of the continuing investigation of various techniques for automatic content analysis, and for the storage and search of structured information. In particular, some storage allocation techniques are examined which are useful in the manipulation of natural language data; a variety of machine programs and experiments are then described for the processing of bibliographic citations; and two models for a completely automatic document retrieval system are outlined.	UNCLASSIFIED 1. Information Retrieval I. Salton, G. UNCLASSIFIED

Defense Documentation Center Catalog Card

The Computation Laboratory of Harvard University, Cambridge, Massachusetts, INFORMATION STORAGE AND RETRIEVAL, SCIENTIFIC REPORT ISR-3, Gerard Salton, Project Director, April 1963, 144 pp. AFCRL-63-134 Unclassified report. This report contains a record of the continuing investigation of various techniques for automatic content analysis, and for the storage and search of structured information. In particular, some storage allocation techniques are examined which are useful in the manipulation of natural language data; a variety of machine programs and experiments are then described for the processing of bibliographic citations; and two models for a completely automatic document retrieval system are outlined.	UNCLASSIFIED 1. Information Retrieval I. Salton, G. UNCLASSIFIED

Defense Documentation Center Catalog Card

The Computation Laboratory of Harvard University, Cambridge, Massachusetts, INFORMATION STORAGE AND RETRIEVAL, SCIENTIFIC REPORT ISR-3, Gerard Salton, Project Director, April 1963, 144 pp. AFCRL-63-134 Unclassified report. This report contains a record of the continuing investigation of various techniques for automatic content analysis, and for the storage and search of structured information. In particular, some storage allocation techniques are examined which are useful in the manipulation of natural language data; a variety of machine programs and experiments are then described for the processing of bibliographic citations; and two models for a completely automatic document retrieval system are outlined.	UNCLASSIFIED 1. Information Retrieval I. Salton, G. UNCLASSIFIED

Defense Documentation Center Catalog Card

The Computation Laboratory of Harvard University, Cambridge, Massachusetts, INFORMATION STORAGE AND RETRIEVAL, SCIENTIFIC REPORT ISR-3, Gerard Salton, Project Director, April 1963, 144 pp. AFCRL-63-134 Unclassified report.	UNCLASSIFIED 1. Information Retrieval I. Salton, G.
This report contains a record of the continuing investigation of various techniques for automatic content analysis, and for the storage and search of structured information. In particular, some storage allocation techniques are examined which are useful in the manipulation of natural language data; a variety of machine programs and experiments are then described for the processing of bibliographic citations; and two models for a completely automatic document retrieval system are outlined.	UNCLASSIFIED

Defense Documentation Center Catalog Card