To the Graduate Council:

I am submitting herewith a thesis written by George R. Patterson III entitled "Results of Experimentation with Automated Library Reference Systems." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Kevin C. O'Kane, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

The Graduate School

RESULTS OF EXPERIMENTATION WITH AUTOMATED LIBRARY REFERENCE SYSTEMS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

George R. Patterson III

June 1984

ACKNOWLEDGEMENTS

I wish to thank my advisor, Dr. Kevin C. O'Kane, for his assistance in this research and in adapting this project to the Vax-11, and I would like to thank the other members of my committee, Dr. David Straight and Dr. Robert Heller.

Dr. Straight provided help with PL/I, the language of the source code, Dr. Charles Pfleeger answered many questions pertaining to the efficiency of algorithms, and William Guinn was quite helpful in explaining the intricacies of the VAX.

In addition, I wish to thank Thomas Richmond and Virginia K. Patterson of UTCC for their assistance.

ABSTRACT

This paper presents results of empirical experiments with an automated library reference system similar to the SMART system. Stop list generation, stem reduction techniques, discrimination functions, synonym handling, and retrieval weighting techniques are discussed in terms of their effects upon query precision and recall. Other techniques are discussed in terms of theory.

The general outline of an alternative system for small computers is also presented, along with performance results.

TABLE OF CONTENTS

CHAPTER

I. STATEMENT OF THE PROBLEM 1

Introduction 1

Overview 3

Basic Systems used 5

II. LIST GENERATION 7

Input Collection Handling 7

The Stop List 8

Stem Production 9

Experimental Results 10

III. STEM LIST REDUCTION 16

Discrimination Value Calculation 16

Weighting of Terms 24

IV. THE SYNONYM THESAURUS 30

Concept 30

Thesaurus Trials 33

Experiments 35

Results 38

V. RETRIEVAL REFINEMENTS 41

Weighted Retrieval 41

An Alternative Method 43

Comparative Results 44

CHAPTER

VI. CONCLUSIONS 46

Evaluation Methods 46

Conclusions 47

BIBLIOGRAPHY 49

APPENDIX 51

VITA 64

LIST OF TABLES

TABLE

I. Stop List Generation Thresholds 56

II. Discrimination Function Results 58

III. Retrieval Weight Results 59

IV. 97 Documents Retrieved 60

V. Thesaurus Method Results 62

VI. Weighted Retrieval With No Thesaurus 63

VII. Weighted Versus Unweighted Retrieval 63

CHAPTER I

STATEMENT OF THE PROBLEM

Introduction

The objective of the work on which this thesis is based is to study techniques commonly used in the design of automated library reference systems. These systems are designed to perform some of the functions of a card catalogue by allowing the user to enter a question in conversational form through a terminal and receive a list of references which deal with the topic of the question.

In a card catalogue system, a great deal of labor by highly trained personnel has been spent in determining the topics with which each text deals, creating cards, and in organizing and maintaining the catalogue. The degree of success obtained during a catalogue search depends to a great extent upon the catalogue user's skill, training, and vocabulary and may take a considerable length of time. The advent of large computers, such as the IBM 7040 series and 360 series machines, made it possible to design automated systems utilizing data sets instead of cards which reduce the amount of time required both for training users and library personnel and for the creation and use of a catalogue.

One approach to this idea is the MEDLARS system, which is an index of articles from medical journals. The design of this system was begun in 1964 and it became generally available in 1971 [1]. The indexing techniques used are completely manual [2], consequently, no advantage is gained over a card catalogue in the area of catalogue creation or

maintenance, and there is a three- to six-month lag between the publication of an article and its appearance in the data set [3]. Topics determined by the indexing personnel are, however, stored in data sets, and the system is "online," making it readily available regardless of the location of the user. Both a Boolean topical search (which requires some degree of user training to handle well) and a program which allows conversational queries are available. Its advantages over card catalogues lie in search speed and availability.

An approach to simplification of the process of catalogue (data set) creation is exemplified by the SIRE and SMART retrieval systems. These operate on the premise that it is possible to determine the subject matter of documents through mathematics rather than reason and consequently possible to generate the topics used in document searches automatically. Some highly skilled labor is required for production of the text synopses or "abstracts" normally used as input, but these may then be entered by any good typist.

The SIRE system uses comparisons of word frequencies in determining the topics of documents, and queries are evaluated in a Boolean fashion. The designers of SMART did not limit themselves to manipulations of word frequencies, and queries in conversational form are used. Both systems produce substantial savings in the number of man-hours necessary for data set creation. Both are the results of research performed at educational institutions and are intended to provide the basis for other systems which should be "custom tailored" to the input documents [2].

For this thesis, a system similar to the SMART system (described more fully in the Overview) was implemented in PL/I. Optimal performance with this system was achieved by varying parameters such as the method of calculating retrieval weights. In addition, an alternative system was implemented which did not use the weighted retrieval or synonym thesaurus mentioned below, and the performance of this alternative system was compared to that of the final version of the original system.

Overview

The SMART system is one of the better known experimental automated library reference systems, having been in operation since 1964 running on IBM-7094 and IBM-370 equipment [4]. It was developed at Harvard University, and a great deal of work has been done, both at Harvard and at Cornell University, to optimize the results. A number of techniques attempted at these institutions proved to be quite expensive in terms of processor time or of little value [2] and are not mentioned here.

A system such as the SMART system consists of at least two programs having four main stages:

1. Input Parsing and "Stop List" Generation: The document abstracts are broken into their component words and a frequency count is obtained for each word over the entire collection. High frequency words, as well as those which are unique, are removed from further consideration by inclusion in a stop list, and the remainder are reduced to

their stems by the removal of commonly occurring suffixes. These steps are detailed in Chapter Two.

2. "Stem List" and Retrieval Weight Production: The frequency of each term stem in each document is obtained and compared to that stem's frequency over the entire collection. During this process, the stop list is checked for each word, and no frequency counts or comparisons are made for words which are in the stop list. Since those terms which have a high relative frequency in a few documents are considered to be the better key words, a list is made of these words. This process is known as calculation of a term's "discrimination value" and is covered in Chapter Three. In addition, the relationship between each term's frequency in each document and its total frequency can later serve as the term's relative importance as a key word during the retrieval process. This "weight" is usually calculated during this step, and this is covered in Chapter Four.
3. Building a "Synonym Thesaurus": Some of the terms which were not proven to be good key words during the preceding step may prove to be valuable as indirect references. In theory, if one word frequently occurs in conjunction with another, it probably references the same topic, or it may usually occur as part of a phrase which is significant. In this stage, a table is built of the co-occurences of the terms. Each item in this table is normalized according to the frequencies of the terms involved, since co-occurences

of high frequency words are less significant than those of low frequency terms. Those terms which display a poor correlation with others, and which are not found to be especially concentrated during step two may now be discarded. Files can now be created, which contain a list of the survivors of these processes, lists of file record numbers for the titles of the documents in which they occur, and the term/document weights calculated in step two. This is discussed in Chapter Four.

4. The Retrieval Program or Query Handler: This program reads queries, reduces words in the queries to word stems, and searches the stem list produced in step two and three above for these stems. The title file record numbers and term/document weights are then obtained for any query stems which are located in the stem list. If the total weights of all query stems which reference an article exceed a given limit, the title of that article will be displayed to the user. This is discussed in Chapter Five.

Basic Systems Used

Three trial systems were implemented for this thesis:

1. A weighted retrieval scheme with a synonym thesaurus. This was designed to run on an IBM/370-3031 and memory paging schemes were necessary to keep memory requirements below 4096 kilobytes. Consequently execution times were excessive, requiring over one and one half hours central

processor time for 202 documents. For this reason, the decision was made to transfer the project to the Computer Science Department's VAX-11/780. This program was used primarily as a prototype for the next system and for the experiments on stem production techniques dealt with in Chapter Two.

2. A VAX based system basically similar to the first but with memory paging handled by the operating system rather than by the program. This version required less than forty-five minutes for execution, and the amount of experimentation performed would not have been possible had the VAX not been available. The majority of the results presented were obtained through modifications of this program.
3. A simpler IBM based system which used no term/document weights or synonym thesaurus. No memory paging was required, and execution times of under five minutes were obtained. The retrieval scheme was based on retrieval of only those documents which contained at least two key words found in the query. Duplication of key words in queries was counted as more than one key word, and retrievals were made on the basis of only one key word in cases where only one was found in the query.

CHAPTER II

LIST GENERATION

Input Collection Handling

The first stage of the test program was similar to that of SMART in that the input text was broken into words and a "Stop list" built. During this stage the main points of consideration were; 1. How much of each document to include, and 2. what frequency count to use as a threshold in generating the stop list.

Since the collection of documents available consisted of the titles and abstracts of articles found in issues of the "Communications of the Association for Computing Machinery," it was impossible in this case to run tests involving entire document texts; however other researchers have found inclusion of entire texts to be at best of limited value [1][2]. It is only necessary to include enough of each article to obtain a good idea as to its topic, and frequently article texts contain much extraneous information so that programs which use entire documents generally are prohibitively expensive and frequently produce poorer results than those which only use a portion of the text or an abstract.

In the work for this thesis, the decisions to be made concerned the abstract and title; no author's names were included as input. Since titles sometimes have little or nothing to do with the subject matter, one run was made using only the words found in the abstracts. Since most authors provide a short description of their subject in the first few sentences, a run was also made using only the titles and the

first two sentences of each abstract. In this collection, the average size of an article and title was eighty-eight words.

The Stop List

The purpose of a stop list is to reduce program costs by eliminating those words whose presence is necessary for comprehension, but which add nothing to the context and are useless as key words. For example, the ten words "the," "of," "and," "to," "a," "in," "that," "it," "is," and "I" account for more than 25 percent of any typical English language text, and it would be a waste of time to attempt to reduce these to stems and obtain discrimination values for them. In fact, about 75 percent of any English text could be eliminated as "unnecessary" [5]. In addition, a word such as "computer," which would be a good key word in a general collection of documents, will probably occur in nearly every document in a collection of articles dealing with Computer Science and should be eliminated [1].

The undesirable words, then, are merely those which occur most often in the collection or are uniformly distributed across it. The latter are dealt with through techniques discussed in Chapter Three. To eliminate the former, one may determine the total frequency of each word and include in the stop list any terms with frequencies higher than a given limit. Many experts also recommend the inclusion of terms which are unique in the collection, on the theory that they will rarely occur in queries [1].

The next point to consider is the word frequency count above which a word becomes a candidate for the stop list. If the frequency break

point is set too high, the purpose of the stop list is defeated; too low, and good discriminators are eliminated. The latter becomes extremely critical when a synonym thesaurus is not used. A constant limit cannot be used; a word which occurs fifty times in a collection of twenty documents is obviously a poor key word, while one which occurs fifty times in a collection of two thousand documents is not necessarily poor. Using some percentage of the number of documents as a frequency break point seemed a logical choice, and a number of runs were made varying break points. In dealing with collections in which a representative document is significantly larger or smaller than that used here, a percentage of the total number of words in the collection would be a better choice, and both percentages are given for the results listed in Table I, Appendix.

Stem Production

Once the stop list has been created, the remaining words are stripped of the common English suffixes in order to transform words with similar meanings (such as Psychologist and Psychology) into identical word stems. A list is made of these word stems, and the final stem list will be chosen from this list through calculation of discrimination and synonym thesaurus values. Decisions must be made at this point as to what sections of a word to remove during word stem production, and whether or not to discard words which reduce to the same stems as stop list words. A number of successful algorithms exist for suffix removal [1], most of which are custom tailored to the subject matter of the documents. For example, that used in MEDLARS

bears little resemblance to the algorithm used by the test program. The one used here causes the removal of the list of suffixes found on page 73 of Salton/McGill [1] and may be found in the appendixes. Removal of prefixes is also a possibility which was explored. During all experiments, one and two character words were discarded rather than considered as possible key words or entered in the stop list. This was done on the assumption that searching the stop list for these words would require as much time as checking every word's length. The amount of additional storage required for their inclusion in the stop list was also a factor in this decision.

Experimental Results

Document Handling. It was found best to use the entire title and body of the abstract as input. The savings in time produced by the omission of titles was found to be immaterial (.03 seconds for 202 documents), and one sample question (number eight on page 55) failed to retrieve two of fourteen titles which were retrieved when titles were included. When only the document titles and the first two sentences of each abstract were included, the same question retrieved twenty-seven titles, fourteen of which were relevant, a decrease of nearly 50 percent in precision (the percentage of the set of documents located which are pertinent to the question asked). These runs were made with an unweighted retrieval scheme and with no synonym thesaurus, and it is possible that titles could have been omitted had more sophisticated techniques been used; however, it

was felt that the low cost of including titles made further experimentation along these lines unnecessary.

Stop List Generation. When a synonym thesaurus was not used, the optimal frequency threshold, above which a word would be placed in the stop list, was determined to be 25 percent of the number of documents. In the test collection, this meant that any word which occurred more than fifty times in the collection was placed in the stop list. Precision suffered markedly when this was increased to 33 percent, while recall (the number of pertinent documents located versus the number of pertinent documents in the collection) was noticeably reduced at a threshold of 20 percent. More leeway was encountered when a synonym thesaurus was introduced. In this case, both recall and precision steadily improved from the use of the number of documents (202) as a threshold to the use of 25 percent of the number of documents (in the case of recall this is the opposite of that found with the unweighted system). Further threshold reductions led to a decrease in recall and precision (Table I). Since optimal results were achieved in both systems at 25 percent, this threshold was used for stop list generation in further experiments. The test program produced a stop list of 1,517 words from an input collection of 17,840 words.

Stem Production. In reducing a word to its stem, it was found advisable to remove only certain suffixes, retaining a minimum stem length. Best results were achieved when steps were taken to ensure that no stems were retained which matched those of high frequency stop list words.

Removal of prefixes was found to be damaging to recall. For example, the words "encode" and "decode" were eliminated from the stem list when either was reduced to the stem "code," because they then matched a high frequency stop list word. This, in turn, had adverse effects upon "encryption" and "encypher" when the synonym thesaurus was used. It would also be rather irritating for a request about "NONrecursive languages" to produce documents dealing with PASCAL or PL/I.

It is advisable to use a great deal of discretion in this process. For example, a minimum stem length was found to be necessary for word recognition. When no minimum was established by the code, some good key words simply disappeared and others became ambiguous; "tree," for example, was reduced to "tr," which also happened to be the stem of "trial," "trivial," and a number of others. These words produced "tre," "tri," and "triv" when the restriction was placed on the algorithm that no suffix might be removed if doing so would cause the stem to fall below a length of three characters.

In choosing suffixes to be removed, it was found that a good synonym thesaurus was capable of correcting some problems which one would think would be caused by failing to obtain an exact match with words of similar meaning; a thesaurus would detect the fact that "discriminate" and "discrepate" are related, for example, because they would tend to occur frequently in the same documents. Overzealous stem reduction led to ambiguity between terms such as "discredit" and "discriminate" which could not be corrected later, producing very poor precision. The reduction algorithm had also to be adjusted to suit the

discrimination function and other techniques used. For example, when the suffix "in" was removed, recall was 5 percent greater and precision 6 percent greater for discrimination function number six; however, the removal of this suffix caused a 3 percent drop in recall and a 13 percent drop in precision when function number three was used. Generally, visual inspection of the stem list was found to be most useful in detecting cases of underreduction, while evaluation of retrieval results was necessary to detect the ambiguity produced by overreduction.

It was apparent from the early retrieval results that relatives of high frequency words were being retained as key words, damaging precision. For example, "computer" might be placed in the stop list, while "computation" would occur infrequently enough to avoid this. This would result in the stem for "computer" being retained, and a retrieval query searching for articles on "Medical Computing" would obtain many more extraneous titles than if that stem were not there. One attempt to solve this problem was to make the stop list available to the retrieval routine; however, this was of no help when the word retained (and in the query) was virtually identical in meaning to a stop list word. In those cases, poor discriminators were retained as key words simply because the discrimination function obtained artificially low frequency counts for them. Adjusting the frequency counts of these stems (eg. adding the frequency of occurrence of "computer" to that of the stem of "computation") produced a significant improvement in results. However, one document became somewhat "generic," showing up as an answer to nearly any question, and it

became apparent that stems of stop list words should not be allowed in the stem list.

An attempt to perform stem reduction first, and then obtain frequency counts and build the stop list from word stems resulted in a marked increase in precision, unfortunately proving quite expensive as well. The solution seemed to be to make stems of the stop list, and remove these from the stem list before going further. This was tried, however, some of the words in the stop list which were unique in the collection also turned out to have relations which were excellent key words and were being eliminated by this process. It was necessary to retain the stems of unique words and delete only the stems of high frequency stop list words from the stem list.

This still left one small problem; it was discovered that certain unique words which reduced to stems which were valid key words had no matching stems in the document in which they occurred. The result was that these document titles were omitted from the title lists for those stems, resulting in failure to retrieve that document. In this collection, for example, "medicine" was a unique word while "medical" occurred frequently enough and specifically enough to be a valid retrieval key during one test run. Since "medical" was not used in the article in which "medicine" was found, that article was treated as if the stem "med" were not present and was not located in response to a query on medical computing. This should be less of a problem with larger collections; however, should recall be of paramount importance or the input collection small, unique words which reduce to the same stems as other words should be deleted from the stop list before going

further. Doing this resulted in a 33 percent increase in recall for one query during one of the test runs.

CHAPTER III

STEM LIST REDUCTION

Discrimination Value Calculation

At this point, the stem list consists of all but the most common word stems. It remains necessary to remove those which are evenly spaced over the document collection, since using a term which is spread throughout the entire collection as a retrieval key will result in retrieval of the entire collection, while terms which are concentrated in a few documents are likely to deal with a specific topic, and queries which contain those terms are also likely to concern that topic. Each stem's specificity may be determined by comparing its total frequency in the collection with its frequency in each document. The stop list comes into play here: nothing is done with any word which is found in the stop list. Words which are not in the stop list are reduced to stems, and, if they occur in the stem list, their frequencies in each document are obtained. The next point of experimentation involved the function used to compare each stem's frequencies for determination of the stem's specificity or "discrimination value."

The goal of this step is to calculate mathematically the value of each term as a retrieval key. Stems which have higher than average values are considered to be valid key words and are retained for the final stem list. Of the remaining stems, all those of above average frequency are discarded, and the lower frequency stems are retained as possible synonym thesaurus terms. Numerous functions have been

developed for discrimination value calculation, many of which attempt to measure each term's divergence from its frequency in a "centroid," that is, a theoretical document which would be representative of the collection [1]. The performance of seven different discrimination functions was evaluated, and recall and precision for the first six may be found in Table II.

$$1.\ DV(t) = \sum_{d=1}^{N} \frac{(F(t,d) - AF(t))*(F(t,d) - AF(t))}{AF(t)}$$

Where DV(t) is the value of the term as a key, d is the document, N is the total number of documents, and t is the term. F(t,d) is the frequency of that term in document d and AF(t) is the average frequency of the term in the collection.

This is related to the sample variance function used in statistics [6]. Since AF(t) is a fraction, the division operation produces an increase in DV(t) which is inversely related to the total frequency of the term, resulting in favoritism for concentrations of low frequency terms. This balances the fact that F(t,d) will be larger for high frequency words.

The function reduces to DV(t) = DV(t) + AF(t) when F(t,d) = 0, which makes some sense, since a frequency of zero is a divergence from AF(t), its value in the centroid. Unfortunately, this also makes DV(t) directly dependent upon AF(t), giving high frequency words preferential treatment.

$$2.\ DV(t) = \sum_{d=1}^{N} \frac{(F(t,d) - AF(t))*(F(t,d) - AF(t))}{AF(t)}$$

for all $F(t,d) > 0$

This is a variant of the first function, restricting F(t,d) to positive values, to produce a stem list which contained more low frequency terms.

$$3.\ DV(t) = \sum_{d=1}^{N} \frac{(F(t,d) - AF(t))*(F(t,d) - AF(t))}{AF(t)}$$

for all $F(t,d) > 0$; $- AF(t)$ when $F(t,d) = 0$

This is another variant of function number one which should produce values which are a more accurate measure of the divergence of a term from the centroid. If the term occurs more often than its average frequency in one document, its divergence is considered to be positive. Should the term not be present, the divergence from the centroid is negative.

$$4.\ DV(t) = \sum_{d=1}^{N} \frac{F(t,d)}{AF(t)}$$

This was an attempt to reduce run time by simplifying the arithmetic involved.

$$5.\ DV(t) = \sum_{d=1}^{N} \frac{F(t,d)}{TF(t)}$$

Where TF(t) is the total frequency of the term in the entire collection. This was another attempt at time reduction.

$$6.\ DV(t) = \text{Log}_2\ TF(t) - \sum_{d=1}^{N} \frac{F(t,d)}{TF(t)}\ \text{Log}_2 \frac{TF(t)}{F(t,d)}$$

This is known as a "Signal" minus "Noise" function [1] and operates on the basis that terms which are concentrated in a single document would produce a "Noise" value (the sum shown above) of 1 $\text{Log}_2(1)$, which is zero. In these cases DV(t) becomes directly proportional to the frequency of the term. Terms which are evenly spaced over the collection would produce values close to $\text{Log}_2(N) - (N * (1/N\ \text{Log}_2(N/1)))$, which will approach DV(t) = 0. This can be expected to produce relatively good precision at the expense. perhaps. of recall.

$$7.\ DV(t) = \sum_{d=1}^{N} F(t,d) * (\text{Log}(N) - \text{Log}(F(t,d)) + 1$$

This is a different approach to measuring a term's distance from its value in the centroid.

Methods of Experiment. Each of the above functions was used in programs which were identical in other aspects. The test questions were fed to the retrieval routine described in Chapter One after each change in discrimination function, and the results of each run were compared. Precision and recall figures for this step may be found in Table II, Appendix.

All results in this step were based on retrieval of 52 documents by each program. In testing different functions, the level of retrieval for each had to be the same, that is, if 50 documents were retrieved from a data set set up with one function and 150 were retrieved from another, no valid conclusions can be drawn about recall and precision differences. For this reason, it was necessary to repeatedly adjust the retrieval weight threshhold and make several runs for each function until one run retrieved exactly as many documents as the program being used as a control. The third function retrieved 52 documents using the average of all term/documents weights as a retrieval threshold, and this was arbitrarily chosen as the control.

In addition, a method was needed to allow precision and recall to be evaluated as a single value, otherwise, unless one function returned poorer values for both recall and precision than another did, it would be difficult to decide which was the better. The evaluation function utilized was similar to this one found in van Rijsbergen [7].

$$E = 1 - \frac{1}{K * (1 / \text{Precision}) + (1 - K) * (1 / \text{Recall})}$$

Where E is the value of a system's performance, and K is a constant. The original equation was based was on management theory, and is

intended to allow the user to bias the results. K would be assigned a value by the evaluator to assign a greater importance to recall or precision, whichever was felt to be more important. For this work, K was set to .5, assigning equal value to each. This equation will produce a value of zero when perfect precision and results are obtained, and one for total failure. To avoid confusion, this equation was inverted, and the equation actually used was

$$E = \frac{1}{.5 * (1 / \text{Precision}) + .5 * (1 / \text{Recall})}$$

Therefore, higher values of E indicate better results.

Three of the thirteen queries used to generate the retrieval tables reference only one document in the text. This was done to accentuate differences in recall produced by the functions used and to allow cases of excessive recall (and consequently poor precision) to be noticed easily. Since in large collections there would be few "one document queries," there existed a possibility that this would produce biased evaluations. To guard against this, it was decided to also calculate "revised averages," that is, the average results concerning only the ten questions which have multiple answers.

Results. Function number one produced 46 percent recall and 65 percent precision, comparing favorably with the 68 percent precision at this level of recall produced by the SMART system using an input collection of TIME magazine articles [1]. After examination of the resulting stem list and the results of using this function on various values with a calculator, however, it was decided that the low frequency words needed a little more help.

This function was then modified to produce function number two, which restricted F(t,d) to positive values. The results were encouraging; recall increased by 2 percent and precision increased 12 percent. As five out of thirteen queries were affected favorably, it appeared that the overall quality of the stem list was improved over that produced by function number one.

Function number three produced additional increases of 3 percent for recall and 1 percent for precision, even though only one query's results differed. It also has a minor advantage in that discrimination values remain smaller than those produce by the previous two functions, so that, in the case of large collections, one need not worry as much about floating point overflow when computing the average of all the discrimination values.

As an attempt to reduce run time, functon number four failed, being the most expensive of the seven, requiring more than forty-four minutes execution time on the VAX. Apparently, this function discarded fewer terms, requiring more work by the synonym thesaurus, as evidenced by the fact that the final stem list produced by this version was the largest so far. Recall improved 1 percent and precision 7 percent over function one. This was still poorer than function three.

The fifth function produced results which were identical to function number four, requiring nearly as much time.

When function number six was used both recall and precision were found to suffer overall in comparison to function three, although results for some isolated queries were improved.

The seventh function was attempted during early experiments and

abandoned when a visual inspection of the stem list produced showed that 292 stems present when function three was used were missing when this one was. A quick check of these words indicated that this function eliminated low frequency terms, an action shown by the data produced using the first three functions to be detrimental to performance. This function also caused 238 stems to be retained which had been eliminated when function three was used. Examination of the stem list verified a preference for high frequency terms, a high proportion of such words as "add," "also," and "but" being present. Recall and precision results were not obtained.

When all thirteen queries were considered, function three placed highest at 51 percent recall and 78 percent precision. Function one exhibited the lowest results, returning recall of 46 percent and precision of 65 percent. According to the revised figures, function three again scored highest, producing 36 percent recall and 71 percent precision. Once more, function one displayed the lowest recall and precision. Function number six was impressive in that it was the only function to cause retrieval of pertinent documents in answer to the last question. Its E value was second only to function three, and its execution time was more than two minutes less.

The excellent overall results of the third function led to its adoption during all other stages of this research, however, it must be emphasized that the small size of the input data set may be partially responsible for its success. In a larger system, the use of function six should also be considered.

The stem list produced by this stage contained 392 words which had

been determined to be significant retrieval keys by the discrimination value calculation process. In addition, there were 764 other stems retained for further handling in the thesaurus section covered in the next chapter.

Weighting of Terms

The principle behind Term/Document weights (the value of a term as a retrieval key for a given document) is quite simple. If a document deals with a given topic, that topic is likely to be referenced comparatively frequently in that document. Consequently, a term which occurs frequently in that document, compared to its frequency elsewhere, should be a good retrieval key word for the topic of the article. If that term occurs infrequently in another document, the second document is less likely to be about the same topic. Thus, each term can be assigned a "weight" for each document as an indicator of its value as a retrieval key based on its concentration in the document. A retrieval scheme which operates on the simple presence or abscence of a term in an article would retrieve a higher percentage of unrelated material than one which takes the relative concentrations of terms into account.

During retrieval, all weights for all key words in a query are summed for each document referenced, responses being made up of all documents whose weight sums exceed a given limit. This section is concerned with methods for determining concentration.

Experiments. As has been mentioned, the retrieval point was varied during the preceding section in the interests of result standardization, and that practice was continued here during the remaining experiments. Five weighting functions were tried, several of which performed with approximately the same degree of success and recall and precision figures for these are in Table III, Appendix.

1. $W(t,d) = F(t,d) / TF(t)$

   All results mentioned so far were generated using this function. This function was used because it is easy to calculate, and, since it has little relation to most of the various discrimination value functions, it was unlikely to interfere with their evaluation. It also seemed to be a matter of common sense that the concentration of a term would be reflected by the relationship between these two variables.

2. $W(t,d) = F(t,d) * (\text{Log } (N) - \text{Log } (TF(t)) + 1)$

   This operates on the same basic principle as the first; that is, the weight has a direct relationship to the term's frequency in each document and an inverse relationship to its total frequency [8]. The resulting weights covered a greater range than other functions tested, which made the number of documents retrieved much easier to control.

3. $W(t,d) = F(t,d) * DV(t)$

   This was used by a version of the SMART system [4]. In that system, it was used as a function for vector calculation,

but Salton recommends it as a potential weighting function as well [1]. Salton's DV(t) is based on calculation of "Document Similarity" versus "Average Similarity." The discrimination function used operates on the same principle, using term/document and average frequencies as indicators of similarity. The only objection which presents itself is that a discrimination function which did not perform well with a particular data set would compound the problem by affecting the weighting scheme as well.

4. $W(t,d) = F(t,d) / DF(t)$

   This was also originally designed as a vector function for a version of SMART; however. It applies the same general principle as the first function, using the number of documents in which a term was found instead of the total frequency.

5. $W(t) = (F(t,d) * TF(t)) / DF(t)$

   This was originally used in a version of the STAIRS system for term weighting. As the total frequency of the term is part of the dividend, this favors high frequency terms, which might be advantageous with large data sets.

Methods. All five functions were tested using the same procedures used in testing the discrimination functions. Since the results showed no clear-cut superiority when fifty-two documents were retrieved, an additional run was made for each of the top four functions retrieving ninety-seven documents. Comparison runs were then made between the two

best ones, functions one and four, utilizing different retrieval levels, and a graph of the results may be found in the appendixes, immediately after Table III. The retrieval test points of fifty-two, eighty, ninety-seven, one hundred twenty-one, and one hundred thirty-seven documents were determined by the fact that the first function produces weights over a very small range, creating a sensitive retrieval situation in that small changes in the retrieval threshold produce large changes in the number of documents retrieved. It was impossible, for example, to set the threshold to allow retrieval of exactly 100 documents. The range of weights produced by function four allowed more precise adjustments, and this fact made it possible to produce the same number of articles with both routines at the points mentioned.

Results. Weighting Functions: The first function produced 51 percent recall and 78 percent precision when fifty-two documents were retrieved. The weights resulting from its use covered a fairly narrow range, making control of the number of documents retrieved difficult.

The weights produced by using function number two covered a greater range than those obtained using any of the other functions tested, which made it much easier to control the number of documents retrieved; however, the fact that recall was 19 percent and precision 31 percent below that of function one ended further consideration. The poor performance demonstrated here is possibly due to the small size of the collection, since the size is a variable in the equation.

The third function produced rather poor results, returning 34

percent recall and 52 percent precision. The fifth performed even more poorly, producing recall of 33 percent and precision of 47 percent.

Function number four, however, provided results which were comparable to function one; recall and precision figures of 49 percent and 77 percent were obtained. In addition, a greater range of weights was produced than by function number one.

Other Considerations: The choice of the weighting function used also affects the relative ease with which the data sets used for title retrieval could be updated should the document collection expand at a later date. If function one were used, for example, additions to a library which dealt with documents which used a fairly static vocabulary would require only that the weights of all terms already in the stem list which also occur in the additional articles be recalculated, since the total frequencies of only those terms would be affected, and this frequency could be stored along with the weights for that term. This is also true of functions three, four, and five.

The inclusion of the number of documents as a variable in the second function would, however, make expansion of the document collection much more difficult to accomplish without re-running the entire program. Use of this function would require recalculation of every weight, since $\log_2(N)$ would change.

Conclusions: The results of the comparison runs between functions number one and four showed a superiority of function one at retrieval levels below approximately ninety documents, function four being more productive above this level. Both functions performed quite well over

the entire range however, the fourth one seemed to be the more stable, exhibiting a range of values for the E function from .5784 to .5989, compared to .5571 to .6167 for function number one. At the highest level of retrieval, figures for both recall and precision exceeded those for function one by 1 percent; On the other hand, examination of the recall and precision figures for individual questions showed that the fourth function was at a disadvantage when short queries were given, apparently requiring more key words. In addition, users generally prefer to keep the set of retrieved articles fairly small. These facts combine to make it less "user-friendly" and less suitable, perhaps, for use with a large data set.

Therefore, it was decided to continue with the first function, with one slight modification; in further experiments a multiplication factor was used during weight calculations in order to expand their range and make finer adjustments of the number of documents retrieved possible.

CHAPTER IV

THE SYNONYM THESAURUS

Concept

The purpose of the synonym thesaurus is to allow retrieval of documents which are relevant to a query in cases in which a "significant" key word, that is, a word which had a high discrimination value, is missing from either the document or the query. In theory, if two words are frequently found together, they probably have similar meanings and can be used nearly interchangeably as retrieval keys, and, for this work, a "synonym" is any non-significant term which is frequently found in the same documents with another term. There is some danger that, if they do have similar meanings, and one has been found to be a poor key word, that the other, which is thought to be significant, is actually as POOR a discriminator as the other. In theory, a weighted retrieval scheme should guard against this, and in fact, retaining synonyms of non-significant, as well as significant, terms produced the best results. Salton goes so far as to use only poor discriminators in constructing his thesaurus [1][2].

There are two basic reasons for using a synonym thesaurus, the first of which is to allow recall to be improved through use of synonyms as retrieval keys for those queries in which few or no significant key words are present. To accomplish this purpose, it is sufficient to build a title reference list for each synonym which contains only those record numbers of documents in which both the significant key and the synonym were found. This technique does not

allow retrieval of relevant documents in which no significant keys are found, but neither does it allow irrelevant documents to be retrieved based solely on the presence of a synonym.

The second reason for use is to allow retrieval of documents which lack significant keys, through use of synonyms for those keys. For this, one must include record numbers for all documents in which the synonym occurs. Both procedures were attempted during this research.

Another method of improving recall through thesaurus use is to regard the synonyms as keys to documents in which significant terms related to it occur, whether or not the synonym occurs in those documents. A synonym can thus serve as a link to documents dealing with the subject requested which do not contain matches for key words found in the query, provided the synonym occurs in the query. In systems such as SMART or MEDLARS this is done by grouping the terms in "clusters" or "classes." Synonyms occurring in the query would then be used to locate classes, and retrieval is based on the class rather than the word itself.

This method has several advantages. Recall at a given level of precision is generally improved by use of this system [1][2], and it is possible to allow the retrieval routine to enable the user to control the amount of precision and recall. Classes may be grouped in the thesaurus file in such a way that each class' neighbors are more or less related topics, the degree of relevance decreasing with distance. "Neighbors," in this case, is defined as nearness in a logical structure, and should not be taken to relate to actual distance between file records. Using this, the user may expand the degree of recall by

specifying that neighboring classes be included in the retrieval. In the SMART system the thesaurus is generated automatically and is completely separate from the dictionary of word stems already located through discriminated value calculation [4]. It may thus be used to improve recall, or ignored by the user.

Alternately, precision can be improved in a system by organizing the classes as a tree, with more specific classes at lower levels. A class can then be located by the initial query and the search for responses to further queries restricted to subtrees of this class. The word stems themselves would be the leaves of this tree, and at any level, documents located by all leaves of that subtree can be located. Obviously, this is hardly likely to be inexpensive in terms of disk storage or machine time. In addition, such thesauri are not easily generated automatically, and systems such as MEDLARS utilize manually constructed class trees [1].

For this work, methods similar to those used in SMART to detect the relationships between words, upon which the classes are based, were used instead to locate additional key words which were retained in the stem list rather than placed in a separate thesaurus. The complexity of the retrieval program was thus minimized, and it was felt that precision would not be damaged with this technique since the synonyms (being evenly distributed terms) would tend to have low weights. Because of this, synonyms would only become a factor in retrieval when they were found in conjunction with another key term in a query. During discussion of the test systems, "thesaurus" will designate the

portion of the final stem list which would not be present had the following techniques not been used.

Thesaurus Trials

The stem list at this point consisted of all keys retained by the discrimination function as significant, as well as all other stems of a lower than average frequency, flagged to indicate which. A matrix containing the term/document weights for all of these stems had also been built and was used to locate documents in which each term occurred.

To implement the thesaurus, a matrix was built of the co-occurrences of stems in documents. Each stem was designated a row and a column in such a manner that each element would contain the number of times the stem represented by the row occurred in the same document as the stem represented by the column. The diagonal would have the number of times each stem occurred in the same document with itself, which, of course, is the number of documents in which it appeared. This co-occurrence matrix was created by comparing one stem's column in the term/document matrix with that of another stem, noting the number of matches. This was repeated for all combinations of stems; a process requiring a M(N(N+1)/2) algorithm where M is the number of documents and N the number of stems. It was only necessary to generate the lower half of the matrix; however, this is still N(N+1)/2 elements.

The values in this matrix had to be be adjusted to reflect the relative frequencies of terms, as a high ratio of co-occurrence between

two low frequency terms should be of greater significance than a strong relationship between a low frequency term and a high frequency term or two high frequency terms. To accomplish this, a third matrix was created from the co-occurrence matrix. Each value in this "normalized" matrix consists of the total number of documents divided by the document frequency of the first term times that of the second, all multiplied by the number of times the two terms occur together;

Mat(i,j) = Concur(i,j) * (N / (Concur(i,i) * Concur(j,j)))

Where MAT(i,j) is the normalized matrix element for term i and term j, Concur(i,j) is the entry in the co-occurrence matrix for those terms, and N is the number of documents.

Since there is no need to compute this for those cases where i = j, this matrix is slightly smaller, containing N(N-1)/2 elements. This brings the total storage requirements to N^2 elements for these two matrices. Fortunately, the first row may be calculated as soon as the second row of the co-occurrence matrix is completed, the second as soon as the third is done, and so forth; requiring only an additional N(N-1)/2 algorithm. Once this matrix was completed, its values were used for augmentation of the stem list. A tree was constructed of all significant terms as well as any others which showed an above average affinity for any other term, that is, had higher than average values in the last matrix.

A file entry for each key was then created containing the file record numbers of the titles to all articles in which that term was found and the weight of the term in that article. All experimentation at this stage used the same basic thesaurus construction technique with

alterations in weighting methods for synonyms and the establishment of linkages for cross-referencing of terms being the only points of variance. Title file record keys and weights were stored for all significant terms in all cases.

Experiments

At first, title keys and weights of non-significant key words were stored only for those articles in which the synonym co-occured with a significant term. Later, in the interests of recall expansion, all title records were stored for synonyms, making no detectable difference in performance.

The first three methods were all intended to improve recall by expansion of the set of useable retrieval keys, using a weighted retrieval scheme to maintain precision at reasonable levels. The only variation lay in the method of assigning weights to the synonyms.

1. At first, the synonym's own weight in each document was stored. The results mentioned previously were obtained using this technique.
2. As a test for the possibility that this technique might be overrating the importance of synonyms, their weights were halved in method two.
3. Method three was designed to test the hypothesis that the synonyms are merely other words for the significant keys (as the term "synonym" suggests). If this were true, some fraction of the weights of the significant keys with which it was found would provide the best measure of its

importance. Salton/McGill [1] recommends using half of the significant terms' weights, and this was attempted. Half weights of any significant terms with which the synonym was paired were stored, no reference being made to documents in which the synonym occurred alone. This would, of course, be useless for detecting cases in which two non-significant terms formed a significant phrase.

Another method of improving recall involves using one key word as a link to the articles referenced by others for which it has demonstrated a high affinity, and this was done next. During these trials non-significant terms were treated in the same fashion as in method one. In addition, file records were created for significant terms which displayed a high correlation to any other terms, containing the record numbers of those terms' title reference lists. The keys to these records were stored with the title and weight lists. This allowed any significant key word to act as a cross-reference to other terms. A query key word was used to retrieve its own set of documents in the fashion previously described, and the new file record, if present for that term, was then used to retrieve the title lists of other terms. The remaining three experiments involved different schemes for assigning weights to these secondary key words.

4. For method four, only half of each weight located through the cross-reference procedure was utilized for retrieval.

5. An attempt was then made to adjust the weights used for indirect references in proportion to the strength of the relationship found between the terms involved. In method five, the value from the third matrix (indicative of that strength) was inverted and subtracted from one. In this fashion, high matrix values became high fractions, and low values low fractions. These values were stored in the files used for cross-referencing, and during retrieval they were multiplied by the weights during indirect title list referencing. Runs were also made using only half of and one quarter of this value during the weight multiplication step. The results presented in Table V, Appendix, are those produced by the best of these runs.
6. In all previous thesaurus constructions, the key words were selected from significant terms and others which occurred with a lower than average total frequency. In method six, non-significant terms with a higher-, rather than lower-than-average total frequency were used with a program otherwise identical to method number five in an attempt to expand the scope of the synonyms. Theoretically, the higher frequency terms would be more widespread in the collection and have a favorable effect on recall.

In order to properly evaluate the results of different thesaurus techniques, it was necessary to have a control. For this purpose, the program was also run without utilizing a thesaurus, building the stem

list from only those terms found to be significant by virtue of having high discrimination values. Table VI, Appendix, contains the results.

Results

Performance. Method one returned 51 percent recall and 78 percent precision. Method number two produced reductions of 4 percent in recall and 6 percent in precision. The decline in the revised average figures was greater than this, indicating that this thesaurus was most useful for the less specific questions. The performance of method three was identical to that of method two.

When the thesaurus was used to provide cross-referencing, method number four produced decreases in both recall and precision of 15 percent and 7 percent, respectively, over method one. Revised averages were close to overall average values indicating that no particular preference for lengthy queries existed when this method was used; however, the variance between the two averages was still greater than that set by method one. The first attempt using method five produced poorer results than method four and another run was made using only half of this value during the weight multiplication step. The results of this were identical to those produced by method four. When only one quarter of the value was used, the results were better, but at no time could better results than those obtained by method one be obtained. Clearly, the cross-referencing of terms as it was done here can only be damaging to results.

Other Considerations. At this stage there were 202 documents and 1,156 stems, and execution times on the VAX were on the order of thirty-eight minutes for the construction of the matrices. Salton/McGill makes reference to a case in which 5,771 stems were retained at this point for a set of 1,033 documents [1]. The algorithm which produces the matrices' values cannot be less than M((N(N+1)/2)+(N(N-1)/2). From this, it should be obvious that this technique becomes extremely expensive in short order as the number of documents increases.

Another point of consideration is the required storage. The combined size of the matrices is N^2 elements, the elements being two bytes each for this research. This is somewhat dependent upon the architecture of the machine used and might be cut by slightly more than one quarter if the equivalent of PL/I's FIXED BIN (7,0) were used for the first matrix; however, paging of memory is imperative unless a sizeable machine is available. Memory requirements for the test program were nearly 5 megabytes, in contrast to the 96 kilobytes used by the alternative system described in Chapter Five. This is dependent on the scope and size of the input data set, and a maximum requirement of approximately fifteen gigabytes for the matrices is theoretically possible, as there are approximately 600,000 unique words in the English language [9], and 14.58 percent of the unique words input were passed to the thesaurus during this work. Of course, this would never be approached in practice, but it indicates that storage requirements could easily exceed the size of the largest machines available today. Execution time was 43 minutes 3 seconds VAX, and, in the absence of a

virtual memory system, programmer implemented paging of the matrices can more than double the time required for execution. If this is necessary, however, it might be used to advantage when new additions to the collection will require frequent alterations of the thesaurus, as the original matrices could be made available to the program used for thesaurus revision, at an additional cost in disk storage.

Alternately, a "sparse matrix" routine might be utilized to minimize storage, though this may be more costly than paging. Attempts to use one algorithm here were abandoned when execution times exceeded four hours.

When no thesaurus was built, execution time and storage requirements were trivial by comparison. If the expense can be justified, however, results can definitely be improved through use of the thesaurus technique. The recall average of 48 percent obtained without the thesaurus compared favorably with that of the more complicated techniques; however, the 66 percent precision produced was poorer than that of any of the methods of thesaurus handling used. A comparison of the results, question by question, showed that thesaurus method one produced equal or superior precision in every case, and the recall produced using a thesaurus was only inferior in some cases of queries of more than one line which would contain a number of key words.

CHAPTER V

RETRIEVAL REFINEMENTS

Weighted Retrieval

The first step in the retrieval program was to obtain the stem list from disk which was produced in the previous steps. Questions could then be entered by the user. The first two words of each question were ignored by the query handler, since this section of any sentence in English is irrelevant to the topic. The remainder of each query was broken into separate words and reduced to stems in the same fashion as that used to handle the document texts. The stem list was searched for each of these words, and the title and term/document weights were retrieved for each stem located. No attempt was made to eliminate duplication of key words in the query, consequently a user could emphasize a topic through the use of term reiteration.

A list of file record keys was maintained for the query, and the title record keys and term/document weights recovered for any key word found were added to this list. If several key words which referenced the same title were found in the query, the weights were summed, producing a total weight for each title found. When the end of the query was reached, this list was gone through, and the title was retrieved from disk and displayed if its total exceeded the threshold established. This procedure was used for production of most of the results mentioned so far.

Modification of this was required for evaluation of the cross-referencing thesauri. This version operated in much the same

way, with the addition that one of the term/document weight fields of each key word's record served as a file record key for cross-referencing purposes. If this was present, it was used to read in a list of the stem list locations of related terms, and this list was then used to obtain the title records of the related terms. Matters then proceeded as before, using only fractional weights for the records located in this fashion.

With either of these systems the retrieval weight threshold could be adjusted by user input to produce nearly perfect recall or precision (though not both). Some experimentation was performed to determine the best default level for the retrieval threshold. Four points were used in these trials;

1. The arithmetic mean of all positive term/document weights for the entire final stem list.
2. The average of these weights.
3. The lowest weight of the significant keys.
4. Half of the largest weight found.

The average of all term/document weights was found to produce the best results when a static retrieval threshold was used (as was done in the stop list threshold experiments described in Chapter Two). In the final version of the retrieval program this was used as level zero which could be adjusted up or down, negative levels of input producing greater recall, positive ones increasing precision.

A second method of improving precision was also incorporated. Upon user request, any additional queries were treated as refinements

of the previous one, and no documents were retrieved which were not located by both. The second query's answers were thus a subset of the first query's. In most cases, perfect precision could be obtained at low levels of recall dynamically through this. The advantage of this technique over level modification lies in the fact that it was applied query by query, and the next topic dealt with would revert back to the original recall level.

An Alternative Method

One other technique was attempted, using a base program which provided no thesaurus or term/document weights. The stem list was made up of terms found to have high discrimination values. The entire query was used, in contrast to the weighted retrieval techniques, and all documents which contained any key found in both the query and the stem list were retrieved.

Early runs using this technique clearly demonstrated the need for some form of weighting technique, as precision was quite poor when all documents for all key words found were retrieved. Since the query handler had no way of knowing how important a given term had been in any document, its decisions on retrieval had to be based on the importance of the term in the question. The principle was simple; if only one term in the query was located in the stem list, all documents to which it refered were retrieved. If, however, more than one term in the query proved to be a key word, only those documents in which more than one of these terms occurred were retrieved. The user could, of course increase recall by repeating important terms in the query.

Comparative Results

The alternative system did not perform well in comparison to the weighted system with thesaurus. It was only tested against a subset of the queries used for weighted tests, producing 50 percent recall and 71 percent precision versus the 57 percent recall and 79 percent precision achieved by the weighted system for those questions. It is possible that better performance would have been achieved had the first words in the query been ignored, however. The system does have a number of advantages. The file building routine is quite inexpensive in terms of execution time, and produces files which require less than one third the disk space required by the weighted system with thesaurus. Updating of the data sets would be comparatively simple when new documents were added to the collection, requiring only the updating of title lists for the existing stem list if the collection deals with a subject whose terminology is relatively stable. Memory requirements of the driver are also considerably less, since the large matrices required for thesaurus construction are not present. The sample document collection could be handled with 64 kilobytes without paging, though nearly 100 kilobytes was actually used here, and extremely large collections would still require less than 500 kilobytes of storage.

These factors make this system advantageous for situations in which large machines are unavailable, frequent updating of the collection would be necessary, or disk storage is extremely limited. There are disadvantages in that the retrieval program takes roughly four times as long to respond, making it less suitable for a multi-user environment, and for most queries, recall was poorer. The variance in

precision results was greater, and, since there is no retrieval weight threshold, recall and precision levels can not be adjusted easily.

CHAPTER VI

CONCLUSIONS

Evaluation Methods

With several discrimination functions and during experiments with suffix removal, a visual inspection of the stem list produced was sufficient for detection of poor techniques. For the most part, however, comparison of answers to a set of sample questions to the correct ones was required.

This set of questions contained three which had only one pertinent document in the collection. They served as a quick indicator of poor performance. One other query was deliberately chosen which contained no significant keys (except when the last discrimination function was used), on the theory that any results obtained for it would serve as an indicator of thesaurus performance. It is a matter of debate as to whether a system which returned incorrect answers to this question is better than one which returns no answers at all (in the latter case, precision is indeterminate), but the results were included in the tables.

The "correct" answers were chosen by hand, using the document abstracts and the key words provided by the CACM (which were not used by the programs) as aids. A certain amount of value judgement was inevitably involved in the decisions as to whether or not a given document was pertinent to the query, however, it is felt that reasonable accuracy was obtained, and the figures were evaluated in a consistent manner from experiment to experiment.

Conclusions

During the course of these experiments, it became obvious that certain basic principles exist which determine the success of some techniques, while at other stages, obtaining adequate results was more of an art. In all cases it was found best to eliminate terms which occurred on the average more than once in every fourth document, and prefix removal was found to be a poor idea during stem reduction. The handling of hyphenated terms and suffix removal techniques involved some degree of custom tailoring to the document set and to the other techniques used by the file building routine. In general, discrimination functions which established a favoritism for low frequency terms were found to be best, though one can go too far in this direction. A function which measures the variance of each term's frequencies in the documents from its average frequency proved to be best here.

The advisability of the synonym thesaurus and weighted retrieval schemes is open to question. A weighted retrieval scheme has mainly the advantage of reduced retrieval program execution time and a greater degree of user control of levels of recall to recommend it, as it does not display a distinct overall improvement in results over the unweighted scheme when used without a thesaurus. The economy in time given by the weighted retrieval program must be balanced against the moderate increase in disk space required for the storage of term/document weights and the increased difficulty of data set alteration during collection expansion. The simpler weighting functions proved best here, dividing a term's frequency in each

document by either its total frequency in the collection, or by the number of documents in which it was found.

The lowest cost in terms of processing time found during these tests was the 4 minutes 5 seconds IBM required by the alternative system mentioned in the preceeding chapter for processing 202 documents. Retrieval program costs in this system were relatively high, but overall results were better than some of the more complicated schemes. Changes in the stop list threshold effected such radical changes in results that it is probable a great deal of "fine tuning" would be required to match this system to a given set of input documents, which would tend to offset the economy in execution time.

The synonym thesaurus techniques used do produce distinct improvements in results, especially in precision; however, they require the use of much larger machines for greater periods of time and produce files which are roughly two to three times the size of those produced without them. Experimental results indicate that synonyms require no special handling; attempts to treat synonyms as anything other than significant keys which were discovered through the thesaurus matrices rather than through use of discrimination values proved to be damaging to results. Since a thesaurus is normally used in conjunction with a weighted retrieval scheme, it presents the same problems when adding to the collection; the thesaurus method found to be best here would not present any additional difficulties.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Salton, Gerard, McGill, Michael J., Introduction to Modern Information Retrieval, McGraw-Hill Inc., 1983

2. Salton, Gerard, Dynamic Information and Library Processing, Prentice-Hall, Englewood Cliffs, New Jersey, 1975

3. Edelhart, M., Davies, O., OMNI Online Database Directory, MacMillan Publishing Co., New York, 1983

4. Salton, Gerard, The SMART Retrieval System; Experiments in Automatic Document Processing, Prentice-Hall, Englewood Cliffs, New Jersey, 1971

5. Kahn, David, The Code-Breakers, The New American Library, Inc., New York, 1973

6. Walpole, Ronald E., Meyers, Raymond H., Probability and Statistics for Engineers and Scientists, MacMillan Publishing Co., Inc., New York, 1978

7. Van Rijsbergen, C. J., Information Retrieval, 2nd Edition, Butterworths, Boston, London, 1979

8. Jones, K. S., "A Statistical Interpretation of Term Specificity and its Application in Retrieval," Journal of Documentation, Vol. 28, No. 1, March 1972, pp. 11-20.

9. Webster's New International Dictionary of the English Language, 2nd Edition, G. & C. Merriam Company, Publishers, Springfield, Mass., 1961

APPENDIX

DEFINITIONS AND FUNCTIONS

Definitions

Precision: $\frac{\text{Number of documents retrieved and relevant}}{\text{Total documents retrieved}}$

Recall: $\frac{\text{Number of documents retrieved and relevant}}{\text{Total relevant documents in the collection}}$

F(t,d): The frequency with which a term occurs in a document.

AF(t): The average frequency of a term in the collection.

TF(t): The total frequency of a term in the collection.

DF(t): The number of documents in which a term occurs.

N: The number of Documents in the Collection.

DV(t): The Discrimination Value of a term.

W(t,d): The Weight of a term in a given document.

Discrimination Functions Used

1. $DV(t) = \sum_{d=1}^{N} \frac{(F(t,d) - AF(t))*(F(t,d) - AF(t))}{AF(t)}$

2. $DV(t) = \sum_{d=1}^{N} \frac{(F(t,d) - AF(t))*(F(t,d) - AF(t))}{AF(t)}$

   for all $F(t,d) > 0$

3. $DV(t) = \sum_{d=1}^{N} \frac{(F(t,d) - AF(t))*(F(t,d) - AF(t))}{AF(t)}$

   for all $F(t,d) > 0$; $-AF(t)$ when $F(t,d) = 0$

4. $DV(t) = \sum_{d=1}^{N} \frac{F(t,d)}{AF(t)}$

5. $DV(t) = \sum_{d=1}^{N} \frac{F(t,d)}{TF(t)}$

6. $DV(t) = \text{Log}_2\ TF(t) - \sum_{d=1}^{N} \frac{F(t,d)}{TF(t)}\ \text{Log}_2 \frac{TF(t)}{F(t,d)}$

Term/Document Weighting Functions

1. $W(t.d) = \frac{F(t,d)}{TF(t)}$

2. $W(t,d) = F(t,d) * (\text{Log}_2(N) - \text{Log}_2(TF(t)) + 1)$

3. $W(t,d) = F(t,d) * DV(t)$

4. $W(t.d) = \frac{F(t,d)}{DF(t)}$

5. $W(t.d) = \frac{F(t,d) * TF(t)}{DF(t)}$

Thesaurus Techniques

1. All synonyms are given their own weights as if they were significant key words.

2. Synonyms are given half of their own weight.

3. Synonyms are given half of the weight of the terms with which they are paired in documents.

4. Synonyms are treated as in method one. Significant keys are cross-referenced to any term which displays a strong co-relationship so that the other term's records may be accessed.

5. Similar to method four with the strength of that relationship entering into retrieval weight calculations.

6. Similar to method five except high frequency stems failing the discrimination test are retained instead of low frequency ones.

TEST QUERIES USED

1. Let me see anything on the MU5 system.
2. I need some AVL tree algorithms.
3. Anything on Mr. Petrov?
4. Give me articles on I/O, especially dealing with the IBM/370 or 360 series machines.
5. Any articles on Pseudochaining?
6. I need information on Relational Databases.
7. What about graph theory?
8. Find articles concerning trees, especially height balanced or balanced trees, and tree searching methods.
9. Find articles concerning database Management systems, including database query languages.
10. Find articles concerning computer architecture including memory organization, parallel processing, multiprocessing and performance evaluation.
11. Find articles concerning operating systems including multiprogramming, task synchronization, timesharing, virtual memory systems and networks.
12. Get me a list of articles on machine architecture.
13. List for me articles about Medical Computing.

TABLE I

Stop List Generation Thresholds

Percentages are of the number of documents; parenthesized percentages are of the total number of words input.

Query #	Total Relevant	100% (1.13%)		50% (.57%)		33% (.38%)		25% (.28%)		20% (.22%)	
		Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.
1	1	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.
2	6	.5	.75	.5	.75	.5	.75	.5	.75	.5	.75
3	1	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.
4	11	.45	.5	.45	.5	.45	.5	.45	.56	.64	.58
5	1	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.
6	5	.4	.67	.4	.67	.4	.67	.4	1.	.4	1.
7	12	.08	1.	.08	1.	.08	1.	.08	1.	.08	1.
8	14	.64	1.	.64	1.	.57	1.	.57	1.	.64	1.
9	8	.88	.78	.88	.78	.88	.78	1.	.75	.62	.71
10	20	.15	.75	.15	.6	.2	.8	.15	.75	.2	.8
11	20	.25	.83	.4	.89	.25	.89	.25	1.	.2	.8
12	10	0.	0.	0.	0.	.2	.29	.2	.29	.2	.4
13	3	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
Average:		.49	.71	.50	.71	.50	.77	.51	.78	.50	.77
Revised Average:		.34	.63	.35	.62	.35	.67	.36	.71	.35	.70

ALGORITHM FOR SUBSCRIPT REMOVAL

Any trailing "S" is first removed, followed by one trailing "E," "Y," or "I."

All combinations of the following subscripts are then removed, with a restriction that a minimum stem size of three characters must be maintained.

"ED," "OR," "OU," "AR," "ER," "IV," "IB," "IZ," "IT," "IC," "AT," "AL," "UA," "ION," "ING," "ABL," "ALL," "ISM," "NES," "ENC," "ATIC," "MENT," "ABIL," "ATOR."

Following this, all trailing "S"s, "E"s, "I"s, "L"s, and "U"s are removed, again maintaining a minimum length.

Finally, one character of any trailing "BB," "DD," "GG," "TT," "MM," "NN," "PP," or "RR" is removed.

TABLE II

Discrimination Function Results

Query #	Funct. 1		Funct. 2		Funct. 3		Funct. 4		Funct. 5		Funct. 6	
	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.
1	1.	.5	1.	1.	1.	1.	1.	1.	1.	1.	1.	.5
2	.5	.75	.5	.75	.5	.75	.5	.75	.5	.75	.5	.75
3	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.
4	.27	.33	.45	.56	.45	.56	.45	.63	.45	.63	.27	.75
5	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.
6	.4	1.	.4	1.	.4	1.	.4	1.	.4	1.	.4	1.
7	.08	.5	.08	1.	.08	1.	.08	1.	.08	1.	.08	1.
8	.64	1.	.57	1.	.57	1.	.57	.89	.57	.89	.57	.89
9	.75	.75	.63	.71	1.	.75	.63	.71	.63	.71	.63	.63
10	.15	.75	.15	.75	.15	.75	.1	.67	.1	.67	.2	.67
11	.25	.83	.25	1.	.25	1.	.15	.43	.15	.43	.2	.5
12	0.	0.	.2	.29	.2	.29	.2	.33	.2	.33	.2	.5
13	0.	Indet.	0.	0.	0.	0.	0.	0.	0.	0.	.33	.5
Aver.	.46	.65	.48	.77	.51	.78	.47	.72	.47	.72	.49	.75
Rev.	.30	.59	.32	.71	.36	.71	.31	.64	.31	.64	.34	.72
E.	.53874		.59136		.61674		.56874		.56874		.59274	
CPU.	43:17.72		44:15.41		43:03.24		44:33.77		44:30.04		40:52.42	

TABLE III

Retrieval Weight Results

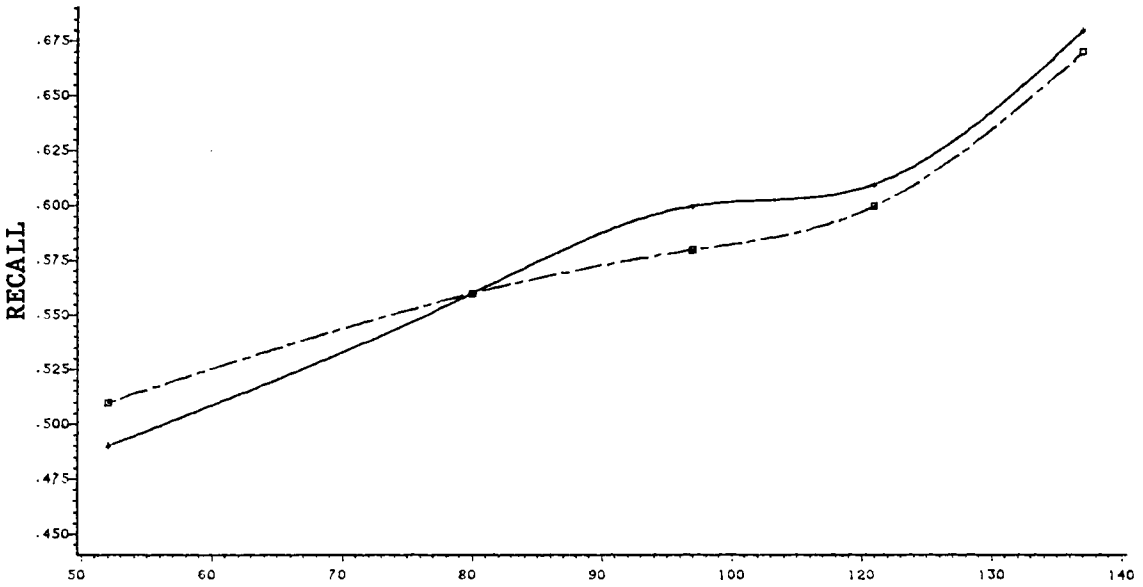
Query #	Funct. 1 Recall	Funct. 1 Prec.	Funct. 2 Recall	Funct. 2 Prec.	Funct. 3 Recall	Funct. 3 Prec.	Funct. 4 Recall	Funct. 4 Prec.	Funct. 5 Recall	Funct. 5 Prec.
1	1.	1.	1.	.5	1.	.5	1.	1.	1.	.5
2	.5	.75	.67	.57	.67	.57	.5	.6	.67	.5
3	1.	1.	0.	0.	0.	0.	1.	1.	0.	0.
4	.45	.56	.27	.5	.27	.75	.55	.67	.18	.67
5	1.	1.	1.	1.	1.	1.	1.	1.	0.	0.
6	.4	1.	.4	.67	.4	1.	.4	1.	.4	1.
7	.08	1.	.25	.75	.33	.8	.08	1.	.33	.75
8	.57	1.	.78	.92	1.	.88	.79	1.	.86	.8
9	1.	.75	.38	.6	.38	.5	.63	.71	.38	.6
10	.15	.75	.25	.83	.2	1.	.1	1.	.2	.4
11	.25	1.	.15	.75	.15	.75	.2	.8	.3	.86
12	.2	.29	0.	0.	0.	0.	.1	.2	0.	0.
13	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
Aver.	.51	.78	.32	.47	.34	.52	.49	.77	.33	.47
Rev.	.36	.71	.32	.56	.34	.63	.34	.70	.33	.57
CPU	43:03.24		41:47.87		42:51.12		41:35.66		41:09.33	

TABLE IV

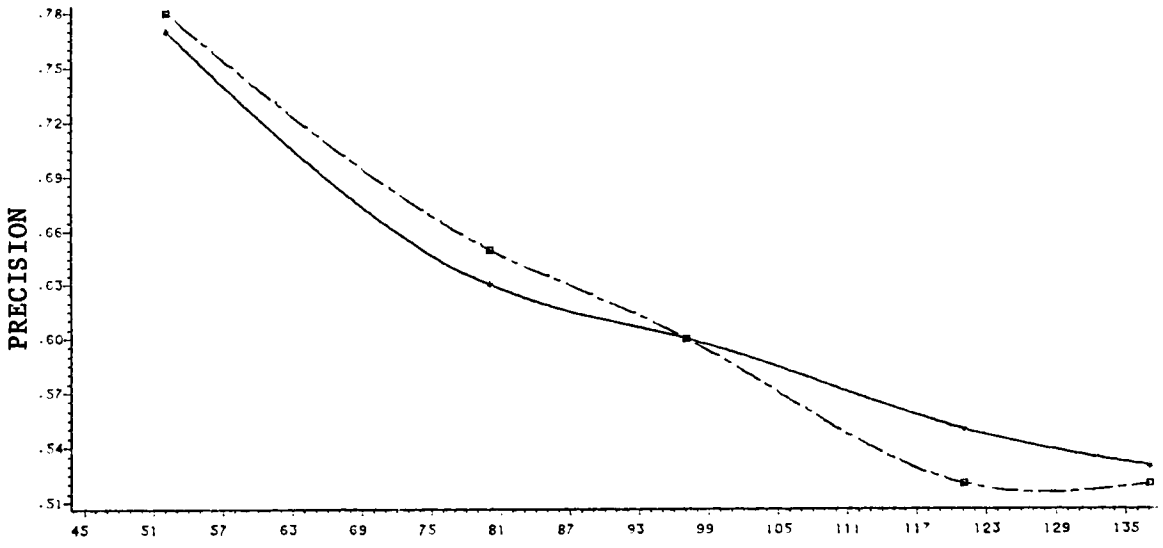
97 Documents Retrieved

Query #	Funct. 1 Recall	Funct. 1 Prec.	Funct. 3 Recall	Funct. 3 Prec.	Funct. 4 Recall	Funct. 4 Prec.	Funct. 5 Recall	Funct. 5 Prec.
1	1.	.5	1.	.25	1.	.5	1.	.25
2	.5	.6	1.	.46	.67	.57	1.	.46
3	1.	1.	1.	1.	1.	1.	1.	1.
4	.55	.38	.36	.4	.64	.47	.36	.4
5	1.	1.	1.	1.	1.	1.	1.	1.
6	.4	.5	.4	.67	.4	.5	.4	.67
7	.25	.75	.42	.5	.33	.8	.33	.57
8	.79	.73	.93	.57	.86	.8	1.	.61
9	.88	.64	.75	.67	.88	.64	.38	.5
10	.45	.64	.35	.7	.4	.67	.4	.73
11	.4	.73	.25	.56	.4	.67	.35	.58
12	.3	.27	.1	.17	.2	.2	.2	.33
13	0.	0.	0.	0.	0.	0.	0.	0.
Aver.	.58	.60	.58	.53	.60	.60	.57	.55
Rev.	.45	.52	.46	.47	.48	.53	.44	.49
E. Aver.	.603287		Not Calc.		.599444		Not Calc.	

Comparison of Functions One and Four



NUMBER OF DOCUMENTS RETRIEVED



NUMBER OF DOCUMENTS RETRIEVED

LEGEND: ⊟⊟⊟ Function 1, ✱—✱—✱ Function 4

TABLE V

Thesaurus Method Results

Query #	Method 1		Method 2		Method 3		Method 4		Method 5		Method 6	
	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.
1	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.
2	.5	.75	.5	.75	.5	.75	.5	.75	.5	.75	.5	.75
3	1.	1.	1.	1.	1.	1.	1.	1.	1.	.25	1.	1.
4	.45	.56	.45	.5	.45	.5	.45	.56	.45	.71	.36	.44
5	1.	1.	1.	1.	1.	1.	1.	.5	1.	.5	1.	1.
6	.4	1.	.4	1.	.4	1.	.4	1.	.4	1.	.4	.5
7	.08	1.	.08	.5	.08	.5	.08	1.	.08	1.	.25	1.
8	.57	1.	.57	1.	.57	1.	.57	.89	.57	.89	.57	1.
9	1.	.75	.63	.71	.63	.71	.63	.71	.63	.71	.75	.67
10	.15	.75	.1	.67	.1	.67	.15	.75	.1	.67	.1	.67
11	.25	1.	.25	1.	.25	1.	.2	1.	.2	1.	.25	1.
12	.2	.29	.1	.17	.1	.17	.2	.33	.2	.33	.1	.2
13	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	Indet.
Aver.	.51	.78	.47	.72	.47	.72	.48	.73	.47	.68	.48	.71
Rev.	.36	.71	.31	.63	.31	.63	.32	.70	.31	.71	.33	.62

TABLE VI

Weighted Retrieval With No Thesaurus

Query #	No Thesaurus Recall	Prec.
1	1.	.5
2	.5	.75
3	1.	1.
4	.36	.44
5	1.	1.
6	.4	1.
7	.17	.5
8	.64	1.
9	.75	.75
10	.15	.75
11	.3	.86
12	0.	0.
13	0.	Indet.
Aver.	.48	.66
Rev.	.33	.61
CPU.	5:12.74	

TABLE VII

Weighted Versus Unweighted Retrieval

Query #	Thesaurus Weighted Recall	Prec.	No Thes. Unweighted Recall	Prec.
8	.64	1.	1.	.78
9	.88	.7	.75	.86
10	.35	.64	.1	.2
11	.4	.80	.15	1.
Aver.	.57	.79	.50	.71

Forty Documents Retrieved

VITA

The author was born in Oak Ridge, Tennessee on November 19, 1948. He moved with his family in 1952 to Knoxville, where he attended elementary school, high school, and the University of Tennessee, obtaining a Bachelor of Science degree in Related Arts and Crafts and Interior Design in 1972. In 1981, he began graduate studies in Computer Science at the University of Tennessee, Knoxville, having been employed in Atlanta, Georgia during the intervening period. He has been employed since January, 1982, as a Graduate Teaching Assistant for the Department of Computer Science.