TESS: AN EFFECTIVE TEXT STORAGE AND SEARCH SYSTEM

W. Bruce Croft
Computer and Information Science Department
University of Massachusetts
Amherst, MA 01003

COINS Tech Report 83-06

This research was supported in part by a Digital Equipment Corporation External Research Grant.

TESS: An Effective Text Storage and Search System

W. Bruce Croft
Computer and Information Science Department
University of Massachusetts
Amherst, MA. 01003

Abstract

A crucial feature of any office system is the method of filing and retrieving documents that contain text. Current systems either use very simple techniques that are not effective in locating relevant information, or place a heavy burden on the user in the query formulation process. Techniques based on probabilistic models of word occurrences in text can provide very effective retrieval compared to conventional techniques, as well as a high degree of flexibility in the style of interaction between the system and the user. In this paper, the statistical techniques are summarized and a text storage and search system (TESS), implemented using these techniques, is described. Methods for dealing with problems arising from this approach, such as updating the word lists and intersecting these lists to produce a ranked list of retrieved documents, are presented. The TESS implementation shows that techniques developed for bibliographic systems can be implemented efficiently in the office environment and that, by careful design of the user interface, the facilities available in this system can be made accessible to a wide variety of office workers.

Categories and Subject Descriptors: H.4.1 [Information Systems Applications]: Office Automation; H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing; H.3.3 [Information Storage and Retrieval]: Information Search and retrieval
General Terms:
Additional keywords and phrases: text storage, document retrieval, probabilistic models

This research was supported in part by a Digital Equipment Corporation External Research Grant.

Introduction

Office information systems provide people in an office environment with tools to help them carry out tasks such as document production, filing, communication and decision-making. Two important criteria that can be used to judge the design of the tools are ease of use and effectiveness. A tool that has powerful functionality but an interface too complex for an average office worker will not be used. It is also essential that a tool should carry out the task it supports with equal or greater effectiveness than the previous methods used.

One of the most important tools in the office system is that used for text document filing and retrieval. Much of the information handled in the office will be in the form of text. Documents such as memos, letters, reports and messages are very common and individual people, as well as organizations, will create large collections of these documents and will need to search them. Most managers will certainly have files containing hundreds or thousands of documents and collections such as a department's memos may be even bigger. In order to find information in these large collections of documents, the office system must provide a means of efficiently searching the documents on the basis of their textual content, as well as specific attributes such as author and date. An example of a query based solely on specific document attributes would be "Find memos written by Joe Smith in January". This should be compared to a more general type of query based on the textual content, such as "Find memos dealing with the sale of widgets to Bill

Clarkson". In order to answer the latter query effectively by locating the relevant documents, the system will require a more sophisticated retrieval strategy than that used for specific attributes.

The main approaches that have been used in the design of document filing and retrieval systems are the "electronic filing cabinet" and Boolean queries with free text. The electronic file cabinet approach (for example, as used in the Xerox Star [18]) gives the user an electronic analog of the manual system of file cabinets, drawers and folders with attached labels. The advantage of this approach lies in its familiarity; it is assumed that office workers will be immediately comfortable with it since the concepts involved are exactly the same as the current manual systems. However, because this approach so closely resembles manual systems, it has the same major disadvantage of having no effective way of searching documents by content, except by sequential scan. This is not important for very small sets of documents but when the number of documents filed is too large for the user to remember the contents of each document, or when a person is searching files other than their own (for example, departmental files), a method of efficiently searching by content is essential.

The other filing and retrieval approach (for example, Tsichritzis and Christodoulakis [23]) allows the user to retrieve documents by specifying character strings connected by Boolean operators. An example of this type of query would be "widgets AND Clarkson". Documents are then retrieved if they contain this combination of strings. This method can be effective, but only

if precisely the right strings and Boolean operators are specified. Documents are either retrieved or not retrieved and the lack of a "near miss" facility leads to very inflexible retrieval. The main problem, however, is the user interface. An extremely heavy burden is placed on the user during the query formulation process, in choosing both strings and Boolean operators. Experience with bibliographic systems using exactly this approach [1] has shown that users tend either to formulate very simple queries and iterate until a set of documents small enough to be scanned manually is found or to delegate the search to a trained intermediary. Neither of these methods of searching is appropriate for an office environment.

The previous discussion shows that current filing systems do not combine the essential features of an office system tool, namely, a friendly user interface, effective performance and efficiency. In this paper, we shall describe the implementation of a text storage and search system (TESS) that does combine these features. TESS is easy to use, efficient and allows flexible and effective retrieval of documents by content. Queries can be specified in natural language or by indicating example documents. Other facilities, such as document classification and user profiles can also be provided. Documents filed through TESS can be considered to be in a single sequential file, although this can be combined with an electronic file cabinet to provide conventional "folder" retrieval.

TESS uses text indexing and retrieval techniques based on statistical models of word occurrences in text [14,17]. These techniques have been tested extensively in systems designed for

scientific literature [5,9,21,22]. Preliminary experiments using these techniques in the office environment [4] have indicated that retrieval is very effective. The purpose of the TESS implementation was not to experiment with new techniques, but to investigate how the best of the current techniques can be implemented efficiently and adapted to an office environment. An important part of this adaptation was to design an interface that would present the facilities provided by the statistical techniques in a clear and concise manner. By giving the users a good understanding of the system, they will be able to achieve the best possible results when searching for relevant documents.

The next section reviews the recent developments in statistical techniques for text retrieval. The reasons for choosing some of these techniques for implementation in TESS are discussed. The advantages of the statistical techniques will be mentioned and algorithms to implement the techniques efficiently in the office environment are suggested. Section 3 contains a description of the TESS system. Section 3.1 outlines the facilities provided in TESS and how they are integrated into an office system. The user interface is described in section 3.2. This is a central part of the design of the system and the menu-based interface is presented in some detail. Efficiency issues such as the number of files used, the storage requirements and the time requirements are discussed in section 3.3.

2. Review of statistical techniques for indexing and retrieval

The two main processes in the statistical approach to text

filing and retrieval are indexing and retrieval. Indexing is the process of representing the content of the documents. Many different indexing strategies have been used in experimental systems [21], but the results indicate that the simplest are the most effective. A typical indexing strategy would be to normalize the document text by removing punctuation, special characters and common words (called stopwords), reduce words to common stems using a stemming algorithm, look up stems in a dictionary to find the corresponding stem numbers and count stem occurrences in the text (Figure 1). A possible final step consists of removing stems which occurred only once in the text in order to reduce the size of the representatives for long documents. It has been shown that this thresholding step will not significantly degrade performance but it can lead to considerable storage savings [4,21]. In the current version of TESS, thresholding is carried out when a document contains more than 50 unique stems.

The document representative produced consists of a set of stem numbers (also called index terms) together with their frequencies of occurrence. A further refinement is to include thesaurus information (i.e. synonyms) in the dictionary lookup. It is possible to generate a thesaurus using statistical information [16], but it will be less effective than a thesaurus constructed manually. Other modifications to the simple indexing process which attempt to capture more of the syntactic and semantic structure of the text, such as using phrases instead of single words, have not proven to be consistently effective and have been left out of TESS to improve efficiency.

It should be mentioned here that queries expressed in natural language are indexed in the same way. Queries and documents, therefore, produce the same type of representation and this implies that documents can be used as queries. This facility will be described in section 3.2.

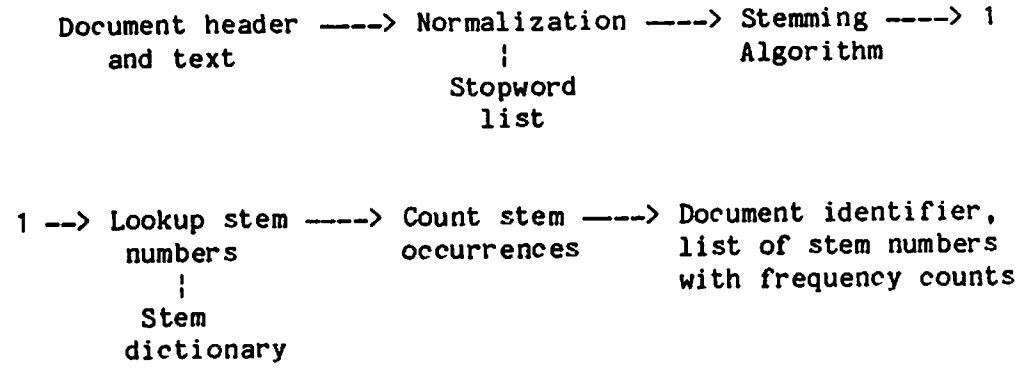


Figure 1. The indexing process.

Each of the steps involved in the indexing process described above is efficient. For example, the stemming algorithm used in TESS is simple and does not rely on a large dictionary of word endings [11]. The algorithm does produce a small percentage of errors but in a system based on statistics this is not crucial. The stopword dictionary consists of a small number (less than 300) of words like "the" and "for" which can be searched very quickly. Statistical indexing also has the advantage of being entirely data-driven. That is, the stems in the dictionary and their patterns of usage depend entirely on the text in the

documents presented to the system. As previously mentioned, external information in the form of a thesaurus can also be used effectively.

The text that is indexed will in general be the entire text of the document. Some documents, however, will be too long to process the full text. Experiments have shown that abstracts, if available, are satisfactory for indexing [16]. Long business documents, which do not typically have abstracts, will have to be treated in another way (see section 3.2).

The basis of the retrieval algorithms in TESS is to estimate, for a given query, the probability of relevance for each document. The documents can then be presented to the user as a ranked list (in decreasing order of probability of relevance) which can be cut off at any point. The estimation of the probability of relevance uses a Bayesian classification model [14,15]. That is, each query is treated as if it divides the collection of documents into a set of relevant documents and a set of nonrelevant documents. The probability of each document belonging to the relevant class is then estimated. If we assume that the documents contain binary index terms (frequency weights of 0 or 1), a document can be represented as a vector $\underline{X}$ where $\underline{X} = (x_1, x_2, \ldots x_v)$ and $x_i = 0$ or 1. If the terms are also assumed to be independent of each other, the simple independence model [14] (described in more detail in Appendix A) specifies that documents should be ranked by the function $g(\underline{X})$:

$$g(\underline{X}) = \Sigma \log \frac{p_i(1-q_i)}{(1-p_i)q_i} x_i \qquad (1)$$

where p_i is the probability that term i is in a relevant document,

q_i is the probability that term i is in a non relevant document, and

the summation is over all terms (in practice this is usually restricted to terms in the query)

The value of q_i is estimated using term frequencies over the whole collection of documents, but initially the only information about the set of relevant documents is contained in the query. If it is assumed that p_i is a constant for all terms in the query [2] and that q_i can be estimated by the proportion of documents in the collection that contain term i, the function used to do the initial ranking of documents reduces to (approximately)

$$g(\underline{X}) = \Sigma \log \frac{N-n_i}{n_i} x_i \qquad (2)$$

where n_i is the number of times term i is used in the collection of documents, and

N is the number of documents in the collection

This means that a score is calculated for each document which is the sum of the weights $(\log (N-n_i)/n_i)$ for each term that matches a query term. The weight is known as the inverse document frequency weight and it implies that low frequency terms are more important for retrieval. Numerous experiments have established the effectiveness of this weight [5,9,22]. In order to scale up the results, the inverse document frequency weight is often calculated as $\log(\max\{n_i\}/n_i)$, where $\max\{n_i\}$ gives the largest value of n_i.

There are two important modifications to this basic model. The first is that the frequencies of terms within documents can be incorporated into the ranking function to give more effective performance [3]. The term weight becomes

$$(w_{ij}/\max\{w_{1j},w_{2j}..w_{vj}\})\log(\max\{n_i\}/n_i),$$

where w_{ij} is the frequency of term i in document j. Whereas the second part of this weight is constant for all documents, the first part (w_{ij}) is different for each document. This is the strategy currently used in the TESS system. The second modification is based on models which do not assume independence between terms [10,13]. Although independence is an unrealistic assumption (many words are strongly related) and much theoretical work has been done on term dependence models, experimental results indicate that they do not give consistent performance benefits. For example, Harper's experiments [9] indicate that apart from providing a heuristic means of finding terms related to query terms, the effectiveness of the dependence model was essentially the same as the simple independence model. This seems to be due to problems with estimating large numbers of parameters. For this reason, the dependence model is not used in TESS.

An important part of the statistical approach is relevance feedback [9,16]. When the user scans a subset of the initial ranked list of documents (for example, the top ten), information can be recorded on the relevance or non-relevance of the documents seen. The system can then use these relevance judgements to form a more accurate picture of the relevant set

and thereby retrieve more documents. The technique used in TESS for relevance feedback is to re-estimate the p_i values in equation (1) by using term frequencies within the set of identified relevant documents. That is, p_i is estimated (approximately) by r_i/R, where r_i is the number of relevant documents that contain term i and R is the number of relevant documents. The original query is also "expanded" by including new terms from the relevant documents. Since there can be many possible terms for this expansion, the best way of deciding which ones to include is to display the terms most likely to be useful (a metric for this is described by Porter [12]) and let the user select them. These new terms are included in the summation in equation (1). Relevance feedback can sometimes be crucial to the performance of the system because the user's first query is often very imprecise. However, in a recent experiment involving a small set of business documents and queries [4], relevance feedback was not required since the desired documents were found near the top of the initial ranking in every case. This led to relevance feedback being available at the user's option in TESS rather than it being a mandatory part of the search. The statistical retrieval process is summarized in Figure 2.

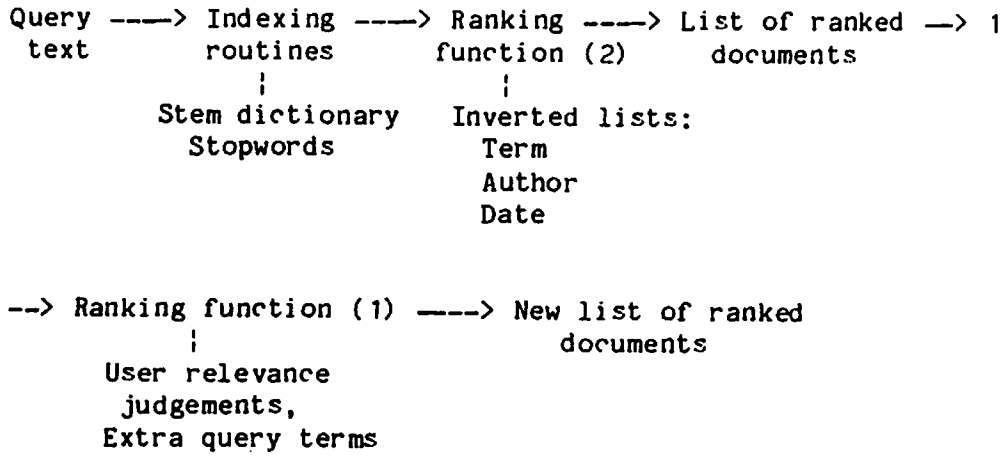


Figure 2. The retrieval process.

The final technique that is important for the TESS system is document clustering or classification [14,17]. In bibliographic systems, document clustering produces groups of similar documents which are used as part of a file organization and for a search strategy. In an office system, the main application for this technique will be the classification of documents into predefined categories. This facility will be used to classify incoming mail messages or to set up user profiles for an information distribution service. The classification problem consists of defining a set of cluster representatives and deciding, for each incoming document, which is the closest matching cluster. For the mail application, an overlapping set of clusters in which documents can belong to more than one cluster is appropriate. Experiments are currently being carried out to determine the best classification method for mail.

The retrieval strategies described here are implemented using an inverted file of documents and terms. This consists of a list, for each term, of the documents described by that term. It has been pointed out [23] that inverted lists lead to considerable overheads when updating the system to include new documents and when merging the lists to calculate scores for the documents. In an office system, the updating of the lists can be done as a background process with little inconvenience to the user. However, merging of the lists to calculate scores can affect the system performance significantly by increasing the response time to queries. For this reason, algorithms have been developed to speed up the calculation of the document scores [7,20]. These algorithms make use of the fact that only the top 10 or 20 of the initial ranked list of documents are required, rather than the entire list. They also make use of the usage characteristics of the terms in the query. For example, one method of calculating document scores is to process each term list in order of increasing length (frequency of use of the term). The score for each document in a list is calculated and this is compared to and if necessary, included in the current top 10 (or 20) scores. Before processing another list, a maximum possible score for any document not seen can be calculated. If this score is less than the lowest of the top 10 seen so far, no more processing is necessary. This algorithm leads to significant speed increases (see section 3) since long term lists are almost never used in calculating the scores. It does require a serial file of the document representations but in section 3.3 we will see that this has other uses.

3. The TESS system

3.1 Functionality

TESS has been designed as a tool to be integrated into an office information system. That is, its functionality is closely linked to other tools such as editors, electronic mail systems and spelling checkers. The main example of this integration is that TESS is designed to be used in conjunction with an electronic file cabinet containing drawers and folders. This gives every user the option of setting up their personal files in the usual fashion. What TESS provides is a means of searching that is independent of any particular folder/drawer arrangement. As a result, large personal files, archived documents, other peoples' files and group files (such as departmental memos) can all be made accessible (Figure 3). Without TESS, many of the documents in the organization could be very difficult to retrieve or, at the very least, difficult to share between people and organizations. From the user's point of view, these document retrieval facilities will require no more effort and, in many cases, will be simpler to use than the current systems.

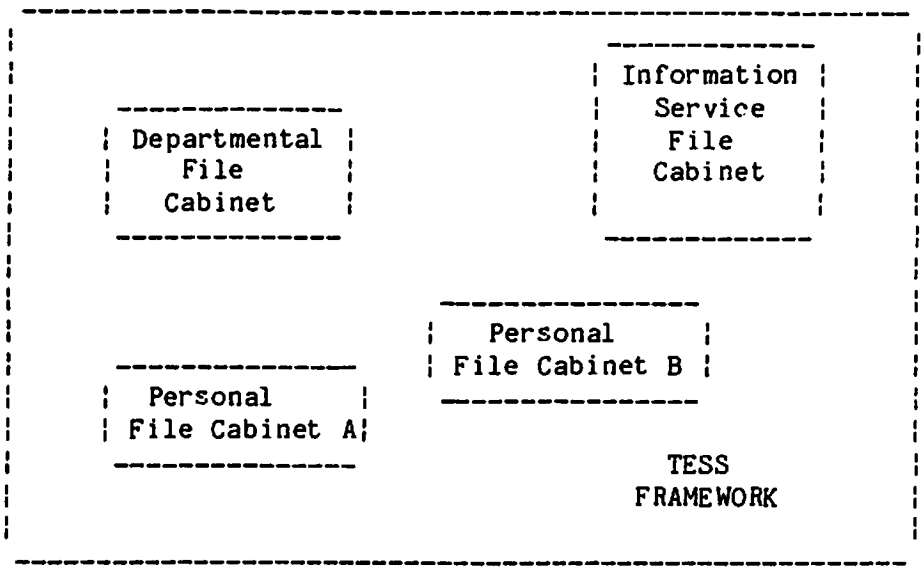


Figure 3. TESS and electronic file cabinets.

In order to concentrate on the user interface and implementation issues, the first version of TESS was implemented essentially as a standalone system. It is this version that shall be described here. TESS is currently being integrated into a commercially available office system.

The main functions provided by TESS are cataloging and retrieval. Cataloging refers to the process of indexing a document and storing its representative in the TESS indexes. Because many candidate documents, such as mail messages, may not be considered important by the user, cataloging is only done on request in the current TESS system. In a future version, more information about document types will be recorded which will enable documents such as departmental memos to be automatically catalogued. Many of the mail messages and memos in an office system will be passed from one user to another. As in

conventional systems, in an office system incorporating TESS each user will make the decision whether to discard the distributed document or to save it using TESS.

Retrieval of documents can be done according to the text content, the values of document header fields such as author, or by a combination of the two. Content queries can be specified in natural language, keywords or with sample documents. Relevance feedback is available if the user is not satisfied with the initial set of retrieved documents. These functions will be discussed in more detail in the next section.

Other functions include the flagging of documents that are to be archived or deleted, displaying the contents of the catalog, and the modification of the dictionary used for cataloging. The last facility, which is not currently implemented, will enable the user to add new words, specify synonyms or abbreviations for existing words and add new stopwords.

An important function mentioned previously is the classification of mail documents into user defined categories. This facility will be essential to filter out important messages from large volumes of electronic mail [6]. The user specifies categories using a combination of header (e.g. author) and content information. The content information can be specified either by the user describing it in natural language or by indicating sample documents that would belong to a category. For example, to specify a category about the sale of widgets to Clarkson, the user can simply point to a current memo on that subject. Although the techniques for classification are

straightforward to implement, experiments are being done to obtain quantitative measures of how effective they will be in practice. Previous experiments with the classification of documents into fixed categories [8] show that agreement with manual classifications may be as high as 80%.

3.2 The user interface

TESS has a menu-driven interface. The style of interaction is not crucial to the design of TESS, but menus were chosen for two reasons; compatibility with the other tools in the commercial office system being used and the lack of suitable workstations to support multiple-window interfaces. Menus and other interface design issues are discussed in Shneiderman [19], but the following general points can be made

1. The menu choices should be unambiguous.

2. Help should be available at any point in the interaction.

3. Help should be specific to the current situation.

4. Error messages should be phrased to assist the user in correcting the problem.

5. Integration should be built in wherever possible. For example, when the user needs to correct an entry or get further information, the appropriate commands should be available without having to interrupt the current task to activate

another tool.

6. Consistency is important. The same methods should be used throughout for actions such as selecting items, going to alternate menus or cancelling a command.

The main problem with menus is that they are often considered irritating by experienced users and some type of bypass mechanism, such as allowing direct typing of commands, is usually provided.

The main TESS menu displays the functions described in the previous section. The actions taken after this for the CATALOG and RETRIEVE functions are outlined below.

CATALOG

If the user selects CATALOG, the series of menus that follow allow the user to select the type of document (mail or other in the current system) and specify the names of documents to be cataloged. If mail documents are being cataloged, they are selected from a display of the user's current mail file. The specified document or documents are then indexed. If the document is too long to index the full text (i.e. longer than some prespecified length), the user is asked to enter either a specification of part of the text that could serve as an abstract, or a description of the document. The same technique is used for documents with too few significant words and it will also be used in a later version to describe non-text documents (such as pictures) or off-line documents. Words in the text which are not found in the dictionary are displayed to the user

who is asked to correct any spelling errors. After this step, a background process is initiated to update the inverted lists and dictionaries. Figure 4 summarizes the sequence of system actions taken for the CATALOG function.

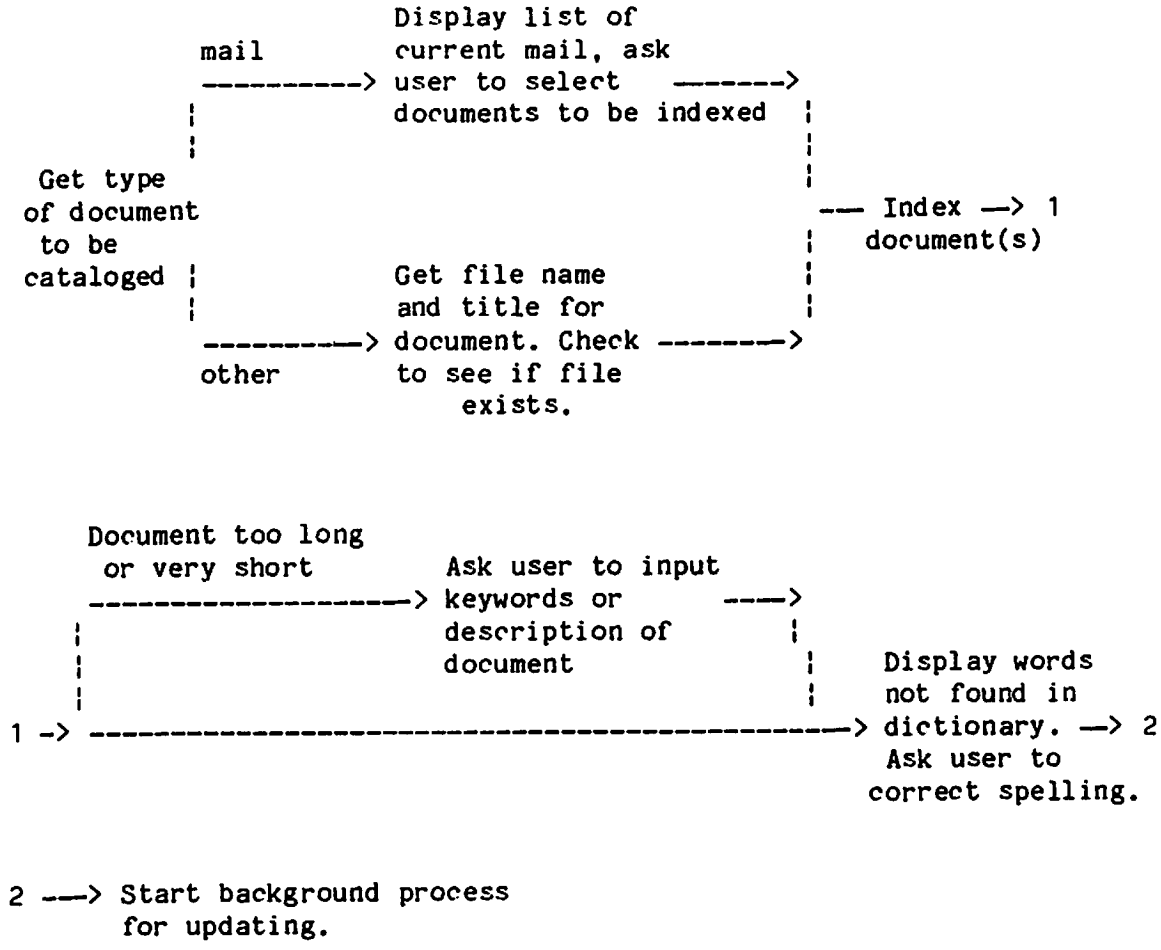


Figure 4. The CATALOG function.

RETRIEVE

The retrieve function allows the user to specify queries in the following ways;

1. Identify a specific document which is similar to the documents desired.

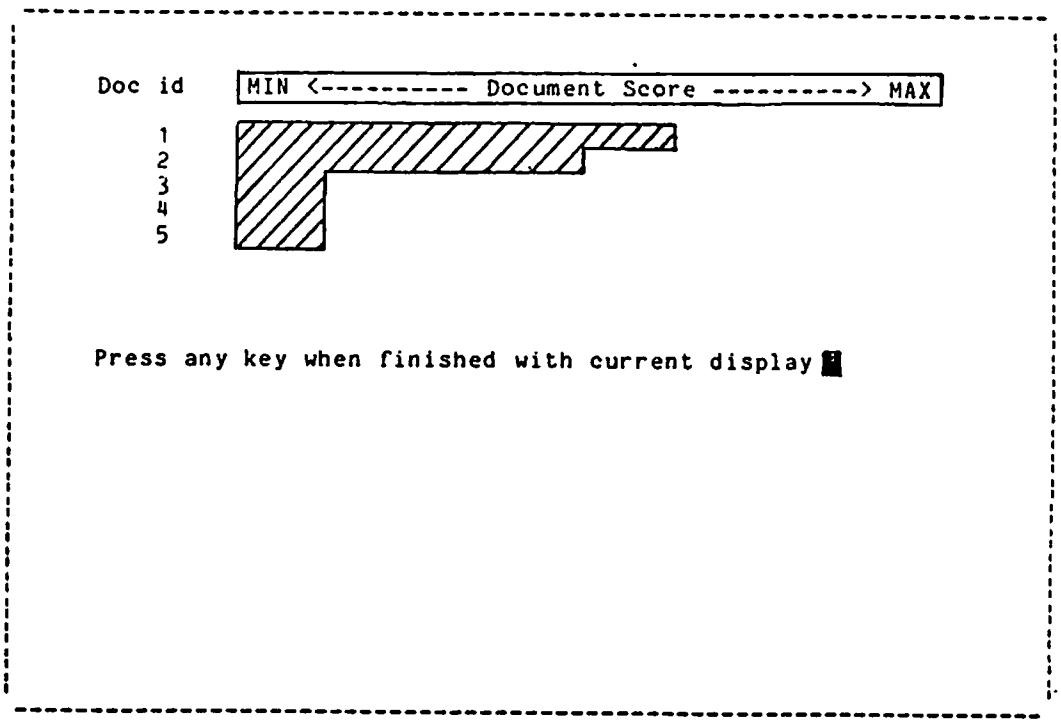
2. Describe the contents of the desired document(s) using natural language or a keyword list.

3. Specify the desired values of the author, date, or document type.

4. Combine the latter two methods to further restrict the documents retrieved.

After entering the query the user is asked to check the spelling of any words not found in the dictionary and then has the option of adding terms if this is necessary. A display of synonyms for certain words to clarify or strengthen a query may be requested. The user also specifies how many documents are to be displayed and which categories of documents are to be searched. The retrieved documents will be displayed in order of decreasing relevance as determined by the function defined in equation (2). In addition, a bar chart will be displayed indicating the estimated relevance of the document to the query. The bar chart is an extremely useful device for conveying to the user information about the system's opinion of the retrieved documents. For example, Figure 5 shows two displays seen by the

user after retrieval. The first display shows the bar chart display for the five documents requested by the user with a separate window for the possible commands. The second display gives the specifications of the documents retrieved. In this example, it is very clear that the system regards the first two documents as very strong candidates for being relevant to the query and the others much less so. Similarly, if no documents had a high probability of relevance, this would be immediately apparent and the user would not be as disappointed if the documents did turn out to have very little to do with the query.



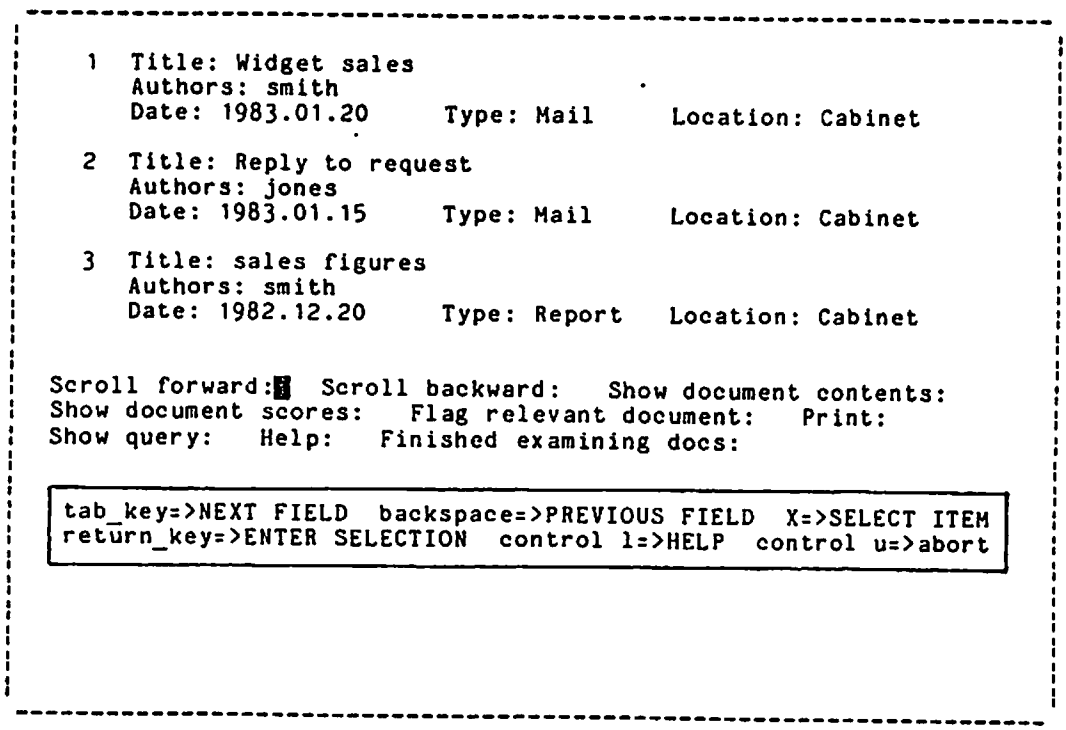


Figure 5. The display of retrieved documents.

The user can browse through the contents of the retrieved documents and return to the displays mentioned above. While the contents of a retrieved document are being scanned, words in the text which have the same stems as those in the user's original query are highlighted with reverse video. This gives the user more information about why the document was retrieved and how the stems are used. Obviously, if the document is exactly what was required, this feature is not necessary but sometimes the relationship between the document and the query is not immediately clear. Another command allows the user to display the original query in a separate window. Figure 6 shows examples of the two displays used when scanning document contents.

Relevance feedback is initiated if the user indicates that more relevant documents are required. After specifying the relevant documents in the initial retrieved list, the user is shown a list of possible words from the documents which could be added to the query. These words are ranked in order of potential usefulness as described in section 2. The user selects words from this list and a new set of retrieved documents will be displayed. Finally, if no relevant documents at all were found, the user must reformulate the query. In the next version of TESS, the system will provide help in this reformulation process by supplying dictionary information such as term frequencies and synonyms. The steps in the TESS RETRIEVE function are shown in Figure 7.

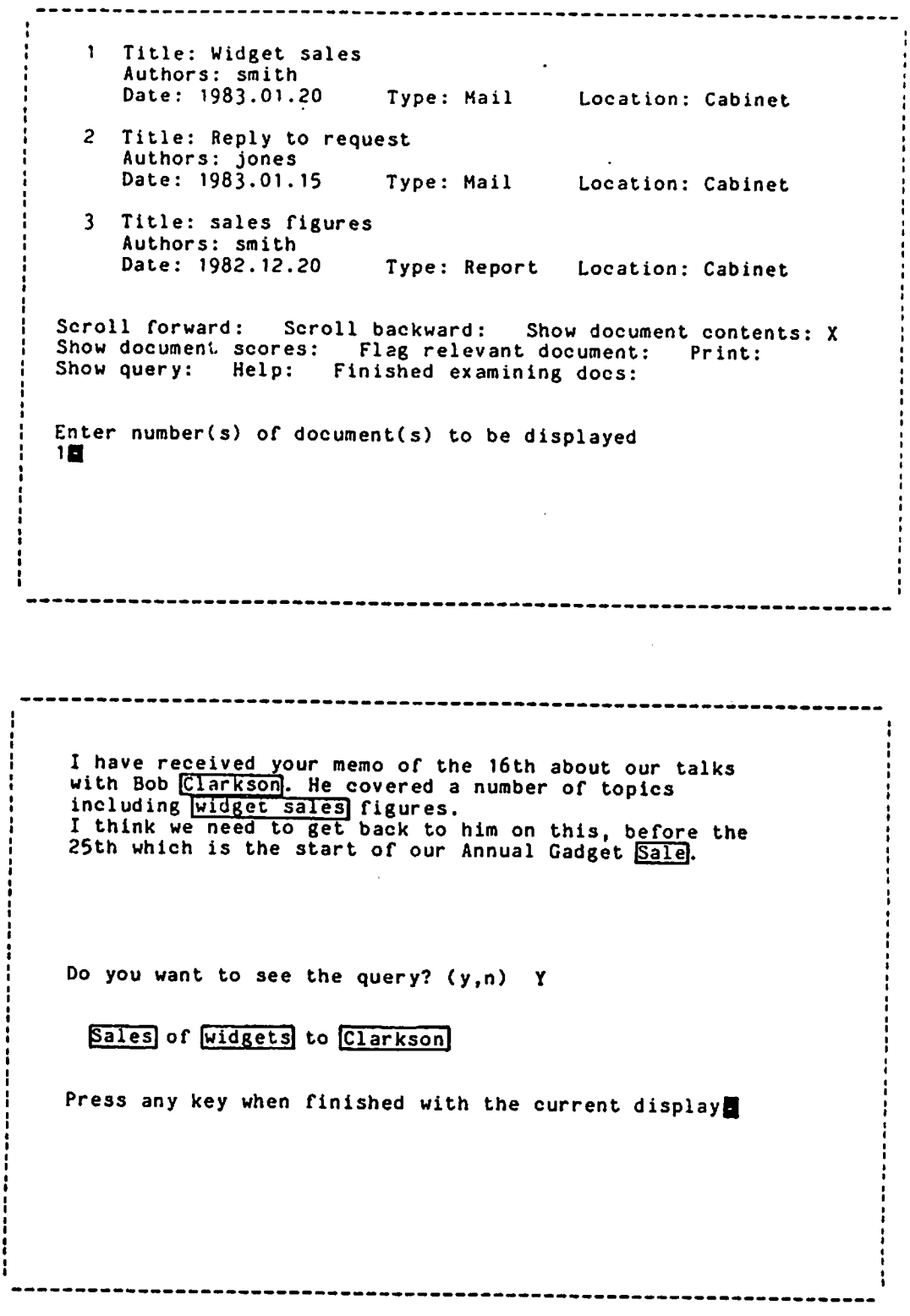


Figure 6. Browsing the contents of a retrieved document.

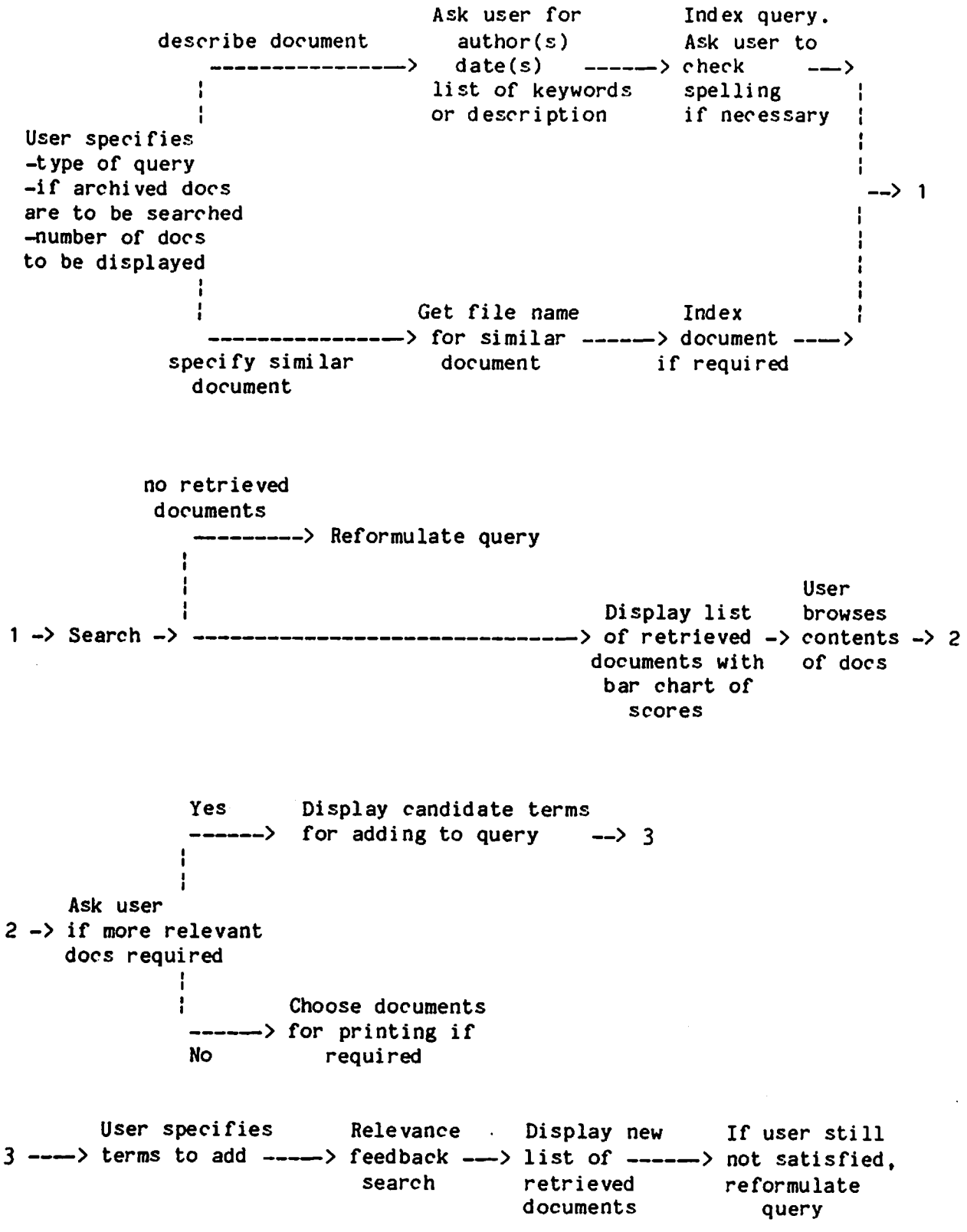


Figure 7. The RETRIEVE function.

The main feature of this interaction is that in most cases it requires very little user effort. In more difficult cases where the original query is not well specified, the system has techniques available (e.g. relevance feedback) to assist the user in finding the required documents. Systems based on Boolean queries and free text have no such techniques and place the burden on the user to reformulate the query.

3.3 Implementation issues

The current version of TESS is running under VMS on a VAX 11/750. An overall picture of the files used appears in Figure 8.

GLOBAL	LOCAL
Word dictionary	Local word dictionary
Stopwords	Document information
Stem dictionary	Document-stem file
	Postings
	Term index
	Author index
	Date index

Figure 8. TESS files.

The files are divided into two types. Global files are shared by all users whereas there is a version of the local files for each user catalog. The organization shown was designed to give

maximum flexiblity in the design of the interface rather than minimum storage overhead. The subset of files that would be absolutely necessary to carry out indexing and basic retrieval would be the stopwords file (a list of words that can be removed from the text, indexed by the full word), the stem dictionary (records the stem number associated with each stem, indexed by stem), the postings file (information on the frequency of use of the index terms, indexed by term number) and the inverted lists (term index, author index, date index). The use of these files has been described in Figures 1 and 2 except for the postings file which is used to calculate the term weights in the document ranking function.

The indexes can represent a considerable storage overhead compared to a sequential file of the document text. However, a recent study with a sample of business text showed that the average number of index terms in a document representative was less than 25% of the number of words in the incoming text [4]. This meant that the average representative contained less than 50 index terms. This estimate assumes that most documents are formal memos rather than typical electronic mail messages. Because each index term is represented by a fixed length stem number which could be held in two bytes, a user catalog of 2000 documents would require about 200KBytes for the inverted lists. About 30-50KBytes more could be needed for the other files, depending on the number of unique index terms identified. This is for a minimal system. If we assume that a document of 200 words requires 1500 bytes of storage, then the sequential file of 2000 documents would require approximately 3MBytes. This means

that the overhead for the minimal system is about 8% of the storage required for the document text. If the information about the frequency of a term in a document is recorded, this would make the overhead approximately 12%. The overhead for the current TESS implementation is greater than this estimate for two reasons; no effort has been made to compress the data and extra files are used to enhance the user interface. The cost of this storage overhead must be considered in terms of the benefits gained by the statistical approach. In any office where filing and retrieving documents is an important task, these benefits will be significant.

In order to have the interface features mentioned in the last section and to support relevance feedback, the following files are required in addition to the local indexes. The global word dictionary (a list of all words with synonyms, indexed by full word) is used to check the spelling of words and to retrieve the stem numbers for the words. This means that indexing proceeds by first checking the stopword list and then searching the word dictionary rather than stemming the word and using the stem dictionary. This method is used for three reasons

(a) Spelling dictionaries are available in most office systems (and therefore do not increase the overhead).

(b) Spelling errors could significantly degrade retrieval performance [4].

(c) The word dictionary allows the user to see what words reduce to the same stem. This is very useful in relevance feedback or query reformulation.

The local word dictionary contains words which are not in the

global dictionary. These words are identified during the indexing process and are placed in the local dictionary to save on updating costs to the much larger global dictionary.

The document information file contains data such as the file specification of the document (documents are referred to in the indexes with a unique identifier), the document type and other header information. The document-stem file contains a list, for each document, of the index terms in the document representative. This information is crucial to the relevance feedback process in order to quickly access the terms in the identified relevant documents. The storage overhead for this file is the same as that for the term indexes.

The document scores are calculated using the algorithm mentioned in section 2. A maximum of 20 of the top ranked documents are found and the user specifies how many of these are to be displayed. This algorithm can save a considerable amount of computation. For example, in a series of experiments done with different document collections [20], the number of document scores calculated was reduced on average by 40%. In terms of disk accesses, the document ranking algorithm will require approximately one disk access for each query term used. This is because the term-document inverted lists are quite short (less than 20 documents per term) in a typical collection of office documents. Since most initial queries will contain very few index terms (typically 2-5), this means that retrieval is a very fast operation. Using documents as the basis for queries produces more index terms, but the ranking algorithm significantly reduces the number of terms (and, therefore, the

number of disk accesses) required to calculate document scores.

The speed of the indexing process depends almost entirely on the speed of the word dictionary lookup. The dictionary used in TESS has 30,000 entries and is currently being re-implemented to increase the speed of indexing. Spelling dictionaries available with some office systems are capable of processing text as fast as it can be scrolled on a typical terminal. The indexing process should, therefore, be able to achieve this degree of efficiency.

4. Conclusion

The TESS system demonstrates that it is possible to provide a powerful text storage and retrieval tool for the office by using techniques based on statistical models. These techniques allow users to locate relevant information easily and efficiently. They also provide the mechanisms for novel facilities such as adapting to a user's requirements (relevance feedback) and classifying incoming mail messages. The process of specifying a query or filing a document is simpler in TESS than in many current systems. The main cost associated with TESS is the storage overhead for the word indexes, which is estimated to be approximately 10-15% of the size of the sequential file of documents. In any environment where the number of text documents kept on file is too large for a sequential search (which includes most offices), this cost will be outweighed by the benefits obtained by being able to locate important information.

APPENDIX A : THE INDEPENDENCE MODEL OF DOCUMENT RETRIEVAL

Each document is assumed to be described by a binary vector $\underline{X} = (x_1, x_2, .. x_v)$, where $x_i = 0$ or 1 indicates the absence or presence of the ith index term. A decision rule can be formulated by which any document can be assigned to either the relevant or non-relevant set of documents for a particular query. The obvious rule is to assign a document to the relevant set if the probability of the document being relevant given the document description is greater than the probability of the document being non-relevant, that is if

$$P(\text{Relevant} | \underline{X}) > P(\text{Non-Relevant} | \underline{X})$$

A more convenient form of the decision rule can be found by using Bayes' theorem. This new rule, when expressed as a weighting function is,

$$g(\underline{X}) = \log P(\underline{X} | \text{Relevant}) - \log P(\underline{X} | \text{Non-Relevant})$$

This means that instead of making a strict decision on the relevance of a document, the documents are ranked by their g(.) value such that the more highly ranked a document is, the more likely it is to be relevant.

The probabilities $P(\underline{X} | \text{Relevant})$ and $P(\underline{X} | \text{Non-Relevant})$ are difficult to calculate directly. However, they can be approximated in a number of different ways. If the assumption is made that the index terms occur <u>independently</u> in the relevant and non-relevant documents then

$$P(\underline{X} | \text{Relevant}) = P(x_1 | \text{Relevant}) P(x_2 | \text{Relevant}) .. P(x_v | \text{Relevant})$$

and similarly for $P(\underline{X} | \text{Non-Relevant})$.

Let $p_i = P(x_i = 1 | \text{Relevant})$ and $q_i = P(x_i = 1 | \text{Non-Relevant})$

where these are the probabilities that an index term occurs in the relevant and non-relevant sets respectively. Then

$$P(\underline{X}|\text{Relevant}) = \prod_{i=1}^{V} p_i^{x_i} (1-p_i)^{1-x_i}$$

$$P(\underline{X}|\text{Non-Relevant}) = \prod_{i=1}^{V} q_i^{x_i} (1-q_i)^{1-x_i}$$

and

$$g(\underline{X}) = \sum_{i=1}^{V} x \log \frac{p_i(1-q_i)}{(1-p_i)q_i} + \sum_{i=1}^{V} \log \frac{1-p_i}{1-q_i}$$

The second term of this function will be constant for a given query and will not affect the ranking of the documents. The first term involves a summation over all the terms in the document collection, but this is usually restricted to just the query terms.

Acknowlegements

The author wishes to thank Sally Lasater and Tony Rogers for their efforts in implementing TESS. Sally Lasater was also involved in the design of the interface.

References

1. BARRACLOUGH, E.D. On-line searching in information retrieval. Journal of Documentation 33, (1977), 220-238.

2. CROFT, W.B. AND HARPER, D.J. Using probabilistic models of information retrieval without relevance information. Journal of Documentation 35, (1979), 285-295.

3. CROFT, W.B. Document representation in probabilistic models of information retrieval. Journal of the American Society for Information Science 32, (1981), 451-457.

4. CROFT, W.B. Experiments with automatic text filing and retrieval in the office environment. SIGIR Forum 16, (1982), 2-9.

5. CROFT, W.B. Experiments with representation in a document retrieval system Information Technology 2, (1983), 1-21.

6. DENNING, P.J. Electronic Junk. Communications of the ACM 25, (1982), 163-165.

7. DOSZCOCS, T. AND RAPP, B.A. Searching Medline in English. Proceedings of the 42nd ASIS Annual Meeting. White Plains, Knowledge Industry Publications, (1979), 131-139.

8. GAUTAM KAR, B. AND WHITE, L.J. A distance measure for automatic sequential document classification. Technical Report CISRC-75-7, Ohio State University, 1975.

9. HARPER, D.J. AND VAN RIJSBERGEN, C.J. An evaluation of feedback in document retrieval using co-occurrence data. Journal of Documentation 34, (1978), 189-206.

10. LAM,K. AND YU, C.T. A clustered search algorithm incorporating arbitrary term dependencies. ACM Transactions on Database Systems 7, (1982), 500-508.

11. PORTER, M.F. An algorithm for suffix stripping. Program 14, (1982), 130-137.

12. PORTER, M.F. Implementing a probabilistic information retrieval system. Information Technology 1, (1982), 131-156.

13. VAN RIJSBERGEN, C.J. A theoretical basis for the use of co-occurrence data in information retrieval. Journal of Documentation 33, (1977), 106-119.

14. VAN RIJSBERGEN, C.J. Information Retrieval. Second Edition, Butterworths, London, 1979.

15. ROBERTSON, S.E. AND SPARCK JONES, K. Relevance weighting of search terms. Journal of the American Society for Information Science 27, (1976), 129-146.

16. SALTON, G. Automatic information organization and retrieval. McGraw-Hill, New York, 1968.

17. SALTON, G. AND McGILL, M.J. Introduction to Modern Information Retrieval. McGraw-Hill, New York, 1982.

18. SEYBOLD, J. The Xerox Star: A professional workstation. The Seybold Report on Word Processing 4/5 (1981), Seybold Publications, Media, PA.

19. SHNEIDERMAN, B. Software Psychology. Winthrop Publishers, Cambridge, Mass., 1980.

20. SMEATON, A.F. AND VAN RIJSBERGEN, C.J. The nearest neighbour problem in information retrieval. Proceedings of the 4th International ACM SIGIR Conference, SIGIR Forum 16, (1981), 83-87.

21. SPARCK JONES, K. AND BATES, R.G. Research on automatic indexing 1974-1976. British Library Research and Development Report 5464, Computer Laboratory, University of Cambridge, 1977.

22. SPARCK JONES, K. Research on relevance weighting 1976-1979. British Library Research and Development Report 5553, Computer Laboratory, University of Cambridge, 1980.

23. TSICHRITZIS, D. AND CHRISTODOULAKIS, S. Message Files. Proc. ACM SIGOA Conference on Office Information Systems, Philadelphia, 1982.