Interactive Information Retrieval

G. Salton

Technical Report
No. 69-40

August 1969

Department of Computer Science
Cornell University, Ithaca, N. Y.

Interactive Information Retrieval

G. Salton*

Abstract

The advent of time-sharing computer organizations and input-output console equipment has made it possible to experiment with interactive information handling methods in which the user takes on important functions both in planning and in executing information searches.

The principal interactive text storage and retrieval processes are briefly reviewed, including text editing, text analysis and indexing, query formulation, and information searching. The user's role is stressed in each case, and particular attention is paid to interactive search procedures in which both the user queries and the stored information files become altered as a result of the user-system interaction. Evaluation results obtained with the SMART retrieval system are exhibited for some of the proposed methodology.

1. Introduction

Until recently, the mechanization of information retrieval operations was largely confined to an automatic matching process between incoming user queries and stored information items. The queries obtained from the user population would be manually transformed by subject experts into search statements usable by the mechanized search system, and responses would be

---

*Department of Computer Science, Cornell University, Ithaca, N. Y. 14850.

This study was supported in part by the National Science Foundation under grant GN-750.

obtained within a few days, or a few weeks, depending on the particular system in use. The user himself would not be a part of the process, once his original search statement was properly received, and no feedback would be received from the user population other than somewhat general statements expressing a greater or smaller degree of satisfaction with the search results

With the advent of time-sharing computer organizations, and of console, display, and transmission equipment, a potentially large user population can now obtain direct access to extensive, central information stores. Messages can then flow back and forth between user and system, expressed in many cases as relatively free language forms, which hopefully lead to an understanding on the part of the user of what the system is able to provide, and on the part of the system of what the user wishes to obtain. Such an interactive mechanism can serve initially to construct the files by introducing new information and identifying relationships between the items to be stored, to construct properly formulated user queries through a process of negotiation between user and system, to perform information searches designed to furnish individual responses to specific user needs, and to update the file structure following the searching process, if there is reason to believe that such a reorganization would lead to improved responses in the future.

In the present study, the interactive retrieval environment is briefly examined, and examples are given of on-line procedures for file generation, content analysis, file organization, and file search. Various types of search negotiations are then described, initiated either by the system or by the user, and leading in each case to a reformulation of the user queries so as to insure a better correspondence with real user needs.

Alternatively, the document identifications themselves may be changed by seeing to it that documents which are jointly relevant to certain user queries are identified by closely similar content descriptions. Evaluation results are cited to indicate the amount of improvement in retrieval effectiveness obtainable through the interactive process. Finally, certain file protection problems are mentioned which require resolution before automatic information networks actually become usable in practice.

2. Text Prodessing and Analysis

A) Input Processing

A fully-automatic library system is often pictured as consisting of a central equipment complex and a set of input-output console devices which allow the user to communicate with the system and vice-versa. The consoles may be simple typewriter-like units with a standard keyboard used to enter messages, and a standard printing unit to receive computer output. In many cases, the input keyboarding is simplified by providing special function keys in addition to the normal keyboard, or special overlays over the standard keys. Graphic displays may also be used, instead of the more conventional printed output. Such devices make possible the display and manipulation of complete pages of printed or graphic information on the face of a cathode ray or storage tube, and lend themselves to the editing and updating of existing files as well as to the creation of new ones. [1,2,3]

Among the operations generally provided for text editing purposes are the following: [4]

a) browsing in a text by moving the display forward or backward, a line or a page at a time;

b) effecting structural changes by creating or erasing special links between pieces of a text, or items in a file;

c) editing a given text by insertion, deletion, or substitution of new words, or phrases, or paragraphs to replace the original ones;

d) reformatting a text by specifying column arrangement, paragraphing, justification, underscoring, and so on.

Normally, the utilization of such a system may be simplified if instructions are given to the user in the course of the operations. Thus, the user may be told at each step what alternative actions may be taken, and how these actions are to be specified. Various prompting techniques may also be included to lead the user along, and to explain to him the system's understanding of previous user actions. Finally, a complete teaching program, or "user guide" may be implemented, designed to introduce the user to the system by means of suitable explanations and examples. [5]

No matter what the specific nature of the application, the operations of an on-line system are normally based on five different types of programming modules:

a) a teleprocessing system which picks up input from the console units and returns answers to these units;

b) a dialog program which carries on the conversation by receiving input dialog from the user, and creating appropriate system responses;

c) a translator program which analyzes and interprets the input statements;

d) a message queueing program which transmits analyzed input to the processing modules and system output to the teleprocessing system;

e) a processing module which actually carries out the required operations, such as the file editing or the information search.

In the remainder of this study, only the translation and processing modules are examined in detail, with emphasis on query processing and file search and retrieval operations.

B) Content Analysis and Indexing

When a text, or document, is originally introduced into the system, it is necessary first to determine its content and to assign to the item appropriate information identifiers. These identifiers are then used to locate the item in storage and to retrieve it on demand. In most practical applications, the assignment of content identifiers — sometimes known as terms, keywords, concepts, and so on — is performed manually prior to the actual input operations. In some experimental systems, however, automatic text analysis procedures may be used for indexing and content specification. [6,7,8]

In the automatic analysis mode, the vocabulary of the input texts, or queries, must inevitably serve as the basis for the generation of content identifiers. However, the words occurring in a text are normally subject to extensive modification before they are actually accepted index terms. A typical procedure may be outlined as follows:

a) the text words are arranged in alphabetical order and reference is made to a word frequency list specifying for each entry its frequency within a given collection or subject area;

b) common words whose frequency is too high are rejected, as are the very rare words whose frequency is too low;

c) weights are assigned to the remaining words based on the dictionary specification and the word frequency in the given text;

d) relational indicators may be assigned to pairs or triplets of cooccurring words to identify them as phrase components;

e) the weighted terms extracted from the original texts may be expanded by addition of new related items extracted from a variety of stored dictionaries, such as synonym lists, hierarchies, or phrase lists, or, alternatively the related terms may be identified statistically by studying the vocabulary in a given document collection;

f) The dictionaries, word frequency lists, and other linguistic analysis tools may be updated during the indexing process by addition of previously missing items occurring in newly introduced document or query texts, or by subtraction of unwanted items.

While the details of the analysis process vary from one system to another, the procedures are normally based on three kinds of tools: stored dictionaries of various types; statistical information derived from the document collection under consideration; and, more recently, feedback information supplied by the user during the indexing process. The following types of dictionaries or word lists may be of use: [8]

a) word stem and word suffix dictionaries used to break down text words into stems and suffixes, repsectively;

b) common word lists used to reject unwanted items in the course of the analysis process;

c) thesauri and synonym lists used to supply related or synonymous entries for the originally available terms;

d) phrase dictionaries used in generating complete phrases from individual term components.

The automatic assignment of phrases as content identifiers (instead of individual words alone) may depend on the inclusion of the phrase components in a phrase dictionary, or, alternatively, on the characteristics of the components within the documents of a collection; that is, whether the components cooccur sufficiently often in the same documents, or paragraphs, or sentences; whether they exhibit an appropriate type of syntactic relationship; whether the occurrence of a component within a phrase accounts for a minimum percentage of the total number of occurrences of the component, and so on.

The analysis process produces in each case an assignment to queries and documents of sets of terms, consisting of individual words or phrases together with term weights, reflecting the presumed importance of each term. These term sets are then compared during the search process to obtain a similarity coefficient between documents and queries. The term sets may also be modified during the operations by the user or by the system as will be seen when the search negotiation methods are described.

3. Query Analysis

In a recent survey dealing with query formulations, four types of query languages are recognized: [9]

a) the "fixed form" languages, consisting of a single list of search terms, interrelated possibly by Boolean connectives such as "and", "or", and "not",

b) the "function and parameter" languages, where certain verbs (such as GET, SEARCH, PRINT, etc.) denote functions, and the objects which follow the verbs are parameters operated upon by the given function; a single well-defined relation normally exists between each function and each parameter;

c) the "phrase structure" languages which provide many relations and many entities or objects, as well as a small number of functional words specifying the type of relation intended; the connections between entities and relations are simple, and no syntactic ambiguities are present;

d) the natural language subsets consisting of relatively free natural language statements in certain subject areas.

For the fixed form languages, no query analysis is required at all, since the search terms are directly provided in the query statement. To retrieve a given item, it is then only necessary to check that the appropriate combination of terms is included in each document identification. Many keyword-based, "coordinate indexing" systems operate with such fixed form languages.

The analysis process is minimized for the function and parameter languages, which utilize a small set of recongized commands, a specific set of entries, and a number of attributes and attribute values for each entity [10,11] In a document retrieval environment, this implies the existence of commands such as FIND, SEARCH, COLLECT, TYPE, PRINT, DISPLAY, SAVE, etc., operating on various types of bibliographic files. For a standard document file, the attributes might consist of

document authors;
document titles;
terms or content identifiers assigned to documents;

citations from one document to another;
publication dates of documents;
document serial numbers.

Normalization techniques may also be usable which correct misspellings of certain attribute values (for example, author names, or specific keywords, or title words) included in the queries; alternatively, truncated query words, limited to a fixed number of letters to simulate a word stem reduction, may also be acceptable as part of a query formulation.

A typical query statement might be specified as:

"PRINT document reference FOR title words
'PLUM' AND 'PEAR' AND date AFTER '1965'."

It is clear from the formulation that the analysis procedure may remain quite rudimentary: each command operates as a function call acting on the immediately following entity, and attribute values immediately follow the respective attribute names ('plum' and 'pear' are title words, and '1965' is a publication date).

The phrase structure languages usually include a large number of entities and relations, but only a small number of function words and simple connections between entities and relations. In the case of a file dealing with authors and documents, the relations might included among many others: [12]

document relations:
was written by
has the title of
was cited by
appeared in journal

author relations:
holds degrees in
belongs to professional society
is employed by

journal relations:
is edited by
is published by, etc.

Relatively free natural language query formulations are permitted, relating entities and relations. However, the sentence structure used to express the information used must obey the usual phrase structure rules, in the sense that it consists of juxtaposed or nested phrases.

In one well known on-line retrieval system, a query consists of a "sentence", which may be broken down into "clauses" connected by the word OR; the clauses may in turn consist of "phrases" connected by AND. Each phrase may include a "field specifier" following by a number of literal strings; the field specifiers consist of quantifiers (all, some, no) followed by a "field name" (citations, phone numbers, authors, documents, etc.), and the literal strings may breakdown into qualifier (not, larger than, greater, equal, etc.) followed by a string of text. [13]

A typical phrase structure-type query might then take the following appearance:

> "find all authors names whose papers appeared between 1965 and 1967 in English language journals published in Europe, <u>or</u> all journal editors who were also authors of journal articles printed between 1965 and 1967".

A simple grammatical analysis of the phrase structure type is used to identify the various relations included in the search statement, as well as the entities being operated upon. This transforms each query into a "function and parameter" type query, except that the number of permissible relations and parameters is relatively large. The meaning of each function word and each entity (author name, journal, English language, editor, article, etc.) is unambiguous, and the transformation into a usable search program is relatively straightforward.

The last type of query language to be considered uses the unrestricted natural language in a specified subject area applicable to the search system (for example, facts related to the baseball season in a given year, or facts about naval personnel and ships in a given area; or information about political subdivisions in certain states, and so on). If the natural language is to serve in a relatively unrestricted form, then syntactic ambiguities, as well as semantic problems arising from the use of synonyms, homographs, and various undefined entities must be resolved before a translation becomes possible to unambiguous search statements which can actually be processed.

Several possible methods exist for the analysis of natural language queries. First, it is theoretically possible, particularly in a restricted topic area, to undertake a formal reduction of an input query into unambiguous statements. Such a process normally involves three different steps:

a) a dictionary look-up process designed to attach to each word of the input query a set of syntactic and semantic markers indicating the possible word functions;

b) a syntactic analysis process which uses the syntactic markers attached to the input text together with a set of phrase structures and/or transformational grammar rules to produce a syntactic tree form for each input statement, exhibiting the syntactic relationship between words and phrases;

c) a sematic interpretation method which uses a set of semantic transformation rules together with the semantic class markers and the syntactic phrase structure to transform the input into a set of operational statements to be processed by the search system.

The present state of the art in linguistic analysis does not make it possible to provide a unique semantic (or syntactic) interpretation for natural language statements in an unrestricted topic environment. Nevertheless, the trend in the direction of greater freedom in query formulation appears to grow, motivated in part by the fact that users find it easier to communicate in a natural rather than an artificial language, and in part because shortcuts are possible in conducting a natural language query analysis.

First, it appears that a requestor's "performance" grammar (that is, the grammatical framework which is actually used to express search requests) is not as rich as the total grammar available in a language. In fact, certain patterns are used repeatedly in query formulation, and these patterns can be recognized by special techniques. Second, the user is often personally available when a query is submitted, and an opportunity arises for the system to request clarifications of the user's intent if difficulties should arise during the analysis.

The problem of query pattern recognition is often solved by a type of template analysis basd on the occurrence of special key patterns in a query formulation. [14,15] Specifically, if key patterns such as "between — and — ", "written by", etc, occur in certain specified contexts within a query formulation, then one or more action routines are called which bypass the normal analysis and produce a translation or interpretation derived only from the input template and the context. It has been found that in a document retrieval environment, date-, journal-, and author phrases incorporated into normal document retrieval queries can be recognized by template analysis alone. [16]

The other possibility which consists in utilizing user feedback techniques to resolve difficulties arising during query analysis has also been tried out experimentally. In that instance, the user is requested to help by providing clarifications under the following circumstances: [17]

a) a word included in a query cannot be found during dictionary look-up, and requires definition;

b) an undefined grammatical phrase is used in the query formulation which needs to be rephrased;

c) an undefined meaning association is detected during the semantic interpretation which must be resolved.

In addition, the user can always reject one or more of the formally constructed query interpretations, in which case new queries are constructed from the responses provided to the system by the user. User-system interaction techniques are described in more detail in section 5 of this study.

4. Basic File Organization and File Search

A) Inverted File

Two principal file organizations are presently used for operational retrieval situations, known respectively as the <u>direct</u> and <u>inverted</u> systems. In the <u>direct</u> system, the file is stored in order by document reference number, and items to be retrieved are identified by a sequential scan of the complete file. Obviously, the direct file system cannot be used if rapid responses are to be made available to waiting customers.

The inverted system on the other hand, is based on an arrangement of the information file in order by the main information identifiers, normally a set of keywords or index terms, or alternatively a set of author names, or possibly journal titles. Thus, all documents indexed by term A are listed together, followed by those identified by B, C, D. and so on. Each information item is then included as many times as there are assigned keywords. To retrieve information stored in an inverted file, it is necessary to extract form the file the sections corresponding to the terms used to formulate the search request. Thereafter, the document references listed under the various index terms must be examined to see whether the search logic is satisfied. The operational systems which currently provide fast responses to incoming queries are based, without major exception, on inverted file organizations.

Under the inverted file organization, search operations are often performed automatically (even if the analysis and indexing should be done manually) using up to four distinct files as follows:

a) a term index, or directory which stores the list of terms together with a pointer for each term giving the appropriate starting address in the inverted term file;

b) an inverted term file, storing lists of terms, and for each term the list of applicable document references with weights;

c) a document index storing lists of document references together with a pointer for each reference giving the appropriate address for the document information in the document file;

d) a <u>document file</u> storing references together with appropriate document information including citations, abstracts, etc.

A typical search would then use the term index to obtain the starting addresses in the inverted term file of the document references corresponding to the individual query terms. The index may be stored in fast storage, whereas the inverted file of document references is stored on disk, or other similar medium. Once the individual document numbers responding to the query statement are known, the document index is consulted to obtain pointers to the addresses of the corresponding information in the document file. The total number of disk accesses to be performed is then equal to the number of terms included in the search request (to obtain the corresponding document references from the inverted file), plus the number of documents actually displayed for the user (to obtain the document information).

The main advantage of the inverted file system is provided by the rapid query-document correlation process. It is generally necessary to examine only the few lists of document references corresponding to the query formulation, and documents which do not match the search logic are never retrieved from the file. The disadvantages are in part those afflicting any organized file, in that the file updating problems are severe; in addition, only a small number of usable terms is normally provided — the more terms apply to a given document, the more often must the corresponding document references be listed in the inverted file. Thus, the search time depends on the length of the query statement, and on the type of query alteration taking place during the search process.

Since the query updating process which results from various search negotiation methods is a principal advantage of on-line retrieval systems, an inverted file system is not necessarily optimal in an interactive environment.

B) Clustered File Organization

It is well known that automatic "clustering" techniques exist for generating from an indexed document collection groups, or clusters of related terms, or documents. This is done normally by comparing the lists of document indentifications attached to the various documents (or, alternatively, the document assignments corresponding to the terms), and grouping those documents (or terms) whose identifying lists, or vectors, are sufficiently similar. [18,19] For each class, a description, or centroid vector, may be constructed to represent the class, consisting of an aggregate of the descriptions of the documents (or terms) included in that class.

In a clustered file organization, consisting of sets of prossibly overlapping document clusters, three types of files are normally used:

a) the class description file which stores for each class the centroid vector (including terms and weights) as well as a list of document references for the documents included in that class;

b) the class index, or directory, which stores for each class the starting address of the block of corresponding document vectors (terms and weights assigned to documents);

c) the document file storing document term vectors and appropriate document information in class order.

A standard file search then takes place by initially comparing the incoming queries with the centroid vectors only. Individual document descriptions are next examined, and documents similar to the queries are retrieved in decreasing similarity order for those documents whose class descriptions were earlier found to be similar to the query description. Typically the documents included in only two or three clusters might be examined. The examination of the document vectors included in certain classes is obviously simplified if the document file is stored in class order, of it the document descriptions within a single class are chained together by appropriate pointers.

The advantages of the clustered organization stem from the fact that the number of file accesses, proportional to the number of document classes examined, does not depend on the number of terms present in the query or in the document. Thus, no disadvantage results if large numbers of terms or concepts are assigned to queries and documents, as is the case for many automatic indexing methods, and the search time may be expected to be much smaller for average-length queries with a clustered file than with the normal inverted organization. Furthermore, the interactive search negotiation procedures, consisting normally of alterations of the query formulations by addition (or subtraction) of terms derived from various sources, do not require special access to the document file (which is not stored in term order). Thus the search negotiation process will be considerably simplified with a clustered organization.

The disadvantages of the clustered organization have to do largely with the document grouping operation itself. The standard grouping methods require of the order of n^2 vector comparison operations, where n is the number of documents (or terms) being grouped, and even the fast clustering methods which use certain shortcuts in the construction of the classes will still require of the order of $n \cdot m$ comparisons, where m is the number of clasees to be constructed. [20] A compromise must be made in practice by clustering a given document collection only once, and limiting the class updating and reorganization operations to relatively infrequent time periods.

In addition to the cluster construction problem, a clustered file organization will likely require slightly more storage space than an inverted file, because it now becomes necessary to store the class descriptions (centroids) in addition to the document descriptions. The storage requirements for a typical document file of 100,000 documents, identified by 3000 distinct terms are outlined in Table 1. It is seen from the table that approximately 3000 or 4000 disk tracks are required in the clustered case, depending on whether small or large document clusters are used, compared with 2600 tracks for the standard inverted file. The difference is not sufficiently large to render the clustered organization infeasible for normal operational use in an on-line environment.

C) Basic File Search

The basic file search process, which does not necessarily involve any extensive search negotiation, is simple to describe for both inverted and clustered file organizations. In both cases a two-level search is required: for the inverted files, a set of document references is first isolated from the inverted term file, and any additional bibliographic constraints are then verified for the chosen documents only in the document file; in the clustered case, the centroid vectors must be searched first in the class description file, followed by a search of certain document clusters in the document file.

For an inverted file organization, the detailed search steps are as follows:

a) the query terms are looked-up in dictionaries and authority lists, and are replaced if necessary by authorized terms only;

b) the originally available terms may be expanded by new, related terms obtained from the authority lists;

c) the terms are used to extract blocks of document references from the inverted file, and these document blocks are appropriately combined in accordance with the specified search logic;

d) document information is extracted from the bibliographic document file for those documents whose reference numbers were earlier identified in the inverted file;

e) bibliographic restrictions, such as publication dates, author or journal names, etc., are verified for the specified items in the document file;

f) positional restrictions, if any, of the terms are verified in the document file to insure, for example, that certain specified terms appear in the same document sentence, or within a certain distance from each other;

g) document information is printed out and shown to the user for those documents obeying all stated restrictions.

In a clustered organization, no inverted file is involved, and both search logic as well as bibliographic and positional restrictions are verified in one operation in the document file. The remaining operations including dictionary look-up, term expansion, and verification of restrictions are comparable to the inverted file case. Among the term specifications which have been implemented in various operational situations are the use of word stems instead of complete words to identify a term; the addition for each term of synonyms, or related words, or both, obtained from dictionaries or word lists; the use of weighted terms and verification of various weight combinations among sets of terms; the verification of positional criteria for sets of terms extracted from document texts or abstracts (occurrence in the same sentence, or paragraph, or in certain phrase arrangements).

The verification of search logic and search restrictions is normally performed in one of two possible ways:

a) one element at a time, that is, by verifying the various conditions for each document which meets the initial selection criteria at the time the document is first isolated;

b) as a sequence of set operations, that is, by first isolating a complete set of documents, and using this to create a smaller set obeying certain language or author restrictions, and going from this to a still smaller set, and so on.

For the first approach, a single complex search program is required, while for the second, the search program is much simpler, but problems may arise with the storage management for the intermediate sets. Which alternative is used depends on the available storage organization (that is, whether all the required files are available in internal storage, or whether separate disk accesses are needed) and on the complexity of the search formulation. For involved search specifications the "sequence of sets" procedure will normally become mandatory.

5. Search Negotiation Methods

Most of the console-based, on-line retrieval systems make available procedures for displaying dictionary information relating to potential search terms in an effort to bridge the gap between the language used by the document author or indexer, and that used by the searcher. [5,6,7,10, 11,13] Often the search negotiation is limited to the display and manipulation of term dictionaries, and the document information is not required unitl a final retrieval operation is eventually carried out. Thus the negotiation process may utilize only information already available in internal storage without touching the main document file. Alternatively, a set of documents may be retrieved in answer to an initial search formulation, and information derived from these documents may be added to the dictionary information in order to construct a new query. This latter mode is more complex since it necessitates calls to the bibliographic files in addition to the term lists.

A typical search negotiation involving dictionary display only may commence by having the user type in certain possible search terms. The system then responds by supplying the frequency of the documents indexed by the suggested terms, thereby indicating to the user an order of magnitude for the output set obtainable from a search using the given terms. The system may also display related dictionary information in the form of synonym lists for each of the original terms, and lists of alphabetically related terms and phrases. The user then picks certain terms (which may be entered into a "save" area), and starts combining terms into term sets using Boolean connectives. For each term combination, the "hit" frequency may again be obtained, and explanations in the form of tutorial statements or term definitions may be supplied. When the user is eventually satisfied with the search formulation, the final search statement is released from the save area and the actual search is performed.

The related terms displayed for the user's attention, as well as the term frequencies are usually prestored in table form. Occasionally, such information may also be computed "on the fly" from the stored document collection by supplying, for example, statistical associations for terms jointly assigned to documents in the collection. [7] Such "on the fly" computations trade increased search times for reduced storage space, and make it unnecessary to construct term dictionaries.

The alternative procedure consists in using both dictionary-type information, as well as information derived from previously retrieved documents in an attempt to construct improved queries. In the latter case, one might typically display the titles of previously retrieved documents, or their abstracts, or the set of assigned index terms, and these would then be used to obtain additional useful terms. The two procedures can, of course be combined by alternating between document and dictionary displays.

One special type of search updating method based on document displays shifts the burden of query reformulation from the user to the system. Here the user supplies only relevance judgments stating how well he does, or does not, like the previously retrieved documents. The system then automatically updates the user query by adding terms contained in documents identified as relevant by the user, and contrariwise by subtracting terms contained in items identified as nonrelevant. This <u>relevance feedback</u> process then automatically produces a query which comes to lie closer to the set of relevant items, and further away from the set of nonrelevant ones. The updated query may then better represent the user's interests and needs than the original.

In a recent study, a number of interactive search negotiation techniques were evaluated using as criteria the amount of user effort (that is, time and effort spent at the console), computer effort (that is, search time), and retrieval effectiveness as measured by recall and precision.* [21] The various query updating methods incorporated in that study are listed in Table 2 including both dictionary display and document display procedures. The former include term weighting, thesaurus display, word frequency listing and source document display (a source document is one which is identified as relevant by the user before a given search of the file is performed; such a document can often be displayed directly, on the same basis as a stored dictionary). The latter include title and abstract displays of previously retrieved documents, as well as relevance feedback methods.

The results of the evaluation process are summarized in Table 3. The precision improvements listed in Table 3 for the interactive search methods are given as percentage improvements over and above a standard fully automatic word stem matching process which retrieves documents based on a match between word stems included in a query formulation and word stems included in document abstracts. The "automatic thesaurus" method uses a stored thesaurus to normalize query and document vocabularies without any user interaction. [8]

It is seen in Table 3, that system performance as measured by the precision improvements at fixed recall levels correlates rather well with computer costs and user effort. In fact, the best interactive methods from a performance viewpoint are the document display procedures. However, these methods are also the most expensive to perform, since two file search

*Recall and precision defined respectively as the proportion of relevant material actually retrieved, and the proportion of retrieved material actually relevant have been used extensively to measure the effectiveness of retrieval systems. The assumption is normally made that an ideal system would retrieve all and only relevant items, thus exhibiting perfect recall and precision values equal to 1.

operations are required — one prior to the interactive process and one following it. From the user's viewpoint, the less information is displayed, the easier will normally be the interactive process. The document display methods which exhibit information relating to previously retrieved documents are therefore relatively onerous in terms of user effort. The exception is the relevance feedback process where the user must only supply a relevance judgment. The cluster search methods are of course the least expensive in terms of computer time, and the user effort is no larger than for full search. However, the performance of partial (cluster) search methods in an interactive environment remains to be evaluated.

6. Query and Document Space Modifications

A) Query Space Modification

It was seen in the previous section that interactive search procedures can be used to update user queries in such a way as to retrieve more useful and less useless materials than the original query formulations. This suggests that an interactive retrieval environment may be created in which user feedback procedures are utilized slowly to modify both the query and the document identifications, so as to bring the document vectors closer to the query vectors to which they appear pertinent. Thus, new queries which may be similar to queries processed at some earlier time may produce better search output than these original queries, since the items which were earlier identified as relevant may now be more easily accessible.

Consider first the basic relevance feedback process. Here the stored identifications of documents previously seen by the user, and judged for relevance, are used automatically to adjust the queries in such a way as to promote terms present in the relevant documents while demoting terms present in the nonrelevant items. A typical query updating process may be represented by the following equation:

$$\underline{q}_{i+1} = \underline{q}_i + \alpha \sum_{i=1}^{n_r} \underline{r}_i - \beta \sum_{i=1}^{n_s} \underline{s}_i \qquad (1)$$

where $\underline{q}_{i+1}$ represents the updated query vector, $\underline{q}_i$ the original query vector, $\underline{n}_i$ is one of n_r document representations indentified as relevant, and $\underline{s}_i$ is one of n_s nonrelevant documents. [22,23]

Two major variants of the relevance feedback process must be discussed. The simpler method, known as <u>positive feedback</u> uses only the retrieved documents judged relevant to alter a given query ($\beta=0$ in feedback equation). The second variant uses both relevant and nonrelevant items to modify the query ($\beta>0$) and is known as <u>negative feedback</u>. A typical positive query alteration process is illustrated in the example of Table 4. An original query statement is given, as well as the analyzed query "vector" in the form of a weighted term list. Following the additional terms from document number 102, previously identified as relevant, the revised query vector retrieved two more relevant documents, numbers 80 and 81, with ranks 7 and 6, respectively (documents are always ranked in decreasing order of similarity with the query). These two documents were originally assigned ranks 14 and 137 using the original query.

Although positive feedback is often successful, it fails to aid the retrieval performance of some queries. This occurs notably when no relevant items are retrieved by the initial query, or when the retrieved items are dissimilar. Performance may thus be improved under these unfavorable conditions by negative strategy which moves the queries away from those items specifically not wanted by the user. An experimental comparison between positive and negative feedback made for a collection of 200 documents and 42 queries in aerodynamics reveals the following main differences in performance: [22,24]

a) the overall average performance differences between positive and negative feedback are not statistically significant;

b) however, the variance in the performance measure is greater for negative feedback than for positive, indicating that for some queries negative feedback is better and for other queries it is worse than positive feedback;

c) queries retrieving no relevant document in an initial search, for which the positive strategy cannot be used, are helped by the negative strategy;

d) the negative strategy alters the query vector much more than the positive strategy (the average correlation between initial and updated queries is about 0.85 for the positive, but only 0.50 for the negative strategy).

The fact that the negative strategy can help in some cases where the positive feedback is unusable, and does not on the whole exhibit a lower overall performance, suggests that a selectively applied negative policy might produce further improvements in overall performance. Under the procedure described by equation (1), <u>all</u> terms included in the identified set of nonrelevant documents are automatically deleted from the query or reduced in weight. Two selective negative procedures which preserve important query terms, even though these terms might also be included in one or more nonrelevant documents may thus be instituted as follows: [24]

a) the first selective strategy consists in assigning negative weights to terms extracted from nonrelevant documents while leaving the original query terms unchanged;

b) the other selective negative process makes use of a stored synonym dictionary, or thesaurus, to provide for each query term a set of related terms; these related terms are first added to the query statement, after which the terms obtained from the nonrelevant documents are subtracted out.

Another feedback strategy designed to control to some extent the strength of alteration of the query vector, and to prevent the promotion of some relevant documents while at the same time demoting certain others, consists in combining the standard cluster search with the query alteration process. Specifically, a query alteration procedure is utilized in which retrieved documents from separate clusters generate distinct queries, each of which operates within a distinct document cluster. Such a cluster feedback process would then operate as follows:

a) the original query is first compared with the centers (centroids) of all document clusters;

b) the clusters whose centroids are closest to the original query are then selected, and the individual document vectors within the selected clusters are compared to the query;

c) relevance judgments are obtained for those documents found to be closest to the query;

d) a new query is constructed for each cluster, using the original query as well as relevant (or nonrelevant) documents from that particular cluster only;

e) each new query is now matched only against the documents in its own cluster, and only the documents retrieved by a given query are used to modify that query in further feedback iterations.

The cluster feedback process then effectively generates from each original query several subqueries, each of which is designed to reflect the subject area represented by one of the document clusters. Several such query "splitting" methods have been investigated with some success, using small test collections. [25]

The cluster feedback algorithm outlined above is also feasible in connection with a request clustering method. This possible alternative to the normal document clustering is suggested by the fact that documents formerly retrieved in answer to similar, previously received queries should be considered in processing a new query. The cluster feedback algorithm using request clusters would then be changed by comparing the new query to the centroids of clusters of previously submitted queries (See step (a)). The clusters of documents searched in steps (b), (c), (d), and (e) would include all documents judged relevant to the queries located in the query clusters closest to the new query. If the user population is fairly homogeneous and many similar queries are produced within a given subject area, then a feedback method making use of request clusters may be expected to be more productive than one operating within the standard document clusters.

B) Feedback Evaluation

Consider the effect of a standard feedback operation, reflected for a sample query by the output of Table 5. The table contains the ranks of retreived documents for an initial search, as well as the ranks after each of three feedback operations. Relevant documents are marked 'R', and the user is initially shown the 15 top documents. It is clear from the example of Table 5, that the improvement in relevant document ranks from one search to the next (which will of course be reflected in recall and precision improvements) is due to two entirely different circumstances:

a) the improvement in rank of documents already seen by the user and previously identified as relevant (document 94 of Table 5);

b) the improvement in rank of relevant documents not previously seen by the user (documents 90, 95 and 91 of Table 5).

Since the user is not necessarily interested in reviewing several times a document which had already been presented to him during previous search iterations, it seems that these two cases ought to be distinguished by removing from consideration during subsequent iterations documents previously seen by the user. In other words, the evaluation of each feedback iteration ought to be based on a constant amount of user effort (assuming that the same number of documents are presented each time), recall and precision should not be made to depend on increases (or decreases) in the ranks of previously retrieved items.

Several procedures for a feedback evaluation independent of user effort exist. One of the most appealing is the test and control methodology in which an attempt is made to separate the document collection into two equal parts, while maintaining identical collection properties (number of documents, number of relevant items per query, etc.). [22,26] The feedback process is first performed with one-half of the collection, the test collection, and the evaluation process is performed later with the second collection, the control collection, which had not previously been used to modify the queries. The process is described in the flowchart of Fig. 1.

The results of a comparison between initial query processing using the control collection, and "test collection modified" query processing using the control collection are shown in the graph of Fig. 2. The results of Fig. 2 are averaged over 153 queries and 424 documents in aerodynamics, the test and control collections consisting of 212 documents each. Average recall values for the 153 queries and plotted against average precision values, and the curve closest to the upper right-hand corner of the graph, where both recall and precision are equal to 1, reflects the best performance. The initial run output of Fig. 2 shows that the test and control collections used in the evaluation process for this example, were not entirely comparable — the performance curve for the control collection being superior to that of the test collection. As a result, the differences due to feedback measured by the cross-hatched area of Fig. 2 may not be entirely reliable. Still, the test and control evaluation appears appropriate if care is taken in the construction of the subcollections. The first iteration control curve, superimposed on the graph of Fig. 2 corresponds to the standard evaluation process, illustrated by the example of Table 5, which does not distinguish between feedback and ranking effects for previously retrieved documents.

No matter which evaluation method is actually used, the query alteration process is seen to produce precision improvements of from 5 to 10 percent at most recall levels for the sample collection of Fig. 2.

C) Document Space Modification

The feedback procedures described up to now all produce a modification of the query space in such a way that queries are moved close to certain identified relevant documents, or away from identified nonrelevant ones. The problem may however also be attacked directly by permanently changing the document vector space. Specifically, the term vector representations of documents judged relevant to a query may be moved closer to the query vector, and contrariwise nonrelevant document vectors may be moved further away. These strategies are more radical than query modifications, since their use implies that queries are more fundamental as subject indicators than the original document identifying terms.

The two main document space modification techniques are illustrated in the example of Fig. 3. In the first one (Fig. 3(a)), the previously identified relevant documents are modified by addition of query terms as follows: [27,28]

i) $\hat{d}_n^i = \beta$

ii) $\hat{d}_n^i = d_n^i + \gamma(120 - d_n^i)$

iii) $\hat{d}_n^i = d_n^i - [(\hat{d}_{n/\delta}^i) + 1]$

where $\hat{d}_n^i$ is the i^{th} term in the new vector representation of document d_n, and β,γ,δ are appropriate constants. Formulas i), ii), and iii) are used according as terms i is present only in the original query vector, is present both in the query vector and in document d_n, or is present only in document d_n.

In the negative strategy of Fig. 3(b), one nonrelevant document is modified by moving it away from the query for each relevant item moved toward the query. The positive modification is performed in accordance with formulas i), ii), and iii); the modification of nonrelevant items is done by formulas iv) or v) according as term i is present in both the original query and the nonrelevant item d_m, or only in d_m as follows:

iv) $\hat{d}_m^i = d_m^i - [(d_m^i/\theta + 1)]$

v) $\hat{d}^i_m = d^i_m + \varepsilon(120 - d^i_m)$.

The output of Fig. 4 shows the improvements in recall and precision obtainable with a modified document space. [28] The collection used for experimental purposes is the same as that used earlier for the query modification process of Fig. 2, consisting of 425 documents and 155 queries. The complete testing process may be described as follows:

a) a control set of 30 queries is chosen from among the total set of 155 available queries; the test set is then taken as the 125 queries not included in the control set;

b) experimental values are chosen for the constants β, γ, δ, ε, and θ;

c) a modified document space is created by modifying all relevant items retrieved (and possibly an equal number of nonrelevant items) for each query in the test set;

d) a full search is now run for the 30 control queries which had not been used in the space modification; recall-precision values are computed for the control query performance for both the original and the modified spaces.

The output of Fig. 4 thus represents average performance over the 30 control queries.

Since both the query and document space modifications are dependent on the user population rendering the relevance judgments, these techniques must be implemented in an operational environment with real users before the improvements in retrieval effectiveness produced can be accurately assessed.

7. Information Networks

In addition to the experimental work now performed with on-line consoles within individual information centers or laboratories, a number of experimental networks of information centers already exist in which a central information store is accessible from several different locations using input-output consoles tied to suitable transmission equipment. One of the best known is in the field of medicine consisting of about a dozen medical libraries jointly connected to a large medical information store. [29] The system provides searches of a combined library catalog by subject terms, author, title, year of publication, language of document, and so on, and operates with inverted files as outlined in section 4(A) of this study. A computerized union catalog of the combined holdings can be produced, in addition to various types of indexes, authority lists, recurring bibliographies, and other by-products.

This type of mechanized information network will take on increasing importance as the coverage of the stored collection becomes wider, and the facilities to obtain access become more convenient and less expensive.

As the research and development of new information handling devices continues, more attention is also being paid to the social problems connected with large information stores. A number of studies have been made of the problems arising with the copyright and patent protection of the stored, and easily reproducible information. The greatest concern, has however, been voiced in connection with the protection of privacy of stored information. [30]

While large information networks provide useful information sharing in many areas, and often permit a rapid fulfillment of individual user requirements (as in automatic ticket reservation systems, or automatic credit certification), adequate safeguards for the protection of sensitive information are not generally provided. Before the mechanized information networks come into more widespread use, it is then necessary to build protective devices which would prevent unauthorized access. Some methods have been developed for this purpose, including special accessing techniques based on user passwords before access is obtained, reversible encoding techniques to conceal stored information, monitoring techniques to keep track of each file access, and processing restrictions limiting access of certain consoles to only certain files. Additional legal safeguards will undoubtedly be created before long to insure fair use of stored information files.

References

[1] W. H. Desmonde, A Conversational Graphic Data Processing System, The IBM 1130/2250, Prentice Hall, 1969.

[2] K. J. Engvold and J. L. Hughes, A General Purpose Display Processing and Tutorial System, Communications of the ACM, Vol. 11, No. 10, October 1968.

[3] J. K. Summers and E. Bennett, Aesop — A Final Report: A Prototype On-line Interactive Information System Science and Technology, D. E. Walker, editor, Thompson Book Co., Washington, 1967.

[4] A. Van Darn, Hypertext Editing System, Report, Brown University. 1967.

[5] R. S. Marcus, P. Kugel, and R. L. Kusik, An Experimental Computer-stored Augmented Catalog of Professional Literature, Proceedings AFIPS Spring Joint Computer Conference, Thompson Book Co., Washington, May 1969.

[6] J. L. Bennett, On-line Computer Aids for the Indexer, Proceedings of the ASIS National Meeting, Columbus, August 1968.

[7] R. M. Curtice and P. E. Jones, An Operational Interactive Retrieval System, A. D. Little Co., Cambridge, June 1969.

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

[9] S. F. Smith and R. M. Shoffner, A Comparative Study of Mechanized Search Languages, Institute of Library Research, University of California, Berkeley, January 1969.

[10] E. B. Parker, Spires — Stanford Public Information Retrieval System, 1968 Annual Report, Institute for Communication Research, Stanford, January 1969.

[11] R. K. Summit, Dialog — An Operational On-line Reference Retrieval System, Proceedings ACM National Meeting, Thompson Book Co., Washington 1967, p. 51-56.

[12] M. E. Maron and R. L. Levien, Relational Data File: Design Philosophy and Implementation, in Information Retrieval — A Critical View, G. Schecter, editor, Thompson Book Co., Washington, 1967.

[13] W. D. Mathews, TIP Reference Manual, TIP Technical Memorandum 104, M.I.T., Cambridge, Massachusetts, August 1968.

[14] J. Weizenbaum, Eliza — A Computer Program for the Study of Natural Language Communication between Man and Machine, Communications of the ACM, Vol. 9, No. 1, January 1966.

References (contd)

[15] J. Weizenbaum, Contextual Understanding by Computers, Communications of the ACM, Vol. 10, No. 8, August 1967.

[16] S. F. Weiss, Template Analysis and its Applications to Natural Language Processing, Scientific Report ISR-16 to the National Science Foundation, Department of Computer Science, Cornell University, 1969.

[17] C. H. Kellogg, On-Line Translation of Natural Language Questions into Artificial Language Queries, Information Storage and Retrieval, Vol. 4, p. 287-307, 1968.

[18] H. Borko and M. D. Bernick, Automatic Document Classification, Journal of the ACM, Vol. 10, No. 2, April 1963.

[19] R. M. Needham and K. Sparck Jones, Keywords and Clumps, Journal of Documentation, Vol. 20, No. 1, March 1964.

[20] R. T. Dattola, Experiments with a Fast Algorithm for Automatic Classification, Scientific Report No. ISR-16 to the National Science Foundation, Department of Computer Science, Cornell University, September 1969.

[21] M. E. Lesk and G. Salton, Interactive Search and Retrieval Methods Using Automatic Information Displays, Proceedings of AFIPS Spring Joint Computer Conference, Thompson Book Co., Washington, May 1969, p. 435-446.

[22] E. Ide, Relevance Feedback in an Automatic Document Retrieval System, Report No. ISR-15 to the National Science Foundation, Department of Computer Science, Cornell University, January 1969.

[23] G. Salton, Search and Retrieval Experiments in Real-Time Information Retrieval, Proceedings IFIP Congress 68, Edinburgh, August 1968.

[24] E. Ide and G. Salton, Interactive Search Strategies and Dynamic File Organization in Information Retrieval, Proceedings of ASIS National Meeting, San Francisco, October 1969.

[25] A. Borodin, L. Kerr, and F. Lewis, Query Splitting in Relevance Feedback Methods, Report No. ISR-14 to the National Science Foundation, Section XII, Department of Computer Science, Cornell University, October 1968.

[26] C. Cirillo, Y. K. Chang, and J. Razon, Evaluation of Feedback Retrieval using Modified Freezing, Residual Collection, and Test and Control Groups, Report No. ISR-16 to the National Science Foundation, Department of Computer Science, Cornell University, September 1969.

References (contd)

[27] T. L. Brauen, R. C. Holt, and T. R. Wilcox, Document Indexing Based on Relevance Feedback, Scientific Report No. ISR-14 to the National Science Foundation, Section XI, Department of Computer Science, Cornell University, October 1968.

[28] T. L. Brauen, Document Vector Modification in On-line Information Retrieval Systems, Report No. ISR-17 to the National Science Foundation, Department of Computer Science, Cornell University, September 1969.

[29] I. H. Pizer, A Regional Medical Library Network, Bulletin of the Medical Library Association, Vol. 57, No. 2, April 1969, p. 101-115.

[30] L. J. Hoffman, Computers and Privacy: A Survey, Computing Surveys, Vol. 1, No. 2, January 1969.

System	Assumptions	Number of Disk Tracks
	100,000 documents - 3000 terms Storage Requirements: term identifier 2 bytes term or document weight 2 bytes document identifier 4 bytes document citation 80 bytes track capacity 7294 bytes	
Inverted File	600 document references per term 2 term groups per track Bibliographic information at 50 documents per track	1500 1100 2600
Small Clusters	20 terms per document 20 documents per cluster 5000 clusters 140 terms per cluster centroid 2 clusters per disk track centroid storage document storage	 455 2500 2955
Large Clusters	40 terms per document 40 documents per cluster 2500 cluster centroids 250 terms per cluster 1.3 clusters per track centroid storage document storage	 414 3572 3986

Storage Requirements for Inverted and Clustered Files

Table 1

Query Alteration Process	Explanation
Dictionary Displays	
1. Repeated Concepts	User chooses query terms to be repeated for emphasis
2. Thesaurus Display	User chooses terms obtained from thesaurus display to update query (with or without time restrictions)
3. Word Frequency	User looks at display of word frequency information before updating query
4. Source Document	User looks at display of source document before updating
Document Displays	
5. Title Display	User looks at titles of first five retrieved documents before updating
6. Abstract Display	User looks at abstracts of first five retrieved documents
7. Relevance Feedback	Query is updated automatically using relevance judgments supplied by user following an initial search
Combined Methods	
8. Abstract plus Thesaurus	User looks at pre- and post-search information

Typical Query Updating Methods
(from Lesk and Salton [21])

Table 2

Processing Method	Computer Costs	User Effort	Precision Rise Over Word Stem	
			Low R	High R
A) <u>Fully Automatic</u>				
word stem match	normal	none	-	-
automatic thesaurus	normal	none	+4%	+6%
B) <u>Dictionary Display</u>				
thesaurus display	normal +	medium	+6%	+4%
source document display	normal +	medium	+8%	+5%
C) <u>Document Display</u>				
title display	high	medium	+13%	+2%
abstract display	high	very high	+17%	+5%
relevance feed-back	high	low	+10%	+7%
D) Partial Search				
cluster search	low	none	+5%	-10%
cluster search with relevance feedback	low +	low	?	?
cluster search with abstract display	medium	very high	?	?

Performance Summary for Interactive Methods

(from Lesk and Salton [21])

Table 3

Vector Type	Illustration
Initial Query Q 146	What information is available for dynamic response of airplanes to gusts or blasts in subsonic regime
Initial Query Vector	airplane 12, available 12, blast 12, dynamic 12, gust 12, information 12, regime 12, response 12, subsonic 12
Relevant Document 102 retrieved with rank 2 (partial vector)	gust 48, lift 48, oscillating 12, penetration 12, response 24, subsonic 12, sudden 12
Query Modified by by Document 102	airplane 12, available 12, blast 12, dynamic 12, gust 60, information 12, lift 48, oscillating 12, penetration 12, regime 12, response 36, subsonic 24, sudden 12
Relevant Document 80 (improves from rank 14 to rank 7) (partial vector)	gust 24, lift 72, penetration 12, sudden 12
Relevant Document 81 (improves from rank 137 to rank 6) (partial vector)	lift 84, oscillating 12, sudden 12

Positive Feedback Illustration
(from Ide and Salton [24])

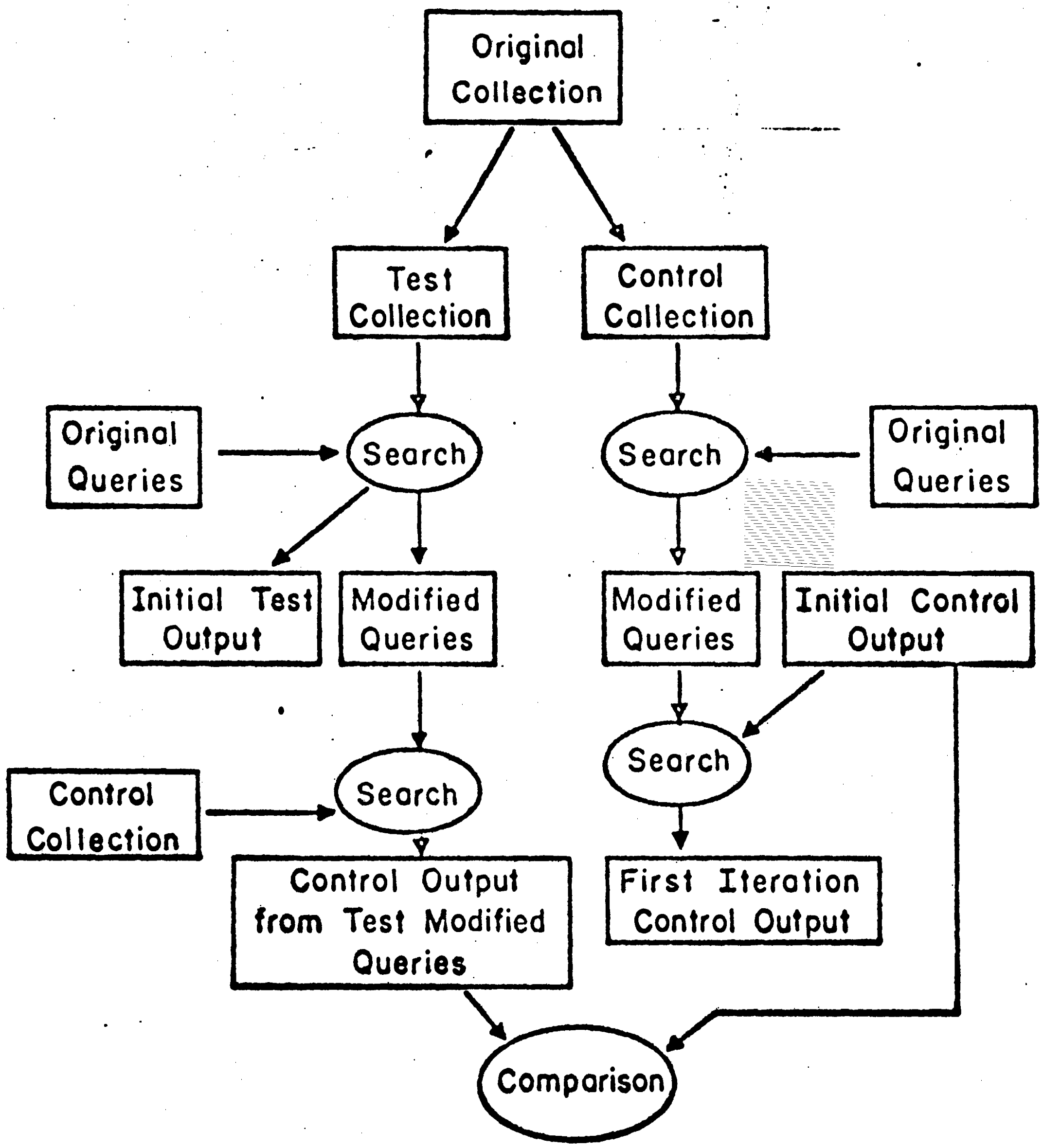
Table 4

Query Q 147: Will forward or apex located controls be effective at low subsonic speeds.

Initial		Feedback Iterations		
		1	2	3
Rank	Doc	Doc	Doc	Doc
1	109	94R	94R	94R
2	60	81	95R	95R
3	121	195	90R	90R
4	192	123	195	195
5	193	80	81	91R
6	119	114	80	81
7	82	90R	114	80
8	24	193	193	111
9	86	122	123	114
10	123	95R	111	193
11	100	111	91R	93R
12	146	64	109	123
13	18	102	159	192
14	94R	109	103	109
15	167	82	192	159
16	125	103	82	155
17	163	78	93R	103
18	114	125	78	82
19	65	20	122	78
20	177	192	102	110
21	93R	124	64	122
22	90R	159	155	153
23	19	194	110	11
24	153	196	11	76
25	181	86	153	64
26	58	63	76	92
27	22	66	196	102
28	172	91R	20	152
29	200	10	194	161
30	64	93R	132	132
31	3	11	152	196
32	195	61	92	96
33	144	77	125	29
34	122	76	124	133
35	63	132	86	194
36	184	104	161	20
37	34	153	61	86
38	74	54	133	104
39	113	49	29	61
40	17	177	104	125
41	95R	144	63	176
42	75	67	96	124
43	67	29	83	121
44	140	60	77	83
76	91R	19	160	160

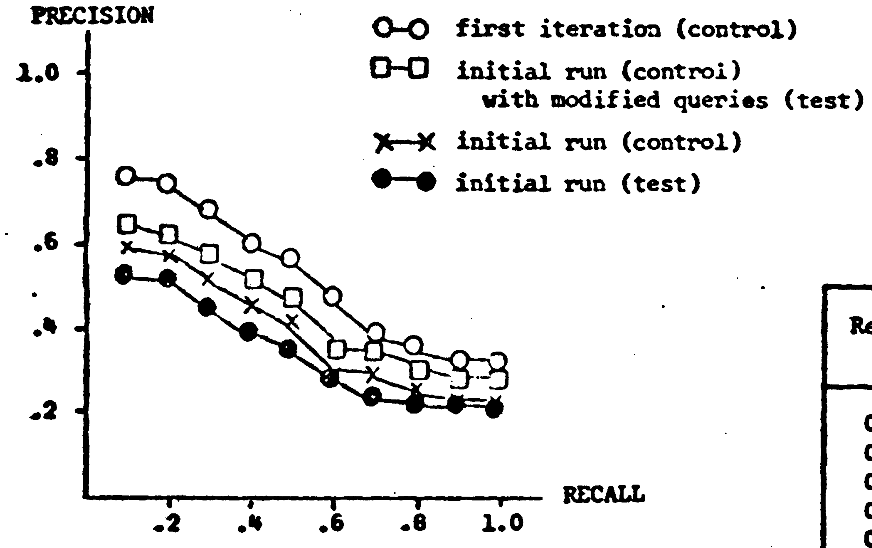
Positive Feedback Strategy for Query Q 147 Showing
Improvements in Relevant Document
Ranks
(from Ide and Salton [24])

Table 6



Test and Control Feedback Evaluation

Fig. 1

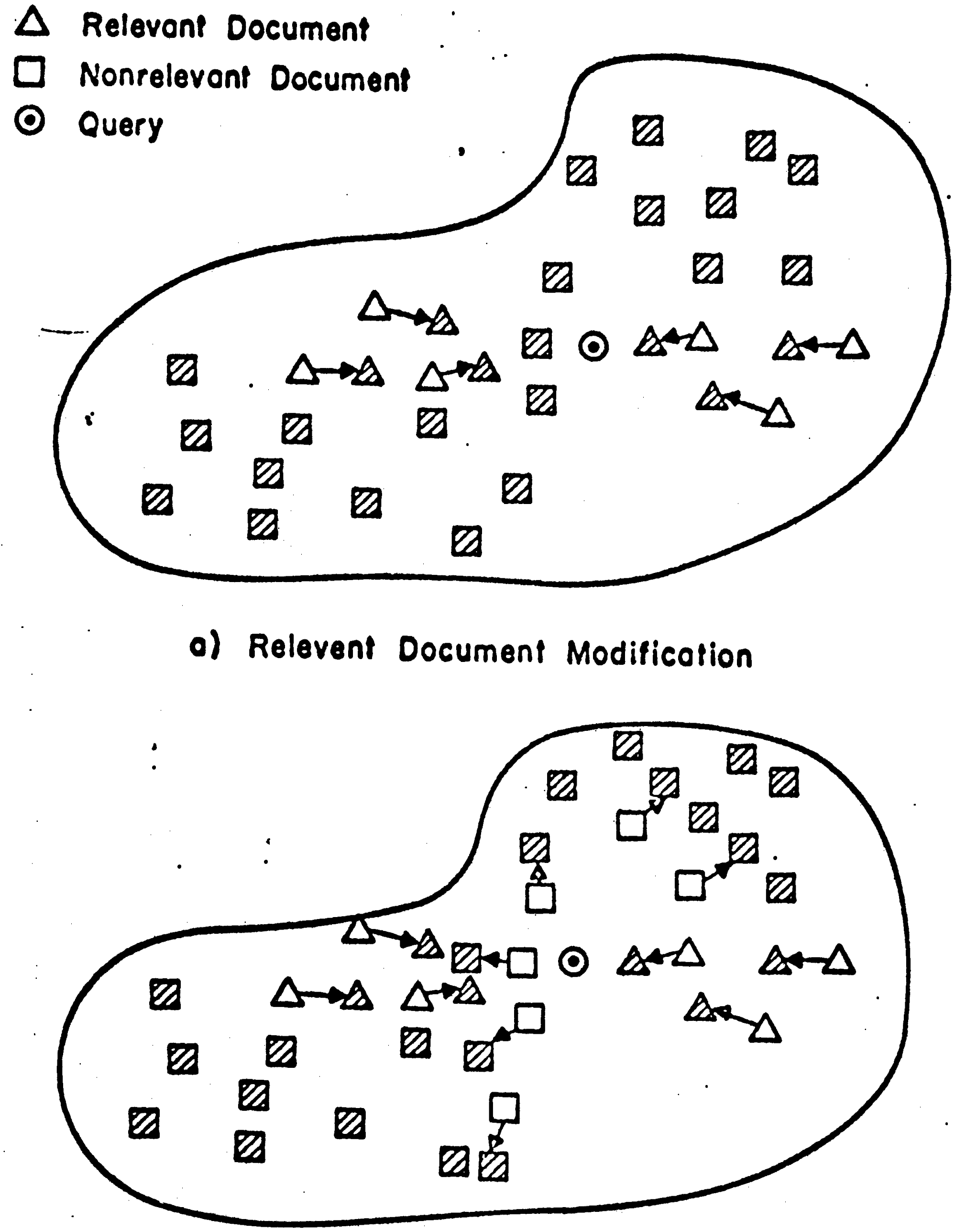


Recall	Precision ●—●	×—×	□—□	○—○
0.1	0.52	0.61	0.63	0.75
0.3	0.49	0.53	0.56	0.68
0.5	0.38	0.42	0.48	0.56
0.7	0.22	0.28	0.32	0.39
0.9	0.22	0.22	0.26	0.32

Test and Control Feedback Evaluation

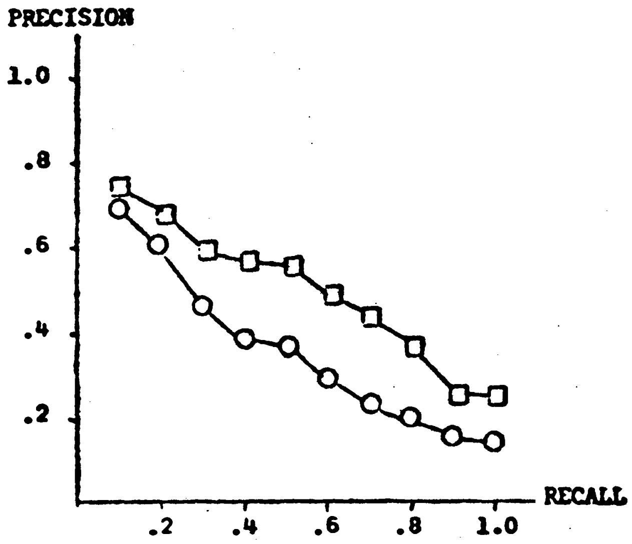
(averages for 153 queries and 424 documents in aerodynamics)

Fig. 2

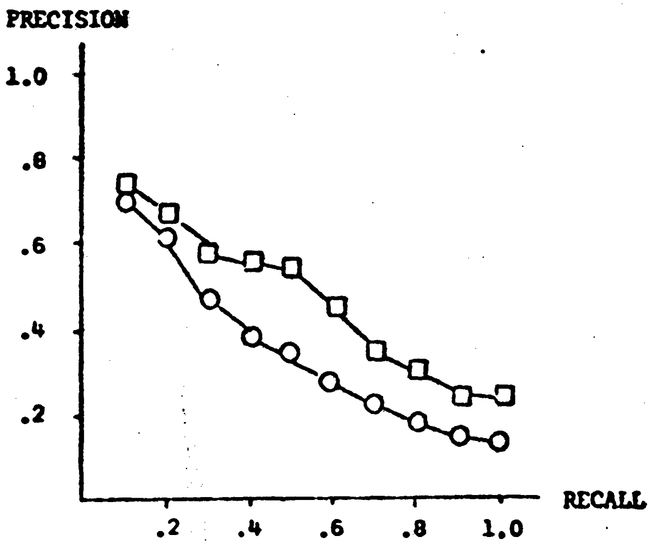


Document Space Modification
(from Ide and Salton [24])

Fig. 3



a) Relevant Document Modification



b) Relevant and Nonrelevant Document Modification

Document Space Modification

(30 control queries, 125 test queries, 424 documents)

Fig. 4