Advanced Feedback Methods in Information Retrieval

G. Salton[*]
E.A. Fox[†]
E. Voorhees[*]

TR 83-570
August 1983

Department of Computer Science
Cornell University
Ithaca, NY 14853

[*]Department of Computer Science, Cornell University, Ithaca, New York 14853.

[†]Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

This study was supported in part by the National Science Foundation under grants IST 81-08696 and IST 80-17589.

Advanced Feedback Methods in Information Retrieval

G. Salton[*], E.A. Fox[†], and E. Voorhees[*]

Summary

Automatic feedback methods may be used in on-line information retrieval to generate improved query statements based on information contained in previously retrieved documents. In this study automatic relevance feedback techniques are applied to Boolean query statements. The feedback operations are carried out using both the conventional Boolean logic, as well as an extended logic producing improved retrieval effectiveness. Experimental output is included to evaluate the automatic feedback operations.

[*] Department of Computer Science, Cornell University, Ithaca, New York 14853.
[†] Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061.

This study was supported in part by the National Science Foundation under grants IST 81-08696 and IST 80-17589.

1. Conventional Bibliographic Retrieval

The operations of most existing commercial bibliographic retrieval systems are based on the use of a dual file system: a main document file storing appropriate bibliographic information for each document in the collection, and a set of auxiliary indexes, known as an inverted file, which supplies for each content identifier, or term, assigned to a document, the list of corresponding document references. Normally, the terms representing document content are manually assigned, but in principle the individual words included in the document texts could also be used for document identification.

In operational retrieval, the search requests may also be formulated by using terms reflecting the user's information needs. However, whereas the document content is often expressed by sets of unrelated terms, the queries are usually represented by Boolean expressions, consisting of search terms interrelated by the Boolean operators *and*, *or*, and *not*. For example, a query such as

((information *and* retrieval) *or* (automatic *and* documentation))

is designed to retrieve items identified by the terms "information" as well as "retrieval", or else (or in addition) the terms "automatic" and "documentation". In a system using inverted files, the Boolean operations are normally implemented by list intersection and list union operations. Thus a Boolean "and" can be performed by a list merge followed by the recognition of duplicated items, and an *or* by a merge followed by elimination of one instance of each duplicated item. These operations are illustrated in Fig. 1 for two sample lists of document identifiers corresponding to the terms "information" and "retrieval". [1,2]

The popularity of the Boolean model is due not only to the relatively simple implementation by list merge operations, but more importantly to the possibility of constructing queries of the appropriate specificity for each individual user. Indeed, the _or_-operator is normally used to represent synonymous constructions, and adding _ored_ terms tends to broaden the query formulation. The _and_ operator generally relates phrase components, and using _and_ operations will render the query more specific. By judicious use of _and_, _or_, and _not_ one can sometimes obtain precisely the desired type of output.

In former times, a good deal of attention was paid to vocabulary control for document analysis (indexing) and query formulation purposes. That is, thesauruses and vocabulary control aids were provided specifying lists of allowable index terms and term relationships (synonyms, broader terms, narrower terms, related terms). The document indexing and query formulation tasks could then be highly demanding, since it became necessary to specify not only the individual terms, but also certain term properties or relationships (links and roles). Such input control were justified by citing examples of the retrieval failures that might occur in their absence--for example, that queries requesting information about "Venetian blinds" might retrieve documents dealing with "blind Venetians".

More recently, the emphasis has shifted to the use of controls at the search end rather than at indexing time, in part because it was realized that some of the examples of potential retrieval failures were fictitious, and more immediately because full-text, and other automatic, indexing systems were developed to replace the more complex, former indexing methods. Most important in assessing the usefulness of indexing controls is the fact that nearly all operational retrieval systems now provide _on-line_ search capabilities where the responses to the submitted search requests are obtained in real-time while the users are waiting for the search output. In such an environment, the users can submit tentative search formulations and consider the partial answers retrieved in response to these formulations before deciding to abandon or continue a search with either original or altered search requests. In general, a given complete search may then be carried out by several distinct partial searches in the hope of obtaining satisfactory search output for the combined search. [1,2]

In actuality, the promise of user-controlled on-line searching may not always be fulfilled because of the intricacies in the design of existing computer systems, and the need to learn console operations, command languages, and system operations. In these circumstances, trained search intermediaries may still be needed in the on-line environment to generate correct Boolean search expressions and to interpret system responses for the user. As a matter of practice, some controls are therefore still provided in existing operational on-line retrieval systems, both at the input end for document analysis and indexing purposes and at output time during the search operation.

Relevance feedback is a largely automatic process designed to simplify the interaction task for retrieval system users by providing methods for an automatic reformulation and improvement of originally available search queries, based on information about the usefulness of documents previously retrieved in earlier search steps. Alternatively, given information about the relevance to the user of certain items included in the stored collections, good initial queries can be automatically constructed prior to any search operation. Advanced relevance feedback operations are considered in the remainder of this note.

2. Automatic Relevance Feedback

Automatic relevance feedback methods were introduced in vector processing environments nearly twenty years ago. In a vector processing system both the queries Q, and documents D, are represented by sets of possibly weighted, independent terms, unrelated by Boolean operations or other relationship indications. An approximation to an optimal query vector may be generated by adding to an initial query formulation terms extracted from previously retrieved documents identified as relevant to the query, and analogously by subtracting from an initial query formulation terms extracted from previously retrieved nonrelevant documents. [3] Assuming that m_1 of the previously retrieved documents are identified as relevant (Rel), and m_2 as nonrelevant (Nonrel), an initial vector query Q_0 can then be updated using vector addition and subtraction as follows:

$$Q_1 = Q_0 + \frac{1}{m_1} \sum_{\text{Rel}} D^{(i)} - \frac{1}{m_2} \sum_{\text{Nonrel}} D^{(i)} \qquad (1)$$

The formulation of expression (1) adds new terms, or alters the weights of existing terms, in accordance with the term occurrences in the relevant and nonrelevant retrieved documents. The new query formulation Q_1 may be expected to exhibit greater similarity with the relevant items, and smaller similarity with the nonrelevant ones than the original formulation Q_0. The available experimental evidence indicates that substantial increases in retrieval effectiveness can be produced by the relevance feedback procedures in vector processing environments. [4-6]

Relevance feedback methods have also been used advantageously in probabilistic information retrieval. [7-9] In that case, an explicit feedback query is not constructed. Instead, a <u>relevance weight</u>, sometimes known as term relevance or term precision, is computed for the terms contained in previously retrieved relevant, or nonrelevant documents. The relevance weight of a given term x_t is determined as a function of the occurrence probabilities of that term in the relevant and nonrelevant documents. Correspondingly, a document relevance factor may be determined for each document as a function of the relevance weights of the individual terms contained in that document.

Assuming that the relevance factors have been computed for all documents, the items may then be ranked and retrieved in decreasing order according to their presumed importance to the user. The evidence indicates that the relevance feedback process is useful in probabilistic information retrieval, as it is in the vector processing system.

The foregoing operations are difficult to apply successfully in a Boolean query environment first because the choice of terms to be used for feedback purposes is more crucial in Boolean systems than in vector processing in view of the differentiated nature of the terms, but more importantly because in a Boolean environment the feedback process requires a choice not only of the feedback terms but also of the Boolean operators relating the terms. [10-12] Recently a Boolean relevance feedback process has been developed which appears to be more effective than previously proposed systems, and provides a viable basis for a practical Boolean feedback system. [13-15]

The new Boolean relevance feedback process, known as the DNF method, consists of two main processing steps, including first the construction of "good" term clauses, and then the generation of a Boolean query statement in disjunctive normal from (DNF) incorporating some of previously chosen clauses, and designed to retrieve approximately T documents. A clause is defined as a single term, or a conjunction of several terms connected by <u>and</u> operators, and T is a parameter specified by the user stating the desired number of documents to be retrieved by the feedback query. The actual use of a given clause in a feedback query depends on two main factors:

a) its _relevance weight_ which expresses the degree to which the clause is likely to be useful in retrieving relevant documents from the collection;

and its expected _postings frequency_ which represents the approximate number of documents which the clause may be expected to retrieve.

A reformulated feedback query then consists of a set of clauses interconnected by _or_-operators chosen in such a way that each clause carries a high relevance weight and that the total set of clauses will retrieve the desired total number T of documents.

Simplified charts of the clause generation and final query reformulation processes appear in Figs. 2 and 3. The chart of Fig. 2 covers the identification of a set of good single terms t (that is, terms with high relevance weight), as well as good _anded_ term pairs, and _anded_ term triples. For each _anded_ clause, an estimated postings frequency is generated, defined as n_t, $n_s \cdot n_t/N$, and $n_r \cdot n_s \cdot n_t/N^2$ for single terms t, _anded_ pairs st, and _anded_ triples rst, respectively, where n_t is the postings frequency of term t and N is the total number of documents in the collection. The computed estimated postings frequencies are exact under the assumption that all terms are assigned independently of each other to the documents of a collection.

The frequency counts for terms occurring in the original queries can be adjusted by using an auxiliary _query count_ (qcount) parameter designed to give extra weight to the original query terms. The qcount parameter specifies that the occurrence of a term in a query is equivalent to its occurrence in k relevant documents. For query terms, the actual number of term occurrences in the retrieved relevant documents is then augmented by the qcount. This adjustment can increase the frequency counts and thus the relevance weights of query terms relative to the weights of terms not occurring in the queries.

The chart of Fig. 3 outlines the successive refinement process used to construct Boolean query statements designed to retrieve the wanted number T of documents. The basic idea consists in starting with a broad query--for example, the set of all previously chosen single terms connected by *or*-operators--and comparing the estimated number of documents retrieved by such an initial query (estret) with the retrieval threshold T. As a first approximation, estret is computed as the sum of the postings frequencies of the individual terms in the query. (This computation is exact when the document sets corresponding to the individual terms are disjoint.) Since estret will initially be much larger than T, single terms are now successively deleted in increasing order of their relevance weights (that is, worst terms are deleted first), and replaced by *anded* term pairs that are not subsumed by the singles still present in a given Boolean statement. For example, following deletion of terms A and B, the more specific clause (A *and* B) can be added to the query. Following each addition and/or deletion, the estret parameter is adjusted. So long as estret exceeds T, shorter clauses are deleted and replaced by more specific longer ones, until eventually T ≥ estret > T/2 when the process stops. The final query is then formed by connecting the clauses still present with *or*-operators.

An example of the query refinement process is included in Table 1 for a sample term set of 3 singles, 3 pairs, and 1 triple. The estimated number of retrieved items decreases from 173 for the broadest query consisting of the set of three _ored_ singles to 0.2 for the narrowest formulation consisting of a single _anded_ clause including all three terms.

Like most other feedback procedures, the query reformulation process of Figs. 2 and 3 can be iterated several times in the sense that the initial queries are transformed into first iteration feedback queries using the set of initially retrieved items; the first iteration feedback queries can then be used to retrieve additional items to be used in generating second iteration feedback queries, and these in turn may be used for further query reformulation steps. The process may be adapted to collections of varying size because the T parameter specifying the desired number of retrieved documents effectively controls the clause formation process: for smaller collections, the queries will be composed of _anded_ term pairs or triples; as the collection size grows, each _anded_ clause may be expected to retrieve a larger number of documents; the final queries may then consist of _anded_ quadruples or quintuples for a given fixed value of T. Finally, the automatic query formulation process makes special provisions for increasing the relevance weights, and hence the importance in the reformulated queries, of the original query terms. An evaluation of the DNF query reformulation process is included at the end of this study.

3. Extended Boolean Retrieval

The conventional Boolean relevance feedback process can be implemented relatively easily to generate improved query formulations. The effectiveness of the conventional system is however limited by the drawbacks of the standard Boolean retrieval model:

a) The size of the output obtained in response to a given query is often difficult to control; depending on the postings frequency of the query terms and the actual term combinations used, a great deal of output could be obtained, or alternatively no output might be retrieved at all.

b) The output obtained in response to a query is not ranked in any order of presumed importance to the user; thus each retrieved item is assumed to be as important as any other retrieved item and a feedback system based on the use of a fixed number, m, of retrieved items must effectively utilize a random m items rather than the best possible m retrieved items.

c) No provisions are made for assigning importance factors, or weights, to the terms attached either to the documents or to the queries; thus all terms included in the documents and queries are assumed to carry equal importance.

d) Boolean query formulations may produce counterintuitive results: for example, in response to an <u>or</u>-query (A <u>or</u> B <u>or</u> .... <u>or</u> Z), a document containing only one query term is deemed to be just as important as an item containing all query terms; analogously, for an <u>and</u>-query (A <u>and</u> B <u>and</u> ... <u>and</u> Z), an item containing all but one of the query terms is assumed to be just as useless as an item containing none of the query terms at all.

Some of these disadvantages are of course absent from the vector processing system which allows term weighting and provides ranked output in decreasing order of the vector similarity between documents and queries. However the vector processing system does not accommodate structured queries, the query terms being considered independent of each other. Document weights and output ranking are also incorporated in various extensions of the classical Boolean system such as the _fuzzy set_ model. [16-18] The fuzzy set system suffers, however, from lack of discrimination among the retrieved output nearly to the same extent as the conventional Boolean retrieval system, since the rank of a retrieved item depends only on the lowest, or highest weighted document term for _and_- and _or_-queries, respectively. In addition, it is difficult to extend the fuzzy set model to situations where term weights are attached also to the query terms instead of only to the document terms.

A new _extended Boolean_ retrieval system was recently introduced in which the value of a document for retrieval purposes is determined by a similarity computation between a Boolean query statement and a set of terms representing document content. The query-document similarity defined in the extended system is based on Lp vector norm computations, and is controlled by a variable parameter p, $1 \leq p \leq \infty$, providing a special interpretation for the Boolean operators. In particular when the value of p is large, that is, close to infinity, the classical Boolean system is maintained with its strict interpretation of the Boolean operators. As the value of p is reduced, the interpretation of the Boolean operators is relaxed more and more, to the point where the distinction between the Boolean _or_ and _and_ operators is lost completely when p reaches its lower limit of 1. The extended Boolean system represents a unifying retrieval model which includes the conventional Boolean system, the fuzzy set model, and the vector processing system as special cases, and maintains the advantages of both the vector processing (term weighting and document ranking) and of the Boolean operations (term relationships). [19-20]

Consider a set of terms $A_1,A_2,\ldots,A_n$, and let d_{A_i} represent the weight of term A_i in some document $D = (d_{A_1},d_{A_2},\ldots,d_{A_n})$, $0 \le d_{A_i} \le 1$. A generalized <u>or</u>-query, $Q_{\underline{or}}$, can now be formulated as

$$Q_{\underline{or}}(p) = [(A_1, a_1)\ \underline{or}^p (A_2, a_2)\ \underline{or}^p \cdots \underline{or}^p (A_n,a_n)]$$

where a_i specifies the weight of query term A_i, $0 \le a_i$ and $1 \le p \le \infty$. Similarly a generalized <u>and</u>-query, $Q_{\underline{and}}$, is written as

$$Q_{\underline{and}}(p) = [(A_1,a_1)\underline{and}^p(A_2,a_2)\underline{and}^p \cdots \underline{and}^p(A_n,a_n)]$$

The similarity between the document $D = (d_{A_1},d_{A_2},\ldots,d_{A_n})$ and the queries $Q_{\underline{or}}(p)$ and $Q_{\underline{and}}(p)$ may now be defined as

$$\mathrm{sim}(D,Q_{\underline{or}}(p)) = \left[\frac{a_1^p d_{A_1}^p + a_1^p d_{A_2}^p + \ldots + a_n^p d_{A_n}^p}{a_1^p + a_2^p + \ldots + a_n^p}\right]^{1/p} \tag{2}$$

and

$$\mathrm{sim}(D,Q_{\underline{and}}(p)) = 1 - \left[\frac{a_1^p(1-d_{A_1})^p + a_2^p(1-d_{A_2})^p + \ldots + a_n^p(1-d_{A_n})^p}{a_1^p + a_2^p + \ldots + a_n^p}\right]^{1/p} \tag{3}$$

It is not difficult to show that when $p=1$, $\mathrm{sim}(D,Q_{\underline{or}(p)}) = \mathrm{sim}(D,Q_{\underline{and}(p)})$. Since both expressions (2) and (3) reduce to $[(a_1 d_{A_1} + a_2 d_{A_2} + \ldots + a_n d_{A_n})/(a_1 + a_2 + \ldots + a_n)]$. In other words, for $p = 1$ no distinction is made between <u>and</u> and <u>or</u> operators and the query-document similarity reduces to the vector product between a document $D = (d_{A_1},d_{A_2},\ldots,d_{A_n})$ and a query Q in which the ith term receives the normalized weight $a_i/(a_1+a_2+\ldots+a_n)$.

When $p = \infty$, expressions (2) and (3) reduce to

$$\text{sim}(D,Q_{\underline{or}}(\infty)) = \frac{\max(a_1 d_{A_1}, a_2 d_{A_2}, \ldots, a_n d_{A_n})}{\max(a_1, a_2, \ldots, a_n)} \tag{4}$$

$$\text{sim}(D,Q_{\underline{and}}(\infty)) = 1 - \frac{\max[a_1(1-d_{A_1}), a_2(1-d_{A_2}), \ldots, a_n(1-d_{A_n})]}{\max(a_1, a_2, \ldots, a_n)} \tag{5}$$

Assuming that the query terms all carry a weight of 1, as they do in conventional Boolean systems (that is, $a_1 = a_2 = \ldots = a_n = 1$), one obtains further

$$\text{sim}(D,Q_{\underline{or}}(\infty)) = \max(d_{A_1}, d_{A_2}, \ldots, d_{A_n}) \tag{6}$$

$$\text{and} \quad \text{sim}(D,Q_{\underline{and}}(\infty)) = \min(d_{A_1}, d_{A_2}, \ldots, d_{A_n}). \tag{7}$$

Thus, when $p = \infty$ and the query terms are not weighted, the query-document similarity is dependent only on the document term of highest weight for $Q_{\underline{or}}$, and the document term of lowest weight for $Q_{\underline{and}}$. This result is precisely the same as that obtained in conventional Boolean retrieval where the values of the d_{A_i} are restricted to 0 and 1; by extension the same result applies in the fuzzy set model where weighted document terms, $0 \le d_{A_i} \le 1$, are allowed.

By varying the value of p between 1 and ∞ and using the query-document similarity functions of expressing (2) and (3), one obtains systems intermediate between a pure vector processing model ($p=1$) and a conventional Boolean system ($p=\infty$). The larger the value of p, the more importance is assigned to the query structure and to a strict interpretation of the <u>or</u> and <u>and</u> operators. On the other hand, the smaller the value of p, the more relaxed is the interpretation of the operators. In general, when $1 < p < \infty$, the normal interpretation of <u>and</u> which requires the presence of <u>all</u> query terms to effect retrieval of a document is replaced by a specification requiring the presence of _most_ terms, or as many terms as possible. Correspondingly, the requirement of _one_ matching term for _or_ operators is replaced by a requirement for _some_ terms, or for _a few_ terms. In each case, the query-document similarity increases with the number of matching terms.

It is not difficult to show formally that for values of p between 1 and ∞, the query-document similarity values obtained by evaluating expressions (2) and (3) are intermediate between the limiting values obtained for p=1 and p=∞, respectively. In fact, for $1 < p < \infty$, one can prove that [13,19,20]

$$\begin{aligned} &\text{sim}(D,Q_{\underline{and}}(\infty)) \le \text{sim}(D,Q_{\underline{and}}(p)) \le \text{sim}(D,Q_{\underline{and}}(1)) \\ &= \text{sim}(D,Q_{\underline{or}}(1) \le \text{sim}(D,Q_{\underline{or}}(p)) \le \text{sim}(D,Q_{\underline{or}}(\infty)). \end{aligned} \tag{8}$$

To evaluate the query-document similarity in the extended Boolean system, expressions (2) and (3) can be used directly for pure _or_-queries and pure _and_-queries respectively, that is, queries containing only _or_- or only _and_-operators. For mixed queries, the basic expressions are used recursively. For example, given the query $Q = \{(A,a)\ \underline{or}^p\ (b,[(E,e)\ \underline{and}^p\ (F,f)]\}$, where b represents a special clause weight for the _and_-clause, and a, b and f are the weights of query terms A, E and F, respectively, the corresponding query-document similarity with document $D = (d_A, d_E, d_F)$ is obtained as

$$\text{sim}(D,Q) = \left\{ \frac{a^p d_A^p + b^p \left[1 - \left(\frac{e^p(1-d_E)^p + f^p(1-d_F)^p}{e^p+f^p} \right)^{1/p} \right]^p}{a^p+b^p} \right\}^{1/p} \tag{9}$$

In many practical cases no special clause weights beyond the individual term weights are assigned. In that case all clause weight parameters, such as b in expression (9), may be assumed to be equal to 1.

In the extended system, the retrieval value of any document D depends on the size of the corresponding query-document similarity computed in accordance with the specifications of expressions (2) and (3). Hence to use the extended system it becomes necessary to compute a similarity coefficient, sim(D,Q), for each document of interest, and to rank the documents in decreasing order of the query-document similarity. When the document collection is large, as it often is in practice, it may become inconvenient to carry out a large number of such similarity computations while the user is waiting at the console for system responses. In that case, the following practical method may serve for implementing the automatic relevance feedback procedures in the extended Boolean model:

a) A broad conventional Boolean query is first used to retrieve a *subcollection* of potentially relevant items. This retrieval operation can be carried out using the conventional techniques currently available in existing retrieval systems. If the initial formulations are sufficiently general, one may expect that most items of interest will be included in the subcollections identified by the initial searches.

b) All subsequent relevance feedback and document ranking operations may then be carried out with the subcollections identified in step a), using a special back-end systems package to perform the search operations in the extended, p-valued mode. A high degree of effectiveness and efficiency should be attainable with subcollections limited in size to a few thousand documents.

The Boolean feedback and extended system operations are evaluated in the remaining section of this study.

4. Evaluation

Two main methodologies must be considered: the Boolean feedback system including the construction of improved Boolean queries based on relevance assessments obtained for certain previously retrieved documents, and the extended retrieval model consisting of the term weighting and document ranking operations and the relaxed interpretation of the Boolean operators.

Three sets of experiments are used to evaluate the effectiveness of the feedback operations in the extended Boolean model, including first a standard relevance feedback test in which all original and reformulated queries are strictly Boolean and conventional retrieval methodologies are used based on inverted file manipulations. In experiment 1, conventional Boolean queries are used to produce conventional first iteration Boolean queries based on relevance assessments obtained for ten randomly chosen documents retrieved by the initial queries. Additional feedback iterations are then used to construct conventional second- and third-iteration Boolean queries.

Experiment 2 is designed to test the operations of the extended retrieval model as it would be implemented with currently existing operational retrieval systems. In that case, ordinary Boolean queries are first used to isolate a subcollection of potentially relevant documents using the conventional Boolean methodologies with inverted files. All subsequent search operations beyond the initial one are however conducted in the extended, p-valued system. In previous experiments, a p-value equal to 2 was found to be generally helpful. [20] Hence p=2 is used as a standard for the extended system experiments reported in this study.

Experiment 3 is designed to test the full operations of the extended system by performing all search iterations, including also the initial search in the extended, p-valued system. Once again, a standard p-value of 2 is used throughout.

To evaluate the effectiveness of a retrieval operation, it is customary to compute values of the search _precision_ (the proportion of the retrieved items that are relevance) averaged over a number of user queries at certain fixed values of the search _recall_ (the proportion of relevant items that are retrieved). Typically a recall-precision table may then be displayed giving the average precision at fixed recall levels from 0.1 to 1.0 in steps of 0.1. [2] Since it is impossible in the present context to display dozens of complete recall-precision tables, a single precision value is used for each run, representing the average precision values obtained for three typical recall levels, including a low recall of 0.25, a medium recall of 0.50, and a high recall of 0.75. Percentage improvement or deterioration values are also given for the composite precision measures.

In a relevance feedback evaluation, special precautions must be taken to avoid crediting the feedback search with improvements in the retrieval ranks of relevant documents already retrieved in an earlier search and used to construct the feedback queries. This can be done by freezing the retrieval ranks of any previously retrieved relevant documents for all subsequent search iterations. At the same time, all nonrelevant previously retrieved documents may be disregarded by removing them from the collection to be searched by the feedback queries. This process is known as _partial rank freezing_. [14,21]

The partial rank freezing operation is illustrated for two search iterations by the sample output of Fig. 4. The assumption is that 10 new documents are retrieved and judged for relevance in each iteration. The relevant retrieved items are then used to construct the feedback queries for the next search iteration. In the example of Fig. 4, two items out of 10 are assessed as relevant--those in ranks 3 and 7. These two items are kept in these same ranks in all subsequent feedback searches, while the nonrelevant items originally in ranks 1, 2, 4, 5, 6, 8, 9, and 10 are discarded. The output obtained for the first feedback iteration queries is then compared against a simulated "continuation" of the original search obtained by replacing the discarded nonrelevant items by new items in the original retrieval order. For the example of Fig. 4 the eight discarded items are replaced by the items originally retrieved in ranks 11 to 18, as shown in the middle portion of Fig. 4.

For the next search iteration 10 items not previously seen must be considered, plus the two original items still present in ranks 3 and 7. Three additional relevant items are now identified--those originally appearing in ranks 11, 13, and 19--and these three together with the original two are now frozen in rank for the second feedback iteration. The output of the second feedback iteration is then compared against a second continuation of the original search shown at the bottom of Fig. 4, or against a first search continuation for the first iteration query output. In general the output of the nth feedback iteration with the appropriate frozen ranks is directly comparable to the original search output with n search continuations, or the output of the (n-1)th iteration feedback queries with one search continuation. The partial rank freezing process is used for all evaluation output included in this study.

Four sample document collections, known as Medlars, CISI, CACM, and Inspec are used for experimental purposes. These collections cover the areas of biomedicine, documentation, computer science and electrical engineering, respectively, and range in size from just over 1000 documents for Medlars to over twelve and a half thousand for Inspec. The relevant collection statistics are included in Table 2. Three iterations of relevance feedback are carried out in each experiment. The feedback queries were constructed in each case using a qcount of 2, and a desired number of retrieved items, T, equal to 50 for the smaller collections (Medlars, CISI, CACM), and T = 100 for the larger Inspec collection. The feedback queries Q were defined as (Q_{new} <u>or</u> Q_{old}), that is, the original query Q_{old} was preserved intact and added to the new feedback terms, Q_{new}, using <u>or</u> operators.

The p-value assignments and term weighting methods used to perform the four searches in each experiment (original search plus three iterations of feedback) may be summarizecd as follows:

a) In experiment 1 covering the conventional Boolean searches all query and document terms are unweighted, that is, all terms receive a common weight of 1, and all $p = \infty$.

b) In the mixed search system of experiment 2, the original searches are pure Boolean (unweighted terms and $p = \infty$); all subsequent searches use p-values equal to 2 and weighted query and document terms.

c) Finally in the extended system searches of experiment 3, p-values equal to 2 are used throughout and all terms are weighted. The weights used for <u>document</u> terms are defined as tf × idf (term frequency times inverse document frequency), where the term frequency is equal to the frequency of occurrence of the term in the given document, and the inverse document frequency is defined as $\log N/n_i$, n_i being the postings frequency of the term and N the collection size. For <u>query</u> term weights a distinction must be made between the initial search when no information is available about the occurrence of query terms in any previously retrieved relevant documents, and the subsequent feedback searches when such information is available. For the feedback searches, term relevance weights can be defined as $(r_i/R - n_i/N)$ where r_i is the number of previously retrieved relevant documents containing term i and R is the total number of retrieved relevant items; for the initial searches, an inverse document frequency weight is used also for the query terms. (It is known that the idf weights represent a good approximation to the term relevance in many circumstances. [22-24])

The basic relevance feedback results for the three experiments appear in Table 3 using the search precision at three recall points averaged over the query set as an evaluation criterion. The following conclusions are apparent:

a) In the initial search (top line of Table 3), the extended Boolean system of experiment 3 far outperforms the conventional Boolean searches (experiments 1 and 2), the difference in performance being at least 55 percent for the CISI collection and as much as 156 percent for Medlars.

b) In the three feedback iterations, the extended system used in experiments 2 and 3 again outperforms the Boolean searches of experiment 1 by a wide margin. The improvement in performance obtained from one search iteration to the next varies from collection to collection, exceeding 10 percent in most cases and reaching over 100 percent in some instances.

c) There is little difference in the overall performance of the mixed strategy of experiment 2 and the pure extended system of experiment 3.

Hence in practice, one might as well use the mixed strategy based on initial conventional Boolean searches since the latter is compatible with existing operational retrieval systems.

d) The overall improvement obtained with the mixed feedback strategy for three feedback iterations is impressive varying from 158 percent for CISI to 288 percent for Medlars (see last line of Table 3).

The evaluation data of Table 3 are produced by three distinct effects: the feedback effect which reflects the retrieval of a larger number of relevant documents in the various search iterations than in the initial search, the effect of the extended Boolean system with the relaxed interpretation of the Boolean operators, and finally the effect of the extra work involved in constructing the feedback queries and performing the new feedback searches.

This latter effect can be removed by comparing the output of each feedback iteration with a simple continuation of the search using the previous query set as previously explained. The relevant performance output appears in Table 4. The following conclusions are evident:

a) The amount of improvement is generally greatest for the first feedback iteration, but additional smaller improvements of a few percentage points are obtainable in subsequent search iterations.

b) The relevance feedback methods are of greatest help for the pure Boolean queries of experiment 1 which exhibit the poorest initial performance, and of least help for the p-valued queries of experiment 3 which exhibit the best initial performance.

c) The mixed retrieval strategy of experiment 2 shows large improvements in the first iteration due in part to the change in the query formulation system from pure Boolean to p-valued; subsequent feedback iterations improve the output by 1 to 14 percent compared with a search continuation using the previous output.

The relevance feedback strategies introduced in this study were used to obtain improvements in retrieval output ranging from over 150 to nearly 300 percent in search precision for document collections in widely varying disciplines. The feedback strategies are compatible with existing technologies. One hopes that appropriate tests can be conducted in operational retrieval environments before long.

References

[1] F.W. Lancaster and E.G. Fayen, Information Retrieval On-Line, Melville Publishing Company, Los Angeles, California, 1973.

[2] G. Salton and M.J. McGill, Introduction to Modern Information Retrieval, McGraw Hill Book Company, New York, 1983.

[3] J.J. Rocchio Jr., Relevance Feedback in Information Retrieval, Chapter 14, in The Smart System--Experiments in Automatic Document Processing, Prentice Hall Inc., Englewood Cliffs, NJ, 1971 (reprinted from Scientific Report ISR-9, Computation Laboratory, Harvard University, August 1965).

[4] E. Ide, New Experiments in Relevance Feedback, Chapter 16, in The Smart System--Experiments in Automatic Document Processing, Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[5] E. Ide and G. Salton, Interactive Search Strategies and Dynamic File Organization in Information Retrieval, Chapter 18, in The Smart System--Experiments in Automatic Document Processing, Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[6] G. Salton, Relevance Feedback and the Optimization of Retrieval Effectiveness, Chapter 15, in The Smart System--Experiments in Automatic Document Processing, Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[7] C.J. van Rijsbergen, D.J. Harper, and M.F. Porter, The Selection of Good Search Terms, Information Processing and Management, Vol. 17, 1981, p. 77-91.

[8] D.J. Harper and C.J. van Rijsbergen, An Evaluation of Feedback in Document Retrieval Using Co-occurrence Data, Journal of Documentation, Vol. 34, 1978, p. 189-216.

[9] S.E. Robertson and K. Sparck Jones, Relevance Weighting of Search Terms, Journal of the ASIS, Vol. 23, No. 3, 1976, p. 129-146.

[10] J.T. Rickman, Design Considerations for a Boolean Search System with Automatic Relevance Feedback Processing, Proceedings of the ACM National Meeting, Association for Computing Machinery, New York, August 1971, p. 478-481.

[11] M. Dillon and J. Desper, Automatic Relevance Feedback in Boolean Retrieval Systems, Journal of Documentation, Vol. 36, 1980, p. 197-208.

[12] M. Dillon, J. Ulmschneider, and J. Desper, A Prevalence Formula for Automatic Relevance Feedback in Boolean Systems, Information Processing and Management, Vol. 19, No. 1, 1983, p. 27-36.

[13] E.A. Fox, Extending the Boolean and Vector Space Models of Information Retrieval with P-Norm Queries and Multiple Concept Types, Doctoral Thesis, Cornell University, August 1983.

[14] G. Salton, E.A. Fox, C. Buckley and E. Voorhees, Boolean Query Formulation with Relevance Feedback, Technical Report TR 83-539, Department of Computer Science, Cornell University, January 1983.

[15] G. Salton, E.A. Fox, and E. Voorhees, Comparison of Two Methods of Boolean Feedback, Technical Report TR 83-564, Department of Computer Science, Cornell University, Ithaca, New York, July 1983.

[16] A Bookstein, A Comparison of Two Systems of Weighted Boolean Retrieval, Journal of the ASIS, Vol. 31, No. 4, July 1981, p. 275-279.

[17] A. Bookstein, Fuzzy Requests: An Approach to Weighted Boolean Searches, Journal of the ASIS, Vol. 31, No. 4, July 1980, p. 240-247.

[18] D.A. Buell and D.H. Kraft, A Mathematical Model for a Weighted Boolean Retrieval System, Information Processing and Management, Vol. 17, No. 3, 1981, p. 127-136.

[19] H. Wu, On Query Formulation in Information Retrieval, Doctoral Thesis, Cornell University, Ithaca, New York, January 1981.

[20] G. Salton, E.A. Fox and H. Wu, Extended Boolean Information Retrieval, Technical Report TR 82-511, Department of Computer Science, Cornell University, August 1982.

[21] Y.K. Chang, C. Cirillo, and J. Razon, Evaluation of Feedback Retrieval Using Modified Freezing, Residual Collection, and Test and Control Groups, Chapter 17, in The Smart System--Experiments in Automatic Document Processing, Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[22] K. Sparck Jones, A Statistical Interpretation of Specificity and its Application in Retrieval, Journal of Documentation, Vol. 28, No. 1, March 1972, p. 11-20.

[23] W.B. Croft and D.J. Harper, Using Probabilistic Models of Document Retrieval without Relevance Information, Journal of Documentation, Vol. 35, No. 4, December 1979, p. 285-295.

[24] H. Wu and G. Salton, A Comparison of Search Term Weighting: Term Relevance vs. Inverse Document Frequency, ACM SIGIR Forum, Vol. 16, No. 1, Summer 1981, p. 30-39.

information : { 3, 15, 23, 29, 30 }		original lists of document identifiers
retrieval : { 2, 15, 21, 27, 29 }		
information retrieval:		
{ 2, 3, 15, 15, 21, 23, 27, 29, 29, 30 }		merged list
information <u>and</u> retrieval:		
{ 15, 29 }		recognition of duplicated items
information <u>or</u> retrieval		
{ 2, 3, 15, 21, 23, 27, 29, 30 }		elimination of one instance of each duplication

List Manipulations Corresponding to Boolean <u>or</u>, <u>and</u>

Fig. 1

a) for all terms t in a relevant retrieved document or in original query:

b) compute adjusted postings frequency n_t of the term (n_t is the number of documents in the collection to which term t is assigned; for query terms n_t is adjusted by the qcount parameters);

c) compute the relevance weights, $relwt_t$, as the difference between the proportion of relevant retrieved documents in which term t occurs and the proportion of documents in the collection in which term t occurs;

d) keep the m single terms with the best relevance weights;

e) repeat steps a) through c) for all pairs of terms s,t where s,t are on the list of m best singles; the postings frequency of a pair is estimated as $n_s \cdot n_t / N = n_{st}$ where N is collection size;

f) keep the m term pairs with the best relevance weight, $relwt_{st}$;

g) repeat steps a) through c) for all triples of terms r,s,t where r,s,t are distinct terms on best singles or pair list; the postings frequency of a pair is estimated as $n_r \cdot n_{st} / N = n_{rst}$;

h) keep the m term triples with best relevance weight, $relwt_{rst}$.

Generation of List of Good Single Terms, anded Pairs, [illegible] Triples

Fig. 2

i) start with query consisting of m single terms related by *or*-operators;

j) compute the estimated number of retrieved items, estret, as sum of postings frequencies of individual terms

$$estret = \sum_t n_t;$$

k) *while* estimated number retrieved is greater than desired number, estret > T, remove the single term or the *anded* term clause with the smallest relevance weight;

if the removed term is a single term t, recompute estret:

$$estret = estret - n_t;$$

(*if* estret < T/2, put term t back in set and exit);
mark term t as discarded;
add to current term set all good pairs s,t where s is another discarded single; recompute estret:

$$estret = estret + \sum_{\substack{\text{added}\\ \text{pairs}}} n_{s,t}$$

else if the removed term clause is a term pair (s *and* t)
$estret = estret - n_{st}$, and mark pair s,t as dicarded;

(*if* estret < T/2, put pair s,t back in set and exit);
add to current term set all good triples r,s,t not subsumed by existing term clauses and adjust estret count

else if the removed term clause is a term triple (r *and* s *and* t)
$estret = estret - n_{rst}$

if estret < T/2, put term triple back in set and exit;

ℓ) form query by *or*-ing together all term clauses left in good term set; optionally add original query using *or* operations

Generation of Feedback Query using DNF Process

Fig. 3

Sample initial ranking (circled items are relevant)

1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17 18 19 20 21 22 23 24

Use items 3 and 7 to construct first iteration feedback query

Sample initial ranking continued once (freeze n_0 relevant items in top 10)

11 12 3 13 14 15 7 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30 ...

Use items 11, 3, 13, 7, 19 to construct second iteration feedback query

Sample initial ranking continued twice (freeze n_0+n_1 relevant in top $10+n_0$)

11 21 3 13 22 23 7 24 25 26 19 27 28 29 30	31 .....

Use items 11 3 13 22 7 19 to construct third iteration feedback query

Example of Partial Rank Freezing

Fig. 4

<u>Original Query</u>: Excretion (ex) of phosphate (ph) in urine (ur)

Term clauses	Estimated Postings Frequency	Clause Relevance Weight
excretion (ex)	52	.9497
phosphate (ph)	43	.9584
urine (ur)	78	.9245
—		
(ex <u>and</u> ph)	2.2	.9979
(ex <u>and</u> ur)	3.9	.9962
(ph <u>and</u> ur)	3.2	.9969
—		
(ex <u>and</u> ph <u>and</u> ur)	0.2	.9998

a) Clause Characterization for Sample Query

Query Formulation in DNF System	Estimated Number of Retrieved Items	
(ex <u>or</u> ph <u>or</u> ur)	173	
(ex <u>or</u> ph)	95	(173 - 78)
(ph <u>or</u> (ex <u>and</u> ur))	46.9	(95 - 52 + 3.9)
(ex <u>and</u> ph) <u>or</u> (ex <u>and</u> ur) <u>or</u> (ph <u>and</u> ur)	9.3	(46.9 - 43 + 2.2 + 3.2)
(ex <u>and</u> ph) <u>or</u> (ph <u>and</u> ur)	5.4	(9.3 - 3.9)
(ex <u>and</u> ph)	2.2	(5.4 - 3.2)
(ex <u>and</u> ph <u>and</u> ur)	0.2	(2.2 - 2.2 + 0.2)

b) Refinement Process for Sample Query

Table 1

Collection Statistics	Medlars (biomedicine)	CISI (documentation)	CACM (computer science)	Inspec (electrical engineering)
Number of documents	1033	1460	3204	12684
Number of queries	30	35	52	77
Average number of relevant documents per query	23.2	49.8	15.3	33.0
Average number of relevant documents retrieved in top 10 by initial				
a) Boolean queries	3.9	1.7	1.8	2.1
b) p-valued queries	5.9	3.0	2.8	3.7
Number of queries for which no relevant items are retrieved in top 10 ranks				
a) Boolean queries	5	12	18	23
b) p-valued queries	1	9	9	10

Collection Statistics

Table 2

Type of Run	Medlars 1033			CISI 1460			CACM 3204			Inspec 12684		
	Exp. 1	Exp. 2	Exp. 3	Exp. 1	Exp. 2	Exp. 3	Exp. 1	Exp. 2	Exp. 3	Exp. 1	Exp. 2	Exp. 3
Original Query Statements (improvement of p-valued queries over Boolean queries)	.2065	.2065	.5279 (+156%)	.1118	.1118	.1728 + 55%)	.1798	.1798	.3131 (+ 74%)	.1159	.1159	.2498 (+116%)
First Iteration Feedback Queries	.4322 (+110%)	.6147 (+198%)	.6967 (+ 32%)	.1367 (+ 22%)	.2168 (+ 94%)	.1969 (+ 14%)	.2550 (+ 42%)	.3706 (+106%)	.3817 (+ 22%)	.1522 (+ 31%)	.3002 (+159%)	.3019 (+ 21%)
Second Iteration Feedback Queries	.5420 (+ 25%)	.7588 (+ 23%)	.7825 (+ 12%)	.1588 (+ 16%)	.2486 (+ 15%)	.2315 (+ 18%)	.2837 (+ 11%)	.4681 (+ 26%)	.4499 (+ 18%)	.1699 (+ 12%)	.3546 (+ 18%)	.3476 (+ 15%)
Third Iteration Feedback Queries	.6334 (+ 17%)	.8003 (+ 5%)	.8211 (+ 5%)	.1827 (+ 15%)	.2885 (+ 16%)	.2738 (+ 18%)	.3217 (+ 13%)	.5252 (+ 12%)	.5157 (+ 15%)	.1933 (+ 14%)	.3990 (+ 13%)	.3786 (+ 9%)
(Overall Improvement of Mixed Feedback over Original Boolean Output)		(+288%)			(+158%)			(+199%)			(+244%)	

Relevance Feedback Results for Three Iterations

(Experiment 1: conventional Boolean feedback;
Experiment 2: mixed strategy;
Experiment 3: extended Boolean feedback)

Table 3

Type of Run	Medlars 1033			CISI 1460			CACM 3204			Inspec 12684		
	Exp. 1	Exp. 2	Exp. 3	Exp. 1	Exp. 2	Exp. 3	Exp. 1	Exp. 2	Exp. 3	Exp. 1	Exp. 2	Exp. 3
Original Queries continued once	.2372	.2372	.6099	.1263	.1263	.1934	.2118	.2118	.3562	.1262	.1262	.2856
First Iteration Feedback Queries	.4322 (+ 82%)	.6147 (+159%)	.6967 (+ 14%)	.1367 (+ 8%)	.2168 (+ 72%)	.1969 (+ 2%)	.2550 (+ 20%)	.3706 (+ 75%)	.3817 (+ 7%)	.1522 (+ 21%)	.3002 (+138%)	.3019 (+ 6%)
First Iteration Feedback continued once	.5144	.6683	.7545	.1544	.2374	.2226	.2914	.4269	.4327	.1720	.3432	.3427
Second Iteration Feedback Queries	.5420 (+ 5%)	.7588 (+ 14%)	.7825 (+ 4%)	.1588 (+ 3%)	.2486 (+ 5%)	.2315 (+ 4%)	.2837 (- 3%)	.4681 (+ 10%)	.4499 (+ 4%)	.1699 (- 1%)	.3546 (+ 3%)	.3476 (+ 1%)
Second Iteration Feedback continued once	.6043	.7947	.8162	.1793	.2803	.2620	.3127	.5270	.4994	.1873	.3950	.3754
Third Iteration Feedback Queries	.6334 (+ 5%)	.8003 (+ 1%)	.8211 (+ 1%)	.1827 (+ 2%)	.2885 (+ 3%)	.2738 (+ 5%)	.3217 (+ 3%)	.5252 (0%)	.5157 (+ 3%)	.1933 (+ 3%)	.3990 (+ 1%)	.3786 (+ 1%)

Comparison of Feedback Query Effectiveness versus Search Continuation

using Previous Query Statements

Table 4