BOOLEAN QUERY FORMULATION WITH RELEVANCE FEEDBACK

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

January 1983
TR 83-539

[†]Department of Computer Science
Cornell University
Ithaca, NY 14853

[*]International Institute of Tropical Agriculture
Ibadan, Nigeria

This study was supported in part by the National Science Foundation, under grant IST-81-08696

BOOLEAN QUERY FORMULATION WITH RELEVANCE FEEDBACK

G. Salton†, E.A. Fox*, C. Buckley† and E. Voorhees†

Abstract

The well-known relevance feedback process uses information extracted from previously retrieved relevant documents to generate improved search formulations for subsequent search iterations. Methods are outlined in this study for the automatic generation of Boolean search statements based on the natural language texts of initially available search requests and of previously retrieved document excerpts identified as relevant by the user population. The search requests are generated both in a conventional Boolean system and in an extended system in which the normal interpretation of the Boolean connectives is relaxed. Experimental output is included which shows that substantial improvements in retrieval effectiveness are obtainable using the automatic relevance feedback methods.

---

†Department of Computer Science, Cornell University, Ithaca, New York 14853.

*International Institute of Tropical Agriculture, Ibadan, Nigeria.

This study was supported in part by the National Science Foundation, under grant IST-81-08696

1. Introduction

It is well known that in most information retrieval situations a single search conducted with an initially available query formulation will not normally produce adequate search results. Instead the search may be conducted in several steps using search requests which are progressively improved based on information obtained as a result of the earlier search efforts. The well-known relevance feedback process uses the texts of previously retrieved items to produce query formulations which are more similar to previously retrieved items identified as relevant than the initial queries, and less similar to the items identified as nonrelevant. Typically, terms may be extracted manually or automatically from the previously retrieved items, and these terms may then be used to enhance the initially available search formulations.

In a vector processing system where queries are expressed as sets of search terms without indicators of term relationship, it is easy to generate improved query formulations automatically by simply adding terms to the queries, or increasing the weight of certain query terms obtained from the relevant documents; alternatively, query terms can be deleted or the term importance for terms obtained from the nonrelevant documents can be decreased. It is known that in a vector processing environment substantial improvement in search effectiveness are obtainable using the automatic relevance feedback methods. [1,2]

The situation is more complicated in a Boolean retrieval system where the queries are expressed as Boolean combinations of index terms. In that case, a query reformulation requires both the choice of new query terms as well as the determination of the Boolean operators to be used with these terms. Attempts have been made to apply an automatic relevance feedback process to Boolean retrieval environments, but no viable procedures of proven effectiveness have so far been suggested. [3-5] Procedures are described in this study for generating Boolean search statements using the texts of previously available query formulations and of documents retrieved in earlier search operations. The query formulations are obtainable in a conventional Boolean form, or in an extended Boolean system. In either case the feedback process produces substantially enhanced retrieval output; however, the extended Boolean system proves much superior to the conventional system which is based on the standard Boolean logic.

2. Automatic Formulation of Boolean Queries

The basic problem in Boolean query formulation consists in first choosing an appropriate set of query terms and in then using the Boolean connectives to generate formulations which are not so broad as to retrieve an unreasonable amount of extraneous matter, nor so narrow as to reject a large proportion of relevant materials. The Boolean connectives can help in producing statements of the correct specificity--that is, neither too broad nor too narrow--by using _and_-clauses, such as (A _and_ B), for terms that are considered too broad, and _or_-clauses, such as (A _or_ B), for terms that are believed to be too narrow.

The specificity of a term (that is, its breadth or narrowness) can be related to the document frequency of the term (the number of documents in which a term occurs). [6-7] The graph of Fig. 1 summarizes the value of a term for retrieval purposes as a function of the document frequency:

a) The terms of low document frequency on the left-hand side of Fig. 1 are not very useful in retrieval, because their rarity renders them unable to distinguish a substantial number of items from the remainder of the collection.

b) The medium frequency terms which occur neither too rarely nor too frequently are the best available search terms.

c) As the document frequency of the terms increases, the value of the terms for retrieval purposes decreases rapidly, the worst terms being those assigned to a substantial number of items. The assignment of high-frequency terms actually renders many of the documents more similar to each other rather than more distinct as required for effective retrieval.

The characterization of Fig. 1 provides a clause formation strategy by which the rare terms are incorporated in _or_-clauses so as to broaden their interpretation, while the high-frequency terms are used in _and_-clauses. In general, the higher the document frequency of a term, the larger should be the _and_-clause in which it will appear. A particular clause forming method shown in Fig. 1 builds _or_-clauses for low frequency terms, _anded_ term pairs for medium high frequency terms, and _anded_ term triples for very high frequency terms. The good medium frequency terms are used as single terms, that is, they produce clauses containing only the given term alone. A complete query statement is then obtainable by connecting the various clauses by an "outside" operator--either a Boolean _or_, or a Boolean _and_. A typical query formulation of this kind appears as

$$\begin{aligned}
&(T_{i_1} \ \underline{or} \ T_{i_2} \ \underline{or} \ldots \underline{or} \ T_{i_k}) \\
&\underline{or} \ T_{j_1} \ \underline{or} \ T_{j_2} \ \underline{or} \ldots \underline{or} \ T_{j_h} \\
&\underline{or} \ (T_{k_1} \ \underline{and} \ T_{k_2}) \ \underline{or} \ (T_{k_3} \ \underline{and} \ T_{k_4}) \underline{or} \ldots \\
&\underline{or} \ (T_{m_1} \ \underline{and} \ T_{m_2} \ \underline{and} \ T_{m_3}) \ \underline{or} \ (T_{m_4} \ \underline{and} \ T_{m_5} \ \underline{and} \ T_{m_6}) \ \underline{or} \ldots
\end{aligned} \tag{1}$$

where the four distinct lines of expression (1) are used to represent the low frequency, medium frequency, medium high frequency and very high frequency terms.*

It remains to specify how the query clauses are actually generated and incorporated in query statements of greater or smaller breadth. [8] For this purpose, it is convenient to use an auxiliary parameter, known as the <u>retrieval threshold</u> T. When T is large, indicating that one wishes to retrieve a substantial number of items, a relatively broad query will be generated; when T is small, the query formulation will be more specific. The actual choice of query clauses can be controlled by <u>clause frequency</u> factors reflecting the estimated number of documents retrieved by each given clause. The following clause frequency estimation process may be used:

a) for single query terms, T_i, the estimated number of retrieved items is taken simply as the postings frequency, n_i, of the term, that is, the number of documents to which term T_i is assigned;

b) for a pair of <u>anded</u> terms (T_i <u>and</u> T_j), the estimated number of retrieved items is $n_{ij} = (n_i \cdot n_j)/N$, where N is the total number of

*The first two lines of expression (1) representing low frequency and medium frequency terms, respectively, are not distinguishable in a conventional Boolean system. An appropriate differentiation arises in extended Boolean systems described later in this study.

documents in the collection;

c) similarly for an <u>anded</u> triple, (T_i <u>and</u> T_j <u>and</u> T_k), the estimated number retrieved is $n_{ijk} = (n_i \cdot n_j \cdot n_k)/N^2$;

d) finally, for sets of <u>ored</u> terms, the estimated number retrieved is $\Sigma\, n_i + \Sigma\, n_{ij} + \Sigma\, n_{ijk}$ where the summation is taken over the singles, <u>anded</u> pairs, and <u>anded</u> triples included in the query formulation.

The suggested clause frequency estimation process is exact only when the terms included in an <u>and</u>-clause are independent of each other, and when no overlap exists among the documents retrieved by the various clauses. However, the method may be good enough to be used in practice.

Consider, as an example, the query "excretion of phosphate in urine" listed at the top of Table 1. The three single terms, as well as the set of <u>anded</u> pairs and <u>anded</u> triples are included in column 1 of Table 1(a). Given the postings frequencies of 52, 43, and 78 for "excretion", "phosphate", and "urine", respectively, the computed frequencies for the <u>anded</u> pairs and triples are shown in column 2 of Table 1(a). Finally, the estimated number of retrieved items is shown in Table 1(b) for two possible Boolean query statements. For the query consisting of three <u>anded</u> pairs, the number retrieved is computed as 2.2 + 3.9 + 3.2 = 9.3. For the more refined query with two pairs, the estimated number retrieved will be 2.2 + 3.2 = 5.6.

The clause frequency parameters can be used to obtain an estimated number of retrieved items for a given query formulation. This number may be compared with the available retrieval threshold T, and a query statement can actually be used for search purposes when the computed number of retrieved items is close to the retrieval threshold. The clause frequency parameters can also be used indirectly to control the query formulation process itself. In fact, it is known that in the absence of special information, the "goodness" of a term, or a term clause, may be determined as an inverse function of the document frequency. [6-7] Inverse document frequency (idf) factors have been used in various systems as weights attached to the terms reflecting the term usefulness. For single terms, term pairs, and term triples with estimated frequencies of n_i, n_{ij}, and n_{ijk}, respectively, the idf factors may be computed as $1 - (n_i/N)$, $1 - (n_{ij}/N)$, or $1 - (n_{ijk}/N)$. For the terms listed in Table 1(a), the inverse document frequency weights are shown in column 3 of the Table.

Given the computed clause frequencies and clause weights for the available single terms, anded pairs, and anded triples, it becomes useful to construct a query using the best possible term combinations such that the estimated number of retrieved items for the query is approximately equal to the wanted retrieval threshold. The following process may be used for this purpose: [8]

a) Consider the set of query terms included in a given natural language query formulation, or the set of terms included in an available Boolean query; use a stop list to eliminate common function words and other terms of no interest for query formulation purposes.

b) Use the remaining single terms and construct anded term pairs and triples (frequency thresholds can be used to avoid constructing pairs and triples of very low weight); compute the clause frequencies and clause weights for all singles, pairs and triples.

c) Use as an initial query a broad formulation consisting of a few single terms of highest weight (lowest frequency) plus those term pairs which do not include any of the chosen singles as components. (For the sample query of Table 1 such an initial formulation might be given as "phosphate <u>or</u> (excretion <u>and</u> urine)".)

d) If the estimated number of items retrieved by the current query is smaller than the retrieval threshold T, the query formulation is broadened by successively adding single terms in decreasing idf weight order while subtracting the pairs subsumed by the added singles. (For example, if term A is added, the pairs A <u>and</u> B, A <u>and</u> C, and so on, become redundant.) Recompute the estimated number of retrieved items following each addition and subtraction.

e) If, on the other hand, the estimated number of retrieved items is larger than the preestablished retrieval threshold, a narrower query formulation is generated by successively removing singles in increasing weight order and adding the missing term pairs which include the deleted singles as components. If the estimated number of retrieved items is still too large, certain term pairs are eventually eliminated in increasing weight order and replaced by triples not subsumed by the existing pairs. For the sample query of Table 1, the narrowest formulation consists of the single triple "excretion <u>and</u> phosphate <u>and</u> urine" with an expected number of 0.2 retrieved items.

The query formation process is illustrated in the next section following the description of the actual relevance feedback process.

3. Automatic Relevance Feedback for Boolean Queries

The automatic query formulation methods outlined in the preceding section offer an approach to a query reformulation based on the texts of previously retrieved documents. Indeed, the initial formulation of Boolean queries based on the texts of natural language statements of user needs is not different in principle from a reformulation of the queries based on certain document texts. Substantially the same system will serve for both purposes.

Since the relevance feedback system is based on the availability of relevance judgments about previously retrieved documents it is, however, possible to replace the simple term weighting system in inverse document frequency order described earlier by a more sophisticated approach. Indeed, when information is available about the occurrences of the term in the relevant and nonrelevant portions of the document collection, then term relevance weights can advantageously be computed. These weights reflect the usefulness of a term as a function of the proportion of relevant items in which the term occurs compared with the total number of items in which it occurs. It is known that in some circumstances the term relevance weights are optimal for retrieval purposes. [9-10] In relevance feedback, only certain previously retrieved relevant items are identified; hence only partial relevance weights can be generated.

In the experiments described in this study, relevance frequency measures are computed as the proportion of relevant documents in which a term occurs minus the proportion of all the documents in a collection in which the term occurs ($r_i/R - n_i/N$), where r_i represents the number of known relevant items with term i, R is the total number of known relevant items, and n_i and N represent respectively the total number of items with term i and the total number of items in the given collection. [11] Correspondingly the relevance weight of a term can be defined as the relevance frequency multiplied by the inverse document frequency. Thus for term i, the relevance weight w_i is defined as

$$w_i = (\frac{r_i}{R} - \frac{n_i}{N}) \; (1 - \frac{n_i}{N}) \tag{2}$$

The construction of the term relevance frequencies used in the automatic query reformulation process is illustrated in Table 2. The initial query is listed at the top of Table 2 together with the title and abstract of a relevant document retrieved during the feedback process. The combined term set from the original query as well as from the relevant document appears in Table 2(a) together with the respective document frequencies (n_i), and the frequencies in the relevant items (r_i). In the computations of Table 2 a term occurrence in the original query is assumed to be equivalent to the occurrence of the term in one relevant document; this is reflected in a special parameter setting, known as _query count_, or q-count, which is set equal to 1 for the computations of Table 2. The query terms can be weighted more highly by using q-count settings of 2, 3, 4 or higher. The total number of relevant documents retrieved, R, is equal to 2 in the calculations of Table 2, and the collection size N is 1033.

Table 2(b) shows the computation of the term relevance frequencies for all pairs which co-occur in the query and the given relevant document. The computation for co-occurring term triples is included in Table 2(c). It is clear from Table 2 that the two terms which are contained in both the query and the relevant document (excretion and phosphate) receive much higher relevance weights than terms contained only in one of the two constructs. The same is true for the pair (ex _and_ ph).

A typical query reformulation process is illustrated in Table 3 for the query used as an example in Tables 1 and 2. The assumption as before is that the relevant item given at the top of Table 2(a) is retrieved in response to the sample query. The initial formulation of the feedback query consists of the six single terms with the highest relevance weights (from Table 2(a)) plus the term pairs from Table 2(b) that are not "covered" by the chosen single terms (that is, those pairs not containing any of the included singles). The expected number of retrieved items is 324.3 for the initial formulation. To refine the initial feedback query one removes the single term of lowest weight (al) while adding the pairs no longer covered because of the deletion (ex <u>and</u> al) and (ph <u>and</u> al). This reduces the expected number of retrieved items to 261.64 as shown in the second part of Table 3. Additional deletions of single terms followed by the addition of term pairs reduces the expected number of retrieved items to 77.08.

If the query is still too broad, certain term pairs may be successively removed in increasing relevance weight order, and any available term triples that are not covered may be added. (In the example of Table 3, no available term triples can actually be added.) A feedback query is eventually obtained in Table 3 which is narrow enough to be used for actual processing. The expected number of retrieved items is equal to 10.99.

The query count used to generate the query of Table 3 is equal to 1. In these circumstances, query terms which are present only in the query but not in the feedback documents may receive low weights in the reformulated query. This is true for the term "urine" in the example of Tables 2 and 3. The final query statement does not in fact contain the "ur" term. Such a possibly undesirable outcome may be avoided by using a higher q-count value. The query reformulation process of Tables 2 and 3 is repeated in Tables 4 and 5 using a query count value of 2. Only a selection of terms is included in Table 4 to illustrate the differences with the earlier list of Table 2. Query terms that occur in the query as well as the retrieved relevant document now receive an r_i value of 3 (that is, 2 + 1), and terms such as "urine" that occur only in the query have an r_i value of 2. Correspondingly the pair (ex and ph) receives a relevance weight of 0.9958, whereas the pairs which include "urine" (ur and ex), and (ur and ph) are assigned relevance weights of 0.6537 and 0.6549 respectively. This is large enough to insure that the "ur" term will not disappear from the final query.

The query construction process is shown in Table 5 for the case where the query count is set at 2. The final query which is expected to retrieve 10.07 items contains 5 pairs which include all of the original query terms. A summary of the various automatically obtained sample queries appears in Table 6. The initial query is listed at the top of Table 6 in natural language form. An automatic formulation obtained without any feedback information using the process of Table 1 follows. Obviously, only the original query terms are included in such a formulation. The two formulations with relevance feedback are shown in the lower part of Table 6 using q-count values of 1 and 2 respectively. The boxed terms are those which are also present in the query constructed without feedback information. It is clear that the formulation using the q-count value of 2 is much more similar to the original query than the alternative form with the q-count equal to 1. This is also seen by comparing the respective term weights listed below each term in Table 6. The last query formulation using the q-count of 2 exhibits the most discriminating term weights. Higher q-count values than 2 are also possible producing reformulated queries which are increasingly similar to the original query statements.

The basic query reformulation process using relevance feedback may be evaluated by comparing the retrieval results obtained for the automatically formulated feedback queries with the results obtained with initially available query formulations. To evaluate the effectiveness of a retrieval system it is customary to compute values of the search recall (the proportion of relevant items retrieved) and of the search precision (the proportion of retrieved items that are relevant) following the retrieval of some fixed number of documents. In general, the presumption is that an average user is interested in retrieving most everything that is relevant (high recall), and in rejecting most everything that is extraneous (high precision).

In some retrieval environments such as the vector processing system, the documents are retrieved in ranked order in accordance with the decreasing values of a query-document similarity measure. This makes it possible to compute recall and precision values after the retrieval of each item. In a conventional Boolean retrieval system where the set of relevant and nonrelevant retrieved items is not ranked by the system, a ranked output can be simulated by defining an order for the retrieved items which places the relevant items randomly in order among the set of nonrelevant items. Such a simulated ordering operation again permits a recall and precision computation following consideration of each document in turn in the simulated retrieval order. By interpolation, the precision values can be calculated for fixed values of the recall, say for a recall of 0.1, 0.2, and so on, up to a recall of 1.0. By averaging the precision values at the fixed recall levels for a number of user queries, one finally obtains a recall-precision table, or a corresponding graph. [12]

Since it is awkward to compare complete tables or graphs with each other, the evaluation results given in this study are stated in terms of a single precision value, representing the average precision obtained at three typical recall levels, including a low recall level of 0.25, a medium recall of 0.50, and a high recall of 0.75. Percentage improvement or deterioration values may then be given for each of these composite precision measures. In evaluating a relevance feedback process, 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 using a partial rank freezing procedure by which the retrieval ranks of any relevant documents identified in an earlier search are frozen in all subsequent searches performed with the corresponding queries. At the same time, the nonrelevant documents retrieved in an earlier search are simply deleted from the collection and not further considered in the feedback process. This procedure insures that any relevant document previously retrieved and used for the construction of feedback queries is counted equally for evaluation purposes in the original as well as any subsequent searches. [13]

A single step of relevance feedback thus consists of the following steps:

a) perform an initial retrieval run A using the originally available queries;

b) identify the relevant and nonrelevant items retrieved in the top n ranks (n = 10 for the experimental output included in this study), and construct a first iteration feedback query; remove all n documents already seen from consideration for further search activities;

c) perform retrieval run B using the first iteration feedback queries; rank the retrieved documents in decreasing order of query-document similarity; at the same time reintroduce into the new ranking all relevant items retrieved by run A in the top n ranks (and hence used for the construction of the first iteration feedback queries), and assign to these relevant items their original retrieval ranks;

d) compare the output of retrieval run B with output of the "continued" run A obtained by freezing the ranks of all relevant in the top n ranks, removing all non-relevant items from the top n ranks, and replacing them in order by items not yet seen, originally appearing in ranks n+1, n+2, and so on.

The outputs of run B and "continued" run A are precisely comparable because the same documents are missing from both runs (namely the nonrelevant appearing in the top n ranks for the original run A), and the relevant documents already seen are assigned the same frozen ranks.

An example of the partial rank freezing process is shown in Table 7 where the assumption is that n = 5, that is, only 5 items are examined by the user for feedback purposes. The original query output for run A is shown in the left portion of Table 7. Based on the identification of items b and d as relevant (R), a new first iteration feedback query is built which is now used to search the reduced collection obtained by removing the originally retrieved nonrelevant items (a, c, and e). The ranked output obtained by run B is shown in the right-hand column of Table 7. As always, the items retrieved by the first iteration feedback query are arranged in decreasing order of the query-document similarity, except that the relevant items b and d used to build the feedback query are maintained in their original ranks (ranks 2 and 4, respectively).

For evaluation purposes the output of the feedback run is compared with the "continued" original run A shown in the middle portion of Table 7, where once again the relevant items already seen (items b and d) keep their original ranks, but all the items not yet seen starting at rank n+1 are promoted into the slots left open by the removal of the nonrelevant items originally located in the top n. The items starting at rank n+1 all keep the same relative order except that they appear at a somewhat better rank in the continued run than they did originally.

The procedure can be repeated for a second feedback iteration, by again looking at the top n items retrieved by the first iteration feedback query (except for items b and d already seen), that is, items j, i, f, g, and k in the example of Table 7. This time, items j and f are identified as relevant, and hence the full set of relevant items identified up to now (b,d,j and f) is used to construct a second iteration feedback query. The output of this second iteration with ranks 1,2,4 and 5 now frozen is then compared against the "continued" run B with nonrelevant items in the top n ranks (i, g, and k) replaced by lower ranking items as explained earlier.

The conventional Boolean feedback process is evaluated in Table 8 using a biomedical collection of 1033 documents and 30 queries (the Medlars 1033 collection). The evaluation output of Table 8(a) covers the performance of the automatically constructed feedback queries using as basic input a set of 30 manually prepared Boolean query formulations. In Table 8(b) the feedback queries are constructed by using as basic input the natural language formulations of user needs originally obtained from the users, and interposing an automatic query construction process which generates conventional Boolean queries from the natural language forms. The steps performed in obtaining the evaluation output of Table 8 are summarized in the chart of Fig. 2.

In addition to initial and first iteration feedback runs, the output of Table 8 also includes the results of a vector processing strategy in which automatically constructed vector queries are used with weighted terms, and a cosine matching coefficient serves for the query-document similarity computations. The output of Table 8(a) shows that the automatically constructed Boolean feedback queries (designated by the parameter $p = \infty$ in Table 8) produce improvements ranging from 65 to 85 percent in the average search precision over the conventional manually constructed Boolean queries used without feedback but with rank freezing as previously explained. However, the automatic Boolean feedback queries show a substantial deficit over the automatically constructed vector processing queries. For the Boolean feedback queries constructed from the natural language formulations, the advantage of the feedback step is 25 to 40 percent over the automatically constructed Boolean statements used in Table 8(b). The deficit of the automatic feedback queries over the vector processing queries noted earlier for Table 8(a) is maintained in Table 8(b).

Two types of feedback queries are used in Table 8 for three q-count values. In the first instance the feedback query is defined simply as the newly constructed query Q_{new} using the processes illustrated in Tables 3 and 5. In the other case, the original query, Q_{old}, is preserved, and the new query is defined as the disjunction of old and new queries (Q_{old} _or_ Q_{new}). This last query formulation produces somewhat better results than the former; the best q-count values are equal to 2.

An optimal search run is included at the bottom of Table 8 which is obtained by making the unrealistic assumption that all relevant documents are known in advance of the search. In such a case, it becomes possible to compute exact term relevance weights for each original query term. Such a retrospective run represents the best possible performance obtainable with the original query formulations and document texts actually in use.

The automatic query reformulation process can be repeated several times and additional enhancements in search effectiveness may be obtained beyond the first search iteration as shown later in this study. It is unlikely, however, that really spectacular search results are obtainable when conventional Boolean queries are automatically generated. The problem is that the terms or term groups included in a given query formulation are determined by using only the formal frequency considerations for the terms. An automatic query thus represents at best a tentative statement of user need. In such circumstances it makes little sense to use the strict interpretation of the term relationship inherent in the conventional Boolean and and or operators. A relaxed form of Boolean query reformulation may serve in these circumstances which may be expected to produce superior retrieval results. This is briefly explained in the next section of this study.

4. Relevance Feedback Using the Extended Boolean System

The automatic query construction process outlined earlier may create semantically questionable phrases. In these circumstances, improved retrieval results may be obtainable by relaxing the strict interpretation of the Boolean operators and using a fuzzy type of query document matching system. An extended Boolean retrieval system has recently been implemented in which an auxiliary parameter p, $1 \leq p \leq \infty$, attached to the Boolean operators is used to control the query-document matching process, providing thereby a variety of more or less stringent interpretations for the query statements. [14]

Consider, as an example, a document D with assigned terms A and B and let d_A and d_B represent the weights or importance of the two terms in the document, $0 \le d_A, d_B \le 1$. A term weight of 0 indicates that the corresponding term is not assigned to an item; a weight of 1 represents a fully weighted term, and weights between 0 and 1 are partial term assignments. Given queries (A <u>and</u> B) and (A <u>or</u> B), the following query-document similarity functions may be defined between these queries and a document $D = (d_A, d_B)$:

$$\mathrm{sim}(Q_{(A\ \underline{\mathrm{and}}\ B)}, D) = 1 - \left[\frac{(1-d_A)^p + (1-d_B)^p}{2} \right]^{1/p} \tag{3a}$$

$$\mathrm{sim}(Q_{(A\ \underline{\mathrm{or}}\ B)}, D) = \left[\frac{d_A^p + d_B^p}{2} \right]^{1/p} \tag{3b}$$

It is easy to verify that when p is large, the similarity function (3a) produces a value of 1 whenever $d_A = d_B = 1$, and values of 0 whenever one or both of the term weights d_A or d_B are equal to 0. In the same way, the function (3b) produces a value of 1 whenever at least one of the document terms has a value of 1, but a value of 0 whenever both d_A and d_B are equal to 0. In other words, when the document terms are restricted to the binary weights 0 and 1, the matching functions of expression (4) produce exactly the results of the conventional Boolean logic when large p values are used. The corresponding example is included at the bottom of Table 9.

Consider now the case when p is equal to 1. In that case values of 1 are produced by both expression (3a) and (3b) when $d_A = d_B = 1$; when $d_A = d_B = 0$ both (3a) and (3b) are equal to 0; and finally when d_A equals 1 and d_B equals 0 or vice versa, the two matching functions produce values of 1/2. Thus for p=1, identical results are produced by both of the similarity functions of expression (3). This implies that the distinction between the Boolean _and_ and the Boolean _or_ connectives is no longer present. Moreover the actual similarity values obtained for p=1 are exactly those produced by a vector processing system in which the similarity between a document $D = (d_1, d_2, \ldots, d_k)$ and a normalized vector query $Q = (\frac{q_1}{k}, \frac{q_2}{k}, \ldots, \frac{q_k}{k})$ is evaluated as the usual vector product $\sum_{i=1}^{k} \frac{d_i \cdot q_i}{k}$. For the sample document $D = (d_A, d_B)$ and query $Q = (q_A/2, q_B/2)$, the results are reproduced in the top part of Table 9.

When the p-value decreases from infinity to 1, matching systems are obtained which are intermediate between the strict Boolean system ($p=\infty$) and the vector processing system (p=1). That is, the Boolean connectives receive an increasingly lax interpretation as p decreases from infinity; eventually as p reaches a value of 1, the distinction between _and_ and _or_ is lost, and the queries (A _and_ B) and (A _or_ B) are treated simply as the normalized vector query (A/2, B/2). The center portion of Table 9 shows the case for p=2 and binary term weights. In that case for $d_A = d_B = 1$ one still obtains a similarity value of 1 with respect to queries (A _and_ B) and (A _or_ B). When $d_A = d_B = 0$ one again obtains a query-document similarity of 0. When one of the two document terms has a value of 1 and the other a value of 0, retrieval values of $1-1/\sqrt{2}$ are obtained with respect to query (A _and_ B), and $1/\sqrt{2}$ with respect to (A _or_ B). In other words, intermediate values between 1 and 0 are obtained when some of the document terms match the query terms and others do not. [14]

The extended Boolean retrieval system is easily applied to the case where term weights are also attached to the query terms in accordance with the presumed importance of each term in the query. Such query weights can then be used in addition to the weights attached to the document terms. In that case the queries may be formulated as [(A,a) *or* (B,b)] and [(A,a) *and* (B,b)] respectively, with a and b, $0 \le a, b \le 1$, representing the weights of terms A and B in the query. The similarity functions with document $D = (d_A, d_B)$ now become

$$\mathrm{sim}\,(Q_{[(A,a)\ \underline{\mathrm{and}}\ (B,b)]}, D) = 1 - \left[\frac{a^p(1-d_A)^p + b^p(1-d_B)^p}{a^p + b^p}\right]^{1/p} \tag{4a}$$

and

$$\mathrm{sim}\,(Q_{[(A,a)\ \underline{\mathrm{or}}\ (B,b)]}, D) = \left[\frac{a^p d_A^p + b^p d_B^p}{a^p + b^p}\right]^{1/p} \tag{4b}$$

Once again when the query and document term weights are binary (restricted to 0 and 1), a standard Boolean retrieval system is obtained for large p values ($p=\infty$). A standard vector processing system which does not distinguish between *and* and *or* connectives is obtained for p=1, and intermediate systems with loose phrase and synonym interpretations are obtained for intermediate p values.

The extended Boolean retrieval system exhibits the following advantages for a practical implementation:

1) it accommodates the normal query structures used in conventional Boolean retrieval but avoids potentially nonsensical interpretations by letting the query-document similarity depend on the number and the weight of the matching terms;

2) it allows the incorporation of term weights into both documents and queries in accordance with the presumed importance of the terms in the respective constructs;

3) it provides for the retrieval of documents in decreasing order of the query-document similarity, thereby controlling the size of the document set to be retrieved in response to a given query;

4) it facilitates the distinction between compulsory phrase and strict synonym interpretations on the one hand, and tentative phrases and looser synonym relations on the other, by using high p-values to characterize the query structures in the former case and low p-values in the latter.

The feedback operations for the extended Boolean retrieval system may be evaluated by using the same set of 1033 documents in biomedicine and 30 queries previously used for the evaluation of the conventional Boolean feedback system. The relevant output is shown in Table 10(a) for extended queries constructed from the originally available conventional Boolean queries, and in Table 10(b) for extended queries constructed from the natural language query formulations. Evaluation results are included in Table 10 for feedback queries using p-values equal to 1 and 2, and q-count values of 1, 2, or 3. For the experiments of Table 10 common p-values were assigned to all Boolean query operators within a given experimental run, and term relevance weights (see expression (2)) were used for all document terms.

The runs used as a basis of comparisons for the extended system of Table 10 are the same as those used earlier corresponding to the conventional manually constructed Boolean queries used without feedback, or to the vector processing queries without feedback, but with rank freezing for the relevant items retrieved in the top ten ranks. It is clear that the output for the automatically constructed feedback queries in the extended Boolean system ($p = 1$ and $p = 2$) is superior to the comparable output for the automatically produced conventional Boolean feedback queries of Table 8 ($p = \infty$). The best results are obtained for small p values between 1 and 2, where the interpretation of the Boolean operators is most relaxed, using feedback queries consisting of the ored combination of the new feedback formulations plus the old original Boolean queries, and q-count values equal to 2. For such a parameter setting the improvement for the automatic feedback queries in the extended Boolean system over the original manual Boolean queries of Table 10(a) is about 155 percent; the corresponding deficit over the vector processing queries is about 5 percent. For the automatic feedback queries constructed from the natural language formulations shown in Table 10(b), the improvement over the initial Boolean formulation exceeds 100 percent. In that case an improvement of about 10 percent is obtained over the natural language vector queries.

The best feedback queries produce a retrieval effectiveness exceeding the optimal retrospective performance of 0.7074 obtained by using the original query terms only. (The feedback queries have terms extracted from the relevant documents in addition to the original query terms.)

5. Utilization of Automatic Boolean Query Environment

The automatic query construction methods can be incorporated into a practical retrieval environment based on Boolean operations and inverted file methodologies. The following sample process requires no alterations in the basic operational bibliographic retrieval methods, except for the addition of "back-end" programs to carry out the query document comparisons in the extended Boolean system using the formulas of expressions (3) and (4):

1) Given a user search statement in natural language form, a broad <u>conventional</u> Boolean query statement is prepared. Typically such a statement consists of medium frequency single terms. The initial Boolean statement must be sufficiently broad to retrieve most potentially relevant items from a large document collection using conventional inverted file technologies. (Alternatively, if a Boolean query statement is initially available, this statement can be used directly for the retrieval of potentially relevant documents in suitably broadened form.)

2) The broad Boolean query constructed in step 1 is used to retrieve a subcollection of potentially relevant documents by searching the available collections using a conventional inverted file technology. (Subsequent operations performed in the extended Boolean system are carried out only with the subcollection of potentially relevant items identified in this step.)

3) The initial natural language query formulation (or initially available Boolean query) is used to construct an automatic query formulation in the extended Boolean system using the query construction process described in Table 1. Unless special information is available, low p-values (p=1 or p=2) should be used, and term weights should be assigned to the document terms.

4) A new search is performed using the extended Boolean query constructed in step 3 and the documents included in the previously retrieved subcollection produced in step 2. For each item in this subcollection a query-document similarity measure is obtained using expressions (3) and (4), and the documents are ranked and retrieved in decreasing order of the query-document similarity.

5) The top few retrieved items are submitted to the user and any items identified as relevant to the user's needs are utilized to generate an improved automatic query using the procedures illustrated in Tables 3 and 5.

6) The procedures of steps 4 and 5 are repeated until the user is satisfied with the retrieved output.

Since only small test collections are available in the current study, the initial step devoted to identification of a potentially relevant subcollection becomes unnecessary. The foregoing process is therefore simulated by using a set of conventional Boolean queries (either manually constructed or automatically obtained from the natural language statements of user need) to retrieve an initial set of items. The relevant items retrieved in the top ten ranks are used to construct first iteration feedback queries in the extended (p-valued) Boolean system. A new search is now conducted with the first iteration queries, and the effectiveness of this first iteration search is compared with the effectiveness of the initial Boolean queries used in a rank freezing mode as previously explained in Table 7.

The search results obtained with the first iteration queries can be used in turn to construct second and third iteration feedback queries by identifying the new relevant items located in the top ten ranks. The search conducted with the second and third iteration queries is then compared with the results of the first and second iteration queries, respectively, used in the rank freezing mode described earlier.

Three iterations of relevance feedback were carried out experimentally for the Medlars collection of 30 queries and 1033 documents. The experimental process for the first two feedback iterations is summarized in the chart of Fig. 3 together with the main parameter values used in the experimental feedback system. The feedback queries in the extended system were constructed by using an estimated retrieval threshold T of 50 documents, a q-count of 2, and a feedback query defined as either Q_{new} or the set union (Q_{new} or Q_{old}). The feedback queries were run in various modes, the most useful output corresponding to p-values equal to 2 using weighted document terms, but unweighted (binary) query terms.

The search results are shown in Table 11. The two columns of search precision figures in Table 11 correspond to the results obtained with originally available, manually constructed Boolean queries, and with automatically constructed Boolean queries, respectively. The percentage improvements included in Table 11 reflect the advantage of the feedback run compared with the base run from which the feedback queries were built used in a rank freezing mode. For the continuation runs, where the original output ranking is preserved but additional documents are examined, the percentage improvement figures of Table 11 are those produced by the retrieval of additional output without query reformulation.

It may be seen in Table 11(a) that the difference in performance between the original Boolean queries in rank freezing mode and the first iteration feedback queries in the extended Boolean system amounts to about 150 percent for the original manual queries, and about 100 percent for the automatic queries. The inclusion in Table 11 of a conventional Boolean first iteration feedback query ($p = \infty$, binary weights) shows that about half of the improvement in the first iteration feedback step is due to the feedback effect (that is, the new terms included in the feedback queries obtained from previously identified relevant items), the other half being due to the use of the extended Boolean system including the term weights and the lower p-values.

The performance results for the second and third feedback iterations are included in Table 11(b). The performance improvements in this case are due entirely to the feedback effect, since extended, rather than conventional queries are used for both the first and second iteration feedback queries. For the manually constructed queries, Table 11(b) shows that the second feedback iteration brings an additional 15 percent improvement over the continued search using the first iteration feedback queries, while the third feedback iteration adds another 2 percent over the continued search with the second iteration queries. The corresponding results for the automatic queries are 3 percent and 0 percent, respectively. The actual precision values obtained after two feedback iterations are equal to .7668 and .8234 for manual and automatic queries, respectively. This increases to .8058 and .8734 after three iterations of feedback for the two query types. In both cases, this represents a high order of performance which substantially exceeds the results obtainable with the automatic vector processing system using term weights, a cosine query-document similarity measure, but no Boolean operators. A summary of the percentage improvements obtainable by the three feedback iterations for the two types of Medlars queries is contained in Table 11(c).

The automatic relevance feedback process for Boolean queries described in this study is easily incorporated in conventional Boolean retrieval operations. Based on the experimental retrieval output included in this study, it appears that two iterations of feedback suffice to provide a very high order of performance.

References

[1] J.J. Rocchio, Jr., Relevance Feedback in Information Retrieval, in The Smart Retrieval System - Experiments in Automatic Document Processing, G. Salton, editor, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[2] E. Ide and G. Salton, Interactive Search Strategies and Dynamic File Organization in Information Retrieval, in The Smart Retrieval System - Experiments in Automatic Document Processing, G. Salton, editor, Englewood Cliffs, New Jersey, 1971.

[3] C. Vernimb, Automatic Query Adjustment in Document Retrieval, Information Processing and Management, Vol. 13, No. 6, 1977, p. 339-353.

[4] J.T. Rickman, Design Considerations for a Boolean Feedback System with Automatic Relevance Feedback Processing, Proceedings of the ACM Annual Conference, Association for Computing Machinery, New York, August 1972, p. 478-482.

[5] M. Dillon and J. Desper, The Use of Automatic Relevance Feedback in Boolean Retrieval Systems, Journal of Documentation, Vol. 36, No. 3, September 1980, p. 197-208.

[6] K. Sparck Jones, A Statistical Interpretation of Term Specificity and its Application in Retrieval, Journal of Documentation, Vol. 28, 1972, p. 11-21.

[7] G. Salton, C.S. Yang and C.T. Yu, A Theory of Term Importance in Automatic Text Analysis, Journal of the ASIS, Vol. 26, No. 1, January-February 1975, p. 33-44.

[8] G. Salton, C. Buckley, and E.A. Fox, Automatic Query Formulations in Information Retrieval, Technical Report 82-524, Department of Computer Science, Cornell University, Ithaca, New York, October 1982.

[9] S.E. Robertson and K. Sparck Jones, Relevance Weighting of Search Terms, Journal of the ASIS, Vol. 27, No. 3, 1976, p. 129-146.

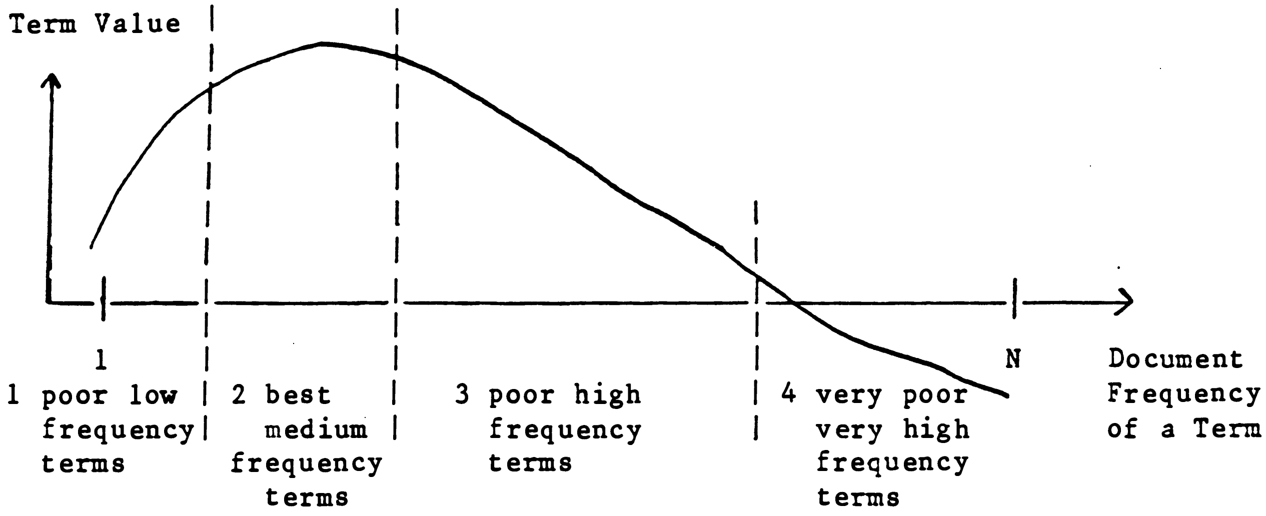
[10] K. Sparck Jones, Experiments in Relevance Weighting of Search Terms, Information Processing and Management, Vol. 15, No. 3, 1979, p. 133-144.

[11] M.F. Porter, Implementing a Probabilistic Information Retrieval System, Information Technology: Research and Development, Vol. 1, No. 2, April 1982, p. 131-156.

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

[13] Y.K. Chang, C. Cirillo and J. Razon, Evaluation of Feedback Retrieval Using Modified Freezing, Residual Collection and Test and Control Groups, in The Smart Retrieval System - Experiments in Automatic Document Processing, G. Salton, editor, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1971.

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



1 poor low frequency terms are broadened by including them in *or*-groups (A *or* B *or* C *or* ...) with other low frequency terms

2 medium frequency terms are used as single terms

3 poor high frequency terms are rendered more specific by using them as *anded* term pairs (A *and* B)

4 very high frequency terms are rendered more specific by including them in larger *and*-groups (A *and* B *and* C)

Query Formation Strategy Using Term Frequency Considerations

Fig. 1

Table 8(a)

Use the initially available Boolean query formulations

Table 8(b)

Take the initially available natural language formulation of user needs and automatically generate conventional Boolean queries using the process of Table 1

Initial Search Run

Perform retrieval run A by comparing the Boolean queries with the document collection

Construction of Feedback Queries

Identify the relevant items retrieved in the top ten ranks and use the vocabulary in these relevant documents to construct first iteration feedback queries by the process illustrated in Table 2-5

First Feedback Iteration

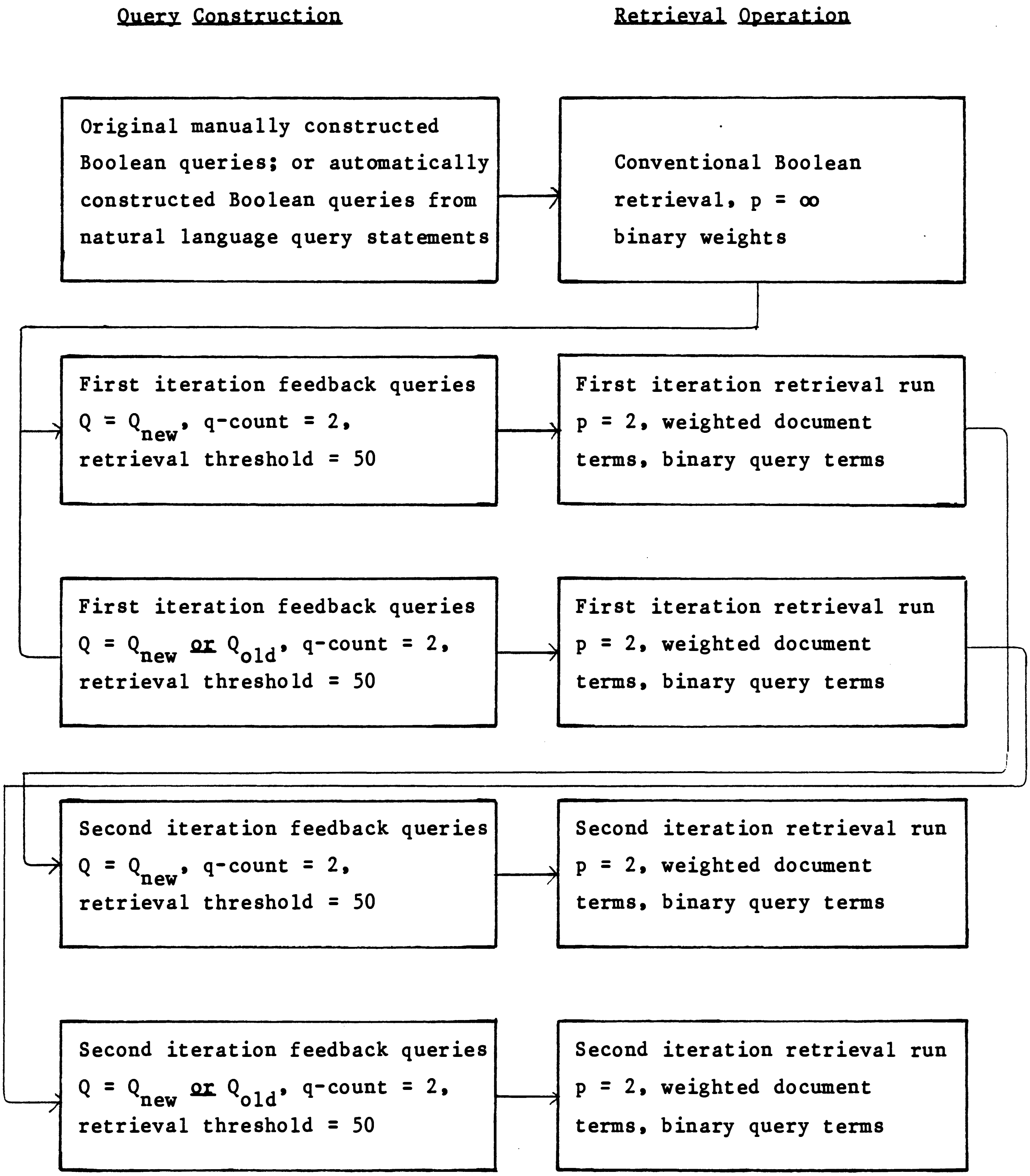
Perform retrieval run B by comparing the feedback queries with the document collection

Evaluation

Evaluate the effectiveness of the feedback run in terms of recall and precision by comparing the output of run B with the output of the "continued" run A (relevant items in top 10 receive frozen ranks, nonrelevant items in top 10 are deleted and replaced by new items)

Evaluation Process for Boolean Feedback System

Fig. 2



Basic Parameter Assignments for Experimental Feedback Operations

Fig. 3

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

Terms	Document Frequency $(n_i, \; n_i \cdot \frac{n_j}{N}, \; n_i \cdot n_j \cdot \frac{n_k}{N^2})$	Term Weight $(1-\frac{n_i}{N}, \; 1-\frac{n_{ij}}{N}, \; 1-\frac{n_{ijk}}{N})$
excretion (ex)	52	.9497
phosphate (ph)	43	.9584
urine (ur)	78	.9245
(ex and ph)	2.2	.9979
(ex and ur)	3.9	.9962
(ph and ur)	3.2	.9969
(ex and ph and ur)	0.2	.9998

a) Term Characteristics for Initial Query Construction

Query Formulation	Expected Number of Retrieved Items
(ex and ph) or (ex and ur) or (ph and ur)	9.3
(ex and ph) or (ph and ur)	5.4

b) Initial Query Formulations for Expected Number Retrieved Between 5 and 10

Original Query Construction

Table 1

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

<u>Relevant Document</u>: Actinomycin (ac) D and the response (res) to parathyroid (pa) hormone (ho). Actinomycin (ac) D inhibits (in) the effect (ef) or parathyroid (pa) hormone (ho) upon bone (bo) without altering (al) its effect (ef) upon the renal (re) excretion (ex) of phosphate (ph)

Term	Document Frequency n_i	Occurs in Doc	Occurs in Query	Number of Relevant Items r_i	Relevance Frequencies $rf_i = (\frac{r_i}{R} - \frac{n_i}{N})$	Relevance Weights $w_i = rf_i(1-\frac{n_i}{N})$
excretion (ex)	52	✓	✓	2	.9497	.9019
phosphate (ph)	43	✓	✓	2	.9584	.9185
urine (ur)	78		✓	1	.4245	.3925
actinomycin (ac)	8	✓		1	.4923	.4885
response (res)	162	✓		1	.3432	.2894
parathyroid (pa)	27	✓		1	.4739	.4615
hormone (ho)	81	✓		1	.4216	.3885
bone (bo)	66	✓		1	.4361	.4082
altering (al)	69	✓		1	.4332	.4043
effect (ef)	248	✓		1	.2599	.1975
renal (re)	76	✓		1	.4264	.3950

(a) Computation of Relevance Weights for Single Terms

	Term Pair	Document Frequency $n_{ij} = \frac{n_i \cdot n_j}{N}$	Number of Relevant Items r_{ij}	Relevance Frequencies $rf_{ij} = (\frac{r_{ij}}{R} - \frac{n_{ij}}{N})$	Relevance Weights $w_{ij} = rf_{ij}(1-\frac{n_{ij}}{N})$
1	ex-ph	2.16	2	.9979	.9958
2	ex-ur	3.92	1	.4962	.4943
3	ex-ac	0.40	1	.4996	.4994
4	ex-res	8.15	1	.4921	.4882
5	ex-pa	1.36	1	.4987	.4981
6	ex-ho	4.08	1	.4961	.4942
7	ex-bo	3.32	1	.4968	.4952
8	ex-al	3.47	1	.4966	.4949
9	ex-ef	12.48	1	.4879	.4820
10	ex-re	3.83	1	.4963	.4945
11	ph-ur	3.25	1	.4969	.4954
12	ph-ac	0.33	1	.4997	.4996
13	ph-res	6.74	1	.4935	.4903
14	ph-pa	1.12	1	.4989	.4984
15	ph-ho	3.37	1	.4967	.4951
16	ph-bo	2.75	1	.4973	.4960
17	ph-al	2.87	1	.4972	.4958
18	ph-ef	10.32	1	.4900	.4851
19	ph-re	3.16	1	.4969	.4954

(b) Computations of Relevance Weights for Cooccurring Term Pairs

Highly Co-Occurring Term Triples	Document Frequency $n_{ijk} = \frac{n_i \cdot n_j \cdot n_k}{N^2}$	Number of Relevant Items r_{ijk}	Relevance Frequency $rf_{ijk} = (\frac{r_{ijk}}{R} - \frac{n_{ijk}}{N})$	Relevance Weight $rf_{ijk}(1-\frac{n_{ijk}}{N})$
ex-ph-res	.3387	1	.4997	.4996
ex-ph-ef	.5186	1	.4995	.4993

(c) Computation of Relevance Weights for Highly Co-occurring Term Triples

Relevance Weight Computations

Table 2

Query Formulation Process	Query Formulation	Expected Number of Retrieved Items $\Sigma n_i + \Sigma n_{ij}$
Use single terms of highest weight plus term pairs not covered by the single terms	(ex or ph or ac or pa or bo or al or (ex and ur) or (ex and res) or (ex and ho) or (ex and ef) or (ex and re) or (ph and ur) or (ph and res) or (ph and ho) or (ph and ef) or (ph and re)	324.30
Remove single term of lowest weight (al) and add pairs not covered (ex and al),(ph and al)		261.64
Consecutively remove single terms in increasing weight order and add pairs not covered until a query is obtained containing all 19 pairs	[ex and (ph or ur or ac or res or pa or ho or bo or al or ef or re)] or [ph and (ur or ac or res or pa or ho or bo or al or ef or re)]	77.08
Remove term pair of lowest weights (ex and ef)		64.60
Successively remove term pairs in increasing weight order; add any triples not covered. Ten pairs are left following removal of the 9 pairs with lowest weight	(ex and ph) or (ex and ac) or (ex and pa) or (ex and bo) or (ph and ur) or (ph and ac) or (ph and pa) or (ph and bo) or (ph and al) or (ph and re)	20.72
Remove next three term pairs of lowest weight (ex and bo), (ph and re), (ph and ur) to reach expected retrieval level of 10 items	(ex and ph) or (ex and ac) or (ex and pa) or (ph and ac) or (ph and pa) or (ph and bo) or (ph and al)	10.99

Query Construction Process (Query Count = 1)

Table [illegible]

Original [illegible]: Excretion (ex) of phosphate (ph) in urine (ur)

Term	Document Frequency n_i or n_{ij}	Occurs in Doc	Query	Number of Relevant Items r_i or r_{ij}	Relevance Frequencies rf_i $=(\frac{r_i}{R}-\frac{n_i}{N})$	Relevance Weights $w_i = rf_i(1-\frac{n_i}{N})$
excretion (ex)	52	1	2	3	.9497	.9019
phosphate (ph)	43	1	2	3	.9584	.9185
urine (ur)	78	-	2	2	.5845	.5404
ur-ex	3.93			2	.6562	.6537
ur-ph	3.25			2	.6569	.6549
ex-ph	2.16			3	.9979	.9958
ex-ac	0.40			1	.3296	.3295
ex-pa	1.36			1	.3287	.3283
ex-bo	3.32			1	.3268	.3258
ph-ac	0.33			1	.3297	.3296
ph-pa	1.12			1	.3289	.3285
ph-bo	2.75			1	.3273	.3264
ph-al	2.87			1	.3272	.3263
ph-re	3.16			1	.3269	.3259

Computation of Relevance Weights Assuming Query Count of 2

(selected terms only)

Table 4

Query Formulation Process	Query Formulation	Expected Number of Retrieved Items
Use 11 pairs of highest weight	(ur <u>and</u> ex) <u>or</u> (ur <u>and</u> ph) <u>or</u> (ex <u>and</u> ph) <u>or</u> (ex <u>and</u> ac) <u>or</u> (ex <u>and</u> pa) <u>or</u> (ex <u>and</u> bo) <u>or</u> (ph <u>and</u> ac) <u>or</u> (ph <u>and</u> pa) <u>or</u> (ph <u>and</u> bo) <u>or</u> (ph <u>and</u> al) <u>or</u> (ph <u>and</u> re)	24.65
Remove pair of lowest weight (ex <u>and</u> bo)		21.33
Remove pair of next lowest weight (ph <u>and</u> re)	(ur <u>and</u> ex) <u>or</u> (ur <u>and</u> ph) <u>or</u> (ex <u>and</u> ph) <u>or</u> (ex <u>and</u> ac) <u>or</u> (ex <u>and</u> pa) <u>or</u> (ph <u>and</u> ac) <u>or</u> (ph <u>and</u> pa) <u>or</u> (ph <u>and</u> al) <u>or</u> (ph <u>and</u> bo)	18.17
Remove four more pairs of lowest weight to reach retrieval threshold of 10 (ph-al), (ph-bo), (ex-pa), (ph-po)	(ur <u>and</u> ex) <u>or</u> (ur <u>and</u> ph) <u>or</u> (ex <u>and</u> ph) <u>or</u> (ex <u>and</u> ac) <u>or</u> (ph <u>and</u> ac)	10.07

Query Construction Process (Query Count = 2)

Table 5

<u>Initial Query</u> (natural language):

Excretion (ex) of phosphate (ph) in urine (ur)

<u>Original Automatic Query</u> (expected retrieval items: 9.3)

(ex <u>and</u> ph) <u>or</u> (ex <u>and</u> ur) <u>or</u> (ph <u>and</u> ur)
.9979 .9962 .9969

<u>Feedback Query</u> (<u>Query Count</u> = <u>1</u>) (expected retrieved items: 10.99)

$\boxed{\text{(ex and ph) .9958}}$ <u>or</u> (ex <u>and</u> ac) <u>or</u> (ex <u>and</u> pa)
.4994 .4981

<u>or</u> (ph <u>and</u> ac) <u>or</u> (ph <u>and</u> pa) <u>or</u> (ph <u>and</u> bo)
.4996 .4984 .4960

<u>or</u> (ph <u>and</u> al)
.4958

<u>Feedback Query</u> (<u>Query Count</u> = <u>2</u>) (expected retrieved items: 10.07)

$\boxed{\text{(ex and ph) .9958}}$ <u>or</u> $\boxed{\text{(ex and ur) .6537}}$ <u>or</u> $\boxed{\text{(ph and ur) .6549}}$

<u>or</u> (ex <u>and</u> ac) <u>or</u> (ph <u>and</u> ac)
.3295 .3296

<u>Added Low Frequency Terms from Relevant Retrieved Items</u>

actinomycin (8), parathyroid (27), bone (66), altering (69)

Query Construction Summary

Table 6

Document Rank	Document Name	Relevant or Not	Document Rank	Document Name	Relevant or Not	Document Rank	Document Name	Relevant or Not
1	a (remove)		1	f	R	1	j	R
2	b	R	2	b	R	2	b	R
3	c (remove)		3	g		3	i	
4	d	R	4	d	R	4	d	R
5	e (remove)		5	h		5	f	R
6	f	R	6	i		6	g	
7	g	—	7	j	R	7	k	
8	h	—	8			8	p	
9	i	—	9	.	.	9	m	R
10	j	R	10	.	.	10	h	

a) Initial Retrieval Ranks (Run A) (user looks at top 5 items and and identifies items a,d as relevant)

b) Initial Rank Continuation (retrieval ranks of relevant in top 5 are frozen; items of rank below 5 are promoted and replace the nonrelevant originally located in top 5)

c) First Interaction Feedback Query constructed from Items a,d (Run B) (retrieved items are ranked in decreasing query-document similarity except for relevant items in frozen ranks)

Relevance Feedback Evaluation with Rank Freezing

Table 7

Medlars 1033, 30 Queries

Type of Run			Average Precision at three Recall Points	Percent Difference from Base Case	Percent Difference from Cosine
Original manually prepared Boolean Queries (unweighted terms) used without Feedback but with Rank Freezing of Relevant Items in Top Ten (Run A)			.2372	-	-63.1%
Original Natural Language Queries using Vector Processing System, Cosine Similarity Function, Weighted Terms, Used without Feedback but with Rank Freezing of Relevant Top Ten			.6427	+171.0%	-
Automatically Formed Feedback Queries from Original Manual Boolean Queries, Retrieval Threshold = 50 (Run B)					
$p=\infty$, binary weights,	q count=1	$Q=Q_{new}$	.3977	+67.7%	-39.1%
		$Q=Q_{new}$ or Q_{old}	.3992	+68.3	-38.8
	q count=2	$Q=Q_{new}$	.4345	+83.2	-32.4
		$Q=Q_{new}$ or Q_{old}	.4322	+82.2	-38.8
	q count=3	$Q=Q_{new}$	.4306	+81.5	-33.0
		$Q=Q_{new}$ or Q_{old}	.4387	+84.9	-31.7
Optimal Case Using Retrospective Term Relevance Weights			.7074	+204.0	+10.1

a) Comparison of Standard Boolean Feedback with Manually Constructed Boolean Queries

Type of Run			Average Precision at three Recall Points	Percent Difference from Base Case	Percent Difference from Cosine
Automatically Constructed Boolean Queries Using Singles, Pairs and Triples, and a Desired Retrieval Threshold of 50 Documents. Unweighted terms, without Feedback but with Rank Freezing of Relevant on Top Ten (Run A)			.3552	-	-45.7
Original Natural Language Queries Using Vector Processing System, Cosine Similarity Function, Weighted Terms, Used without Feedback but with Rank Freezing of Relevant Top Ten			.6427	+81.0	-
Automatically Formed Feedback Queries from Original Automatic Queries. Unweighted Terms Retrieval Threshold = 50 (Run B)					
$p=\infty$, binary weights,	q count=1	$Q=Q_{new}$	.5035	+41.7%	-21.7
		$Q=Q_{new}$ or Q_{old}	.4883	+26.2	-30.2
	q count=2	$Q=Q_{new}$	.5009	+41.0	-22.1
		$Q=Q_{new}$ or Q_{old}	.4628	+30.3	-28.0
	q count=3	$Q=Q_{new}$	.4972	+40.0%	-22.6
		$Q=Q_{new}$ or Q_{old}	.4442	+25.1	-30.9
Optimal Case Using Retrospective with Term Relevance Weights			.7074	+99.2	+10.1

b) Comparison of Standard Boolean Feedback Queries with Automatically Formed Boolean Queries

Evaluation of Conventional Boolean Relevance Feedback Process

Table 8

Assume $D = (d_A, d_B)$ $\quad 0 \le d_A,\ d_B \le 1$

$Q = (A\ \underline{and}\ B)$ or $Q = (A\ \underline{or}\ B)$ (unweighted query terms)

$$Sim(D, Q_{A\ \underline{and}\ B}) = 1 - \left[\frac{(1-d_A)^p + (1-d_B)^p}{2}\right]^{1/p} \qquad Sim(D, Q_{A\ \underline{or}\ B}) = \left[\frac{d_A^p + d_B^p}{2}\right]^{1/p}$$

$p = 1$	$d_A=1, d_B=1$	$Sim(D,Q) = 1$	$p = 1$	$d_A=1, d_B=1$	$Sim(D,Q) = 1$
	$d_A=1, d_B=0$ $d_A=0, d_B=1$	$Sim(D,Q) = 1/2$		$d_A=1, d_B=0$ $d_A=0, d_B=1$	$Sim(D,Q) = 1/2$
	$d_A=0, d_B=0$	$Sim(D,Q) = 0$		$d_A=0, d_B=0$	$Sim(D,Q) = 0$
$p = 2$	$d_A=1, d_B=1$	$Sim(D,Q) = 1$	$p = 2$	$d_A=1, d_B=1$	$Sim(D,Q) = 1$
	$d_A=1, d_B=0$ $d_A=0, d_B=1$	$Sim(D,Q) = 1-1/\sqrt{2}$		$d_A=1, d_B=0$ $d_A=0, d_B=1$	$Sim(D,Q) = 1/\sqrt{2}$
	$d_A=0, d_B=0$	$Sim(D,Q) = 0$		$d_A=0, d_B=0$	$Sim(D,Q) = 0$
$p = \infty$	$d_A=1, d_B=1$	$Sim(D,Q) = 1$	$p = \infty$	$d_A=1, d_B=1$	$Sim(D,Q) = 1$
	$d_A=1, d_B=0$ $d_A=0, d_B=1$	$Sim(D,Q) = 0$		$d_A=1, d_B=0$ $d_A=0, d_B=1$	$Sim(D,Q) = 1$
	$d_A=0, d_B=0$	$Sim(D,Q) = 0$		$d_A=0, d_B=0$	$Sim(D,Q) = 0$

Query-Document Similarity for Unweighted Queries

$D = (d_A, d_B)$; $Q = (A\ \underline{and}\ B)$, $Q = (A\ \underline{or}\ B)$

Table 9

Medlars 1033, 30 Queries

Type of Run			Average Precision at three Recall Points	Percent Difference from Base Case	Percent Difference from Cosine
Original manually prepared Boolean Queries (unweighted terms) used without Feedback but with Rank Freezing of Relevant Items in Top Ten			.2372	-	-63.1%
Original Natural Language Queries using Vector Processing System, Cosine Similarity Function, Weighted Terms, Used without Feedback but with Rank Freezing of Relevant Top Ten			.6427	+171.0%	-
Automatically Formed Feedback Queries from Original Manual Boolean Queries, Retrieval Threshold = 50.					
p=1, binary weights,	q count=1	$Q=Q_{new}$	.4562	+92.3%	-29.0%
		$Q=Q_{new}$ <u>or</u> Q_{old}	.5592	+135.8	-13.0
	q count=2	$Q=Q_{new}$	.4856	+104.7	-24.4
		$Q=Q_{new}$ <u>or</u> Q_{old}	.5609	+136.5	-12.7
	q count=3	$Q=Q_{new}$	.4472	+ 88.5	-30.4
		$Q=Q_{new}$ <u>or</u> Q_{old}	.5481	+131.1	-14.7
p=1, weighted doc. terms, binary query terms	q count=1	$Q=Q_{new}$	.5117	+115.7	-20.4
		$Q=Q_{new}$ <u>or</u> Q_{old}	.6130	+158.4	- 4.6
	q count=2	$Q=Q_{new}$	.5408	+128.0	-15.9
		$Q=Q_{new}$ <u>or</u> Q_{old}	.5977	+152.0	- 7.0
	q count=3	$Q=Q_{new}$	.5352	+125.6	-16.7
		$Q=Q_{new}$ <u>or</u> Q_{old}	.5934	+150.2	- 7.7
p=2, binary weights,	q count=1	$Q=Q_{new}$	.4889	+106.1%	-23.9%
		$Q=Q_{new}$ <u>or</u> Q_{old}	.5635	+137.5	-12.3
	q count=2	$Q=Q_{new}$	.5065	+113.5	-21.2
		$Q=Q_{new}$ <u>or</u> Q_{old}	.5701	+140.3	-11.3
	q count=3	$Q=Q_{new}$	.4903	+106.7	-23.7
		$Q=Q_{new}$ <u>or</u> Q_{old}	.5776	+143.5	-10.1
p=2, weighted doc. terms, binary query terms	q count=1	$Q=Q_{new}$	.5528	+133.1	-14.0
		$Q=Q_{new}$ <u>or</u> Q_{old}	.6086	+156.6	- 5.3
	q count=2	$Q=Q_{new}$	.5695	+140.1	-11.4
		$Q=Q_{new}$ <u>or</u> Q_{old}	.6103	+157.3	- 5.0
	q count=3	$Q=Q_{new}$	.5598	+136.0	-12.9
		$Q=Q_{new}$ <u>or</u> Q_{old}	.6075	+156.1	- 5.5
Optimal Case Using Retrospective Term Relevance Weights			.7074	+204.0	+10.1

Evaluation of Extended Boolean Feedback System Starting with

Manually Constructed Boolean Queries

Table 10(a)

Medlars 1033, 30 Queries

Type of Run			Average Precision at three Recall Points	Percent Difference from Base Case	Percent Difference from Cosine
Automatically Constructed Boolean Queries Using Singles, Pairs and Triples, and a Desired Retrieval Threshold of 50 Documents. Unweighted terms, without Feedback but with Rank Freezing of Relevant on Top Ten			.3552	-	-45.7
Original Natural Language Queries Using Vector Processing System, Cosine Similarity Function, Weighted Terms, Used without Feedback but with Rank Freezing of Relevant Top Ten			.6427	+81.0	-
Automatically Formed Feedback Queries from Original Automatic Queries. Unweighted Terms Retrieval Threshold = 50.					
p=1, binary weights,	q count=1	$Q=Q_{new}$	.5702	+60.5%	-11.3
		$Q=Q_{new}$ <u>or</u> Q_{old}	.6325	+78.1	- 1.6
	q count=2	$Q=Q_{new}$	.6255	+76.1	- 2.7
		$Q=Q_{new}$ <u>or</u> Q_{old}	.6265	+76.4	- 2.5
	q count=3	$Q=Q_{new}$	.5931	+67.0	- 7.7
		$Q=Q_{new}$ <u>or</u> Q_{old}	.6115	+72.1	- 4.8
p=1, weighted doc. terms, binary query terms	q count=1	$Q=Q_{new}$	.6516	+83.4	+ 1.4
		$Q=Q_{new}$ <u>or</u> Q_{old}	.7024	+97.7	+ 9.3
	q count=2	$Q=Q_{new}$	.6928	+95.0	+ 7.8
		$Q=Q_{new}$ <u>or</u> Q_{old}	.6985	+96.6	+ 8.7
	q count=3	$Q=Q_{new}$	.6911	+94.6	+ 7.5
		$Q=Q_{new}$ <u>or</u> Q_{old}	.6938	+95.3	+ 8.0
p=2, binary weights,	q count=1	$Q=Q_{new}$	.6183	+74.1	- 3.8
		$Q=Q_{new}$ <u>or</u> Q_{old}	.6682	+88.1	+ 4.0
	q count=2	$Q=Q_{new}$	.6495	+82.8	+ 1.1
		$Q=Q_{new}$ <u>or</u> Q_{old}	.6629	+86.6	+ 3.2
	q count=3	$Q=Q_{new}$	.6345	+78.6	+ 0.8
		$Q=Q_{new}$ <u>or</u> Q_{old}	.6481	+82.4	+ 8.7
p=2, weighted doc. terms, binary query terms	q count=1	$Q=Q_{new}$	.6890	+94.0	+ 7.2
		$Q=Q_{new}$ <u>or</u> Q_{old}	.7155	+101.4	+11.3
	q count=2	$Q=Q_{new}$	.7067	+ 98.9	+10.0
		$Q=Q_{new}$ <u>or</u> Q_{old}	.7173	+100.8	+11.0
	q count=3	$Q=Q_{new}$	.6984	+96.6	+ 8.7
		$Q=Q_{new}$ <u>or</u> Q_{old}	.6992	+96.8	+ 8.8
Optimal Case Using Retrospective with Term Relevance Weights			.7074	+99.2	+10.1

Evaluation of Automatic Boolean Feedback Starting with

Automatically Formed Boolean Queries

Table 10(b)

Medlars 1033, 30 Queries

Type of Run	Average Precision and Percent Improvements for Initially Available Boolean Queries	Average Precision and Percent Improvements for Automatic Boolean Queries Built from Natural Language Input
Run A Basic Boolean Queries (orginally available or automatically built from natural language input)	Improvement of continuation with partial rank freezing	
original run $p=\infty$ binary weights	.2065	.2899
original run continued once with partial rank freezing	.2372 (+15%)	.3552 (+22.5%)
Original Natural Language Queries using vector processing system with cosine similarity and weighted terms		
original run	.5473	.5473
original run continued once with partial rank freezing	.6427 (+17%)	.6427 (+17%)
Run B First Iteration Feedback Queries from Conventional Boolean Output $Q=Q_{new}$, q-count=2, threshold=50	Improvement over Original Queries, continued once, used in rank freezing mode	
run as strict Boolean ($p=\infty$, binary weights)	.4345 (+83%)	.5009 (+41%)
run as p-valued ($p=2$, weighted document terms)	.5695 (+140%)	.7067 (+99%)
First Iteration Feedback Queries $Q=Q_{new}$ **or** Q_{old}, q-count=2		
run as strict Boolean ($p=\infty$, binary weights)	.4322 (+82%)	.4628 (+30%)
run as p-valued ($p=2$, weighted document terms)	.6103 (+157%)	.7131 (+101%)

Evaluation of Boolean Feedback Process (First Iteration)

Table 11(a)

Medlars 1033, 30 Queries

Type of Run	Average Precision and Percent Improvements for Initially Available Boolean Queries	Average Precision and Percent Improvements for Automatic Boolean Queries Built from Natural Language Input
First Iteration Feedback Queries continued once with partial rank freezing		
$Q = Q_{new}$, p=2, weighted document terms	.6199 (+9%)	.7902 (+12%)
$Q = Q_{new}$ <u>or</u> Q_{old}, p=2, weighted terms	.6672 (+9%)	.8014(+12%)
Original vector queries using cosine similarity and weighted terms continued twice with rank freeze	.7023 (+9%)	.7023 (+9%)
<u>Run C</u> Second Iteration Feedback Queries from First Iteration p-Valued Queries (p=2, weighted document terms)		
$Q = Q_{new}$, q-count=2, threshold=50 run as p-valued, (p=2, weighted terms)	.7221 (+16%)	.8044 (+2%)
$Q = Q_{new}$ <u>or</u> Q_{old}, q-count=2, threshold=50 runs as p-valued, p=2, weighted terms	.7668 (+15%)	.8234 (+3%)
Second Iteration Feedback Queries continued once with partial rank freezing		
$Q = Q_{new}$, p=2, weighted terms	.7571 (+5%)	.8485 (+5%)
$Q = Q_{new}$ <u>or</u> Q_{old}, p=2, weighted terms	.8081 (+5%)	.8701(+6%)
Original vector queries using cosine similarity continued three times with rank freeze	.7376 (+5%)	.7376 (+5%)
<u>Run D</u> Third Iteration Feedback Queries from Second Iteration p-valued Queries (p=2, weighted document terms)		
$Q = Q_{new}$, q-count=2, threshold=50 run as p-valued, p=2, weighted terms	.7729 (+2%)	.8498 (0%)
$Q = Q_{new}$ <u>or</u> Q_{old}, q-count=2, threshold=50 run as p-valued, p=2, weighted terms	.8058 (0%)	.8734 (0%)

Evaluation of Boolean Feedback Process (Second and Third Iteration)

Table 11(b)

Medlars 1033, 30 Queries	Initially Available Boolean Queries	Automatic Boolean Queries built from Natural Language
Improvement of First Feedback Iteration over First Continuation		
feedback effect (new terms)	+83%	+41%
effect of extended system (p-value, term weights)	+57%	+58%
Improvement of Second Continuation over First Feedback Iteration	+ 9%	+12%
Improvement of Second Feedback Iteration over Second Continuation	+15%	+ 3%
Improvement of Third Continuation over Second Feedback Iteration	+ 5%	+ 5%
Improvement of Third Feedback Iteration over Third Continuation	+ 2%	0%

Summary of Improvements for Boolean Feedback Process

Table 11(c)