Automatic Query Formulations in Information Retrieval*

G. Salton, C. Buckley, and E. A. Fox
Department of Computer Science, Cornell University, Ithaca, NY 14853

Modern information retrieval systems are designed to supply relevant information in response to requests received from the user population. In most retrieval environments the search requests consist of keywords, or index terms, interrelated by appropriate Boolean operators. Since it is difficult for untrained users to generate effective Boolean search requests, trained search intermediaries are normally used to translate original statements of user need into useful Boolean search formulations. Methods are introduced in this study which reduce the role of the search intermediaries by making it possible to generate Boolean search formulations completely automatically from natural language statements provided by the system patrons. Frequency considerations are used automatically to generate appropriate term combinations as well as Boolean connectives relating the terms. Methods are covered to produce automatic query formulations both in a standard Boolean logic system, as well as in an extended Boolean system in which the strict interpretation of the connectives is relaxed. Experimental results are supplied to evaluate the effectiveness of the automatic query formulation process, and methods are described for applying the automatic query formulation process in practice.

1. The Conventional Retrieval Environment

In operational information retrieval, Boolean query formulations are used to express the customers' information needs. The standard rules of Boolean logic may not, however, provide an ideal environment for the formulation of effective search requests. For one thing, the conventional Boolean logic assigns a strict interpretation to the connectives *and*, *or*, and *not*, which produces unwanted retrieval results in certain circumstances. Thus, in response to an *or*-query such as "*A or B or ... or Z*," a document containing all of the terms specified in the query will not be rated more highly than a document containing only one of the query terms. Analogously, in response to an *or*-query such as "*A and B and ... or Z*," a document containing all but one of the query terms is deemed to be just as useless as a document containing none of the query terms.

Such a situation would not necessarily be injurious if the average retrieval system user were familiar with the rules of Boolean logic and could evaluate the occurrence characteristics of the query terms in the document collections under consideration. In that case, it would be possible to assess the likelihood of co-occurrence of two or more terms in the documents of a collection and to reach a decision whether such terms ought to be treated as synonymous by including them in an *or*-clause, or as members of a phrase by using an *and* connective to relate them. Unfortunately, there exists much evidence to show that ordinary users are unable to master the complications of Boolean logic, and even professional indexers and searchers find it difficult to construct consistently effective index representations and search statements [1–3].

One possible solution consists in giving up the Boolean logic entirely and using a *vector processing* retrieval model [4,5]. In that case, both the stored records and the information requests are unstructured and expressed simply by sets of keywords of varying lengths. That is, instead of using a query such as "*A and B*" or "*A or B*" one uses simply "*A,B*", the assumption being that the presence of both search terms in a query is worth more than the presence of only one of the terms. In the vector processing system, both the query and the document terms can be weighted in accordance with the presumed importance of the terms, and a similarity computation between queries and stored records makes it possible to obtain ranked output in decreasing order of the query-document similarity. By choosing an appropriate retrieval threshold, the user can then obtain as much or as little output as desired.

The vector processing system compensates for the lack of query structure by the much larger number of terms which are normally used to represent the stored documents and the search queries, and by the ability to assign term weights and to obtain ranked retrieval output. Un-

*This study was supported in part by the National Science Foundation under Grant No. IST-80-17589.

Received October 28, 1982; revised December 8, 1982; accepted December 27, 1982

fortunately, in the vector processing system it is not possible to distinguish query phrases using *and* connectives, such as "information *and* retrieval" from synonymous constructions such as "hand-held calculators *or* pocket computers *or* microcomputers." In addition, the operations required by a vector processing retrieval system may not be compatible with the methodologies already implemented in existing, operational retrieval environments.

The question then arises whether viable methods can be devised for an automatic formulation of Boolean queries, or at least of structured queries compatible with the established retrieval methodologies. In that case, a user's initial natural language statement of information need could be taken as input to an automatic query formulation process. Alternatively, the texts of previously retrieved items judged to be relevant to a given query could be used to reformulate or improve a given available Boolean query statement. This problem is considered in the remainder of this study.

2. Automatic Formulation of Conventional 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 operators to generate a formulation which is not so broad as to retrieve an unreasonable amount of extraneous matter thereby causing a loss in search precision, nor so narrow as to reject a large number of relevant items thereby causing a loss in search recall. A term that is considered to be too broad can be rendered more specific by including it in an *and*-clause, such as (*A and B*); analogously, a term considered to be too specific can be broadened by using it in an *or*-clause together with other related, specific terms.

These considerations suggest that it may be possible to generate automatic Boolean query formulations in "disjunctive normal form" by using Boolean *or* connectives to relate clauses of terms consisting of single terms of the correct specificity, or *anded* combinations of broader terms that need to be rendered more specific. A typical query could then be formulated as

$$S_i \text{ or } S_j \text{ or } \ldots \text{ or } (P_{k1} \text{ and } P_{k2}) \text{ or } (P_{m1} \text{ and } P_{m2}) \text{ or } \ldots$$

$$\text{or } (T_{n1} \text{ and } T_{n2} \text{ and } T_{n3}) \text{ or } (T_{p1} \text{ and } T_{p2} \text{ and } T_{p3}) \text{ or } \ldots$$

where S_i designates a single term, P_{ki} is a component of a term pair and T_{ni} is a component of a term triple. To determine the productive terms or term combinations, it is convenient to relate the specificity of a term, or term combination, to the frequency with which the term is assigned to the documents of a collection. For single terms, in particular, one can use the postings frequencies normally available in operational retrieval systems, specifying the number of documents (postings) in a collection to which a term is assigned. The following automatic query generation method may be used as a start:

(1) consider the initial set of terms included in a given natural language query formulation, or an initially available Boolean formulation;
(2) use a stop list to eliminate common function words and words of no interest in an automatic query formulation; stop lists consisting of several hundred common function words are used in most operational retrieval environments;
(3) use the remaining single terms to generate all *anded* term pairs and term triples;*
(4) for each single term, term pair and term triple compute the expected number of "hits," that is, the document frequency or expected number of documents in the collection, which contain the given single term, term pair or term triple;
(5) formulate a query in disjunctive normal form consisting of singles, pairs, and triples in such a way that the complete query may be expected to retrieve a given specified number of documents.

It remains to choose the singles, term pairs and term triples to be incorporated in a given query and to determine the document frequency of a given term combination (that is, the expected number of documents which will be retrieved by a given term combination). A simple way of estimating the document frequency for a term combination consists in first taking the document frequency of each single term, that is, the postings frequency n_i for each term i. If one assumes that the terms occur independently of one another in the N documents of a collection, the estimated document frequency of a pair of terms (T_i *and* T_j) may be specified as

$$n_{ij} = \frac{n_i \cdot n_j}{N + 1} . \qquad (1)$$

For an *anded* triple such as (T_i *and* T_j *and* T_k), one similarly obtains

$$n_{ijk} = \frac{n_i \cdot n_j \cdot n_k}{(N + 1)^2} . \qquad (2)$$

In general, for a conjunction of p terms (T_1 *and* T_2 *and* ... *and* T_p) the estimated document frequency in a collection of N documents is set at $(n_1 \cdot n_2 \cdots n_p)/(N + 1)^{p-1}$.

Earlier work in the theory of term importance has shown that in the absence of specific information about the actual importance of query or document terms, the inverse document frequency (idf) represents a useful indi-

*A method for limiting the number of term pairs and triples actually generated is given later in this study.

cation of term importance [6,7]. This implies that the usefulness of a term, or a term combination may be determined as an inverse function of the corresponding document frequency. The idf weight of a term, or term combination, of document frequency n_i may conveniently be computed as

$$w_i = \log \frac{N+1}{n_i} \quad \text{or} \quad w_i = 1 - \frac{n_i}{N+1}. \qquad (3)$$

The document frequency computations [(1)-(3)] for terms and term combinations provide a simple way of determining the order in which the terms, or term combinations are incorporated into a given query formulation: The best terms—those with the highest inverse document frequency weight should be used first, ahead of terms with lower idf weights; analogously when eliminating terms, or term combinations, from a query formulation, those with lowest idf weights and hence highest estimated document frequency should be discarded first.

The automatic query formulation process thus consists in choosing an appropriate set of single terms, term pairs, and term triples until some preestablished retrieval threshold is reached, that is, until the number of documents that may be expected to be retrieved by the query reaches some previously established limit. The individual query components (that is, single terms, term pairs, or term triples) are chosen *without any linguistic input* or grammatical theory. The sole criterion is the estimated number of documents retrievable by a term or term combination. Since no linguistic component is involved, the automatic query formulation process is mechanizable simply and inexpensively.

As a first order of approximation, the number of documents retrieved by a query in disjunctive normal form may be taken simply as the sum of the estimated document frequencies of the individual terms or term combinations. (The simple summing of the estimated document frequencies is strictly correct only when no overlap exists among the documents retrieved by the individual terms or term combinations.) Given the set of single terms, term pairs, and term triples with their respective estimated document frequencies, the disjunctive normal form of a sample Boolean query can now be generated in the following way:

(1) Choose a few single terms in decreasing inverse document frequency weight order plus all term pairs which do not include any of the chosen singles as components. Add the document frequencies n_i for all included singles, and the estimated document frequencies n_{jk} for all included pairs to obtain the estimated number of retrieved items for this initial query formulation.

(2) If the estimated number of retrieved items is smaller than the preestablished retrieval threshold, it becomes necessary to broaden the query formulation. This is done by successively adding singles in idf weight order and subtracting the corresponding term pairs subsumed by the added singles. (For example, if term A is added to a query formulation, the pairs *A and B*, *A and C*, *A and D*, etc. become redundant.) Following each addition and subtraction operation, the estimated number of retrieved items must be recomputed.

(3) If, on the other hand, the estimated number of retrieved items is larger than the preestablished retrieval threshold, it becomes necessary to generate a narrower query formulation. This is done by successively eliminating the single terms in increasing idf weight order while adding the missing term pairs which include the deleted singles as components. If the estimated number of retrieved items is still too large, the term pairs are eventually eliminated in increasing idf weight order and the missing term triples are added. The deletion and addition operations stop when the desired number of retrieved items is eventually obtained.

A summary of the automatic Boolean query construction process appears in Table 1.

The automatic query construction method may be illustrated by using as an example a query submitted to the Medlars search system which is operated by the National Library of Medicine in Washington. The original natural language query is listed at the top of Table 2 together with the Boolean search statement generated manually by a trained Medlars searcher. The analyzed natural language statement is shown at the bottom of Table 2. Following deletion of common words and suffix removal, the original statement is reduced to seven terms. The term "pyrophosphate" could not be found in the stored automatic term dictionary and therefore does not appear in the analyzed term list. The term "effect" which does appear is deleted because of the large number of postings (248 postings in a collection of 1033 documents).* Table 2 also shows the document frequency (the number of postings) for each term found in the term dictionary.

The generation of term pairs and term triples is illustrated in Table 3 for the six terms appearing at the bottom of Table 2. To avoid repeating the full natural language representation of each term, Table 3 uses the abbreviations listed at the bottom of Table 2. The estimated document frequencies and the inverse document frequency weights are also included in Table 3 for all term pairs and triples. For m original terms there are $m(m-1)/2$ pairs and $m(m-1)(m-2)/2 \cdot 3$ triples. Therefore, 15 pairs and 20 triples are generated for the six terms of Table 2.

The actual query generation process is shown in Table 4 for the sample Medlars query of Tables 2 and 3. The assumption used in Table 4 is that the user wishes to retrieve approximately 20 documents. The first try consists

*In the current program, terms occuring in more than 20% of the documents of a collection are deleted.

TABLE 1. Summary of Boolean query construction process in disjunctive normal form.

1. Take the set of terms included in a natural language query formulation; use a stop list to eliminate unwanted function words.

2. Look up the document frequency (number of postings) for all remaining single terms, and eliminate terms with excessive document frequencies.

3. For the remaining terms generate all term pairs and term triples and compute the corresponding expected document frequencies and inverse document frequency weights.

4. Construct an initial query formulation using a few singles with the highest idf weights and term pairs which do not include the chosen singles. Compute the expected number of retrieved items as $\Sigma n_i + \Sigma n_{jk}$.

5. If the expected number of retrieved items is too small, broaden the query by adding singles in decreasing idf weight order and removing redundant pairs. Recompute the expected number of retrieved items following each query alteration.

6. If the expected number of retrieved items is too large, render the query more specific by eliminating singles and adding pairs, or eliminating pairs and adding the corresponding triples. Recompute the expected number of retrieved items and stop the deletion and addition operations when the desired number of items is obtained.

TABLE 2. Initial query analysis.

Initial Query Formulation

Query 19: Excretion of phosphate or pyrophospate in the urine or the effect of parathyroid hormone on the kidney.

Boolean Formulation by Trained Searcher

Q 19: (urine and (phosphate or pyrophospate))

Initial Analysis (dictionary look-up, stop list, suffix removal)

Concept Number	Truncated Natural Language Term	Query Frequency	Document Frequency(n_i)	Term Weight (1 - n/(N+1)
2325	effect	1	248	.7602
2636	1 excres(ex)	1	52	.9497
3524	2 hormon(ho)	1	81	.9217
4215	3 kidney(ki)	1	78	.9246
5646	4 parathyr(pa)	1	27	.9739
5856	5 phosp(ph)	1	43	.9584
8369	6 urin(ur)	1	78	.9246

("pyrophosphate" not found in dictionary; "effect" is deleted due to excessive frequency of 248/1033)

TABLE 3. Formation of term pairs and triples. (a) Term pair formation. (b) Term triple formation.

Pair Formation	Estimated Frequency $n_{ij} = \frac{n_i \cdot n_j}{N+1}$	Term Pair Weight $1 - \frac{n_{ij}}{N+1}$	Pairs in Order of "Goodness"	Estimated Frequency
2-1 ho-ex	4.1	.9961	1. 5-4 ph-pa	1.1
3-1 ki-ex	3.9	.9962	2. 4-1 pa-ex	1.4
3-2 ki-ho	6.1	.9941	3. 6-4 ur-pa	2.0
4-1 pa-ex	1.4	.9987	4. 4-3 pa-ki	2.0
4-2 pa-ho	2.1	.9980	5. 4-2 pa-ho	2.1
4-3 pa-hi	2.0	.9980	6. 5-1 ph-ex	2.2
5-1 ph-ex	2.2	.9979	7. 5-3 ph-ki	3.2
5-2 ph-ho	3.4	.9967	8. 6-5 ur-ph	3.2
5-3 ph-ki	3.2	.9969	9. 5-2 ph-ho	3.4
5-4 ph-pa	1.1	.9989	10. 3-1 ki-ex	3.9
6-1 ur-ex	3.9	.9962	11. 6-1 ur-ex	3.9
6-2 ur-ho	6.1	.9941	12. 2-1 ho-ex	4.1
6-3 ur-ki	5.9	.9943	13. 6-3 ur-ki	5.9
6-4 ur-pa	2.0	.9980	14. 6-2 ur-ho	6.1
6-5 ur-ph	3.2	.9969	15. 3-2 ki-ho	6.1
				50.6

a

Term Triple	Frequency $n_{ijk} = \frac{n_{ij} \cdot n_k}{N+1}$	Weight $1 - \frac{n_{ijk}}{N+1}$	Term Triple	Frequency $n_{ijk} = \frac{n_{ij} \cdot n_k}{N+1}$	Weight $1 - \frac{n_{ijk}}{N+1}$
1,3-2 Ex, Ki-Ho	.31	.9997	2,4-3 Ho, Pa-Ki	.16	.9998
1,4-2 Ex, Pa-Ho	.11	.9999	2,5-3 Ho, Ph-Ki	.25	.9998
1,4-3 Ex, Pa-Ki	.10	.9999	2,5-4 Ho, Ph-Pa	.09	.9999
1,5-2 Ex, Ph-Ho	.10	.9998	2,6-3 Ho, Ur-Ki	.46	.9996
1,5-3 Ex, Ph-Ki	.16	.9998	2,6-4 Ho, Ur-Pa	.16	.9998
1,5-4 Ex, Ph-Pa	.06	.9999	2,6-5 Ho, Ur-Ph	.25	.9998
1,6-2 Ex, Ur-Ho	.31	.9997	3,5-4 Ki, Ph-Pa	.08	.9999
1,6-3 Ex, Ur-Ki	.30	.9997	3,6-4 Ki, Ur-Pa	.15	.9998
1,6-4 Ex, Ur-Pa	.10	.9999	3,6-5 Ki, Ur-Ph	.24	.9998
1,6-5 Ex, Ur-Ph	.16	.9998	4,6-5 Ta, Ur-Ph	.08	.9998

b

in using the two best single terms (those with the highest idf weight) plus all pairs not including one of these two single terms (parathyroid and phosphate). The computation of the estimated number of retrieved items shows that $27 + 43 + 3.9 + 3.9 + 4.1 + 5.9 + 6.1 + 6.1 =$ 100 items might be retrieved by this initial query formulation. A narrower statement therefore is needed which is obtained by deleting the existing single term with the poorest idf weight (phosphate) and adding the pairs containing "phosphate" which are not already included in the query (Ph *and* Ex, Ph *and* Ki, Ph *and* Ur, Ph *and* Ho). The modification process changes the expected number of retrieved items by $-43 + 2.2 + 3.2 + 3.2 + 3.4 =$ -31 items. The second query statement in Table 4 would thus retrieve $100 - 31 = 69$ items.

Successive narrowing operations remove the last remaining single term (parathyroid) which is replaced by five pairs containing the Pa term; this reduces the expected number of retrieved items to 50.6. The corresponding query statement is shown in step 3 of Table 4. Since no further single terms remain, it is necessary next to remove pairs of terms in order to obtain narrower query statements. In steps 4 and 5 of Table 4 the term pairs with highest estimated document frequency (lowest idf weight) are successively removed to reduce the estimated number of retrieved items to 38.4. In step 6 one additional term pair (Ur *and* Ki) is removed, but since the three pairs (Ki *and* Ho), (Ur *and* Ho) and (Ur *and* Ki) are now absent, the triple (Ur *and* Ki *and* Ho) is no longer redundant. This triple is therefore added in step 6 following the removal of (Ur *and* Ki). Successive deletions of three more term pairs followed by the addition of nonredundant term triples finally produce a query statement consisting of the 9 pairs and 4 triples shown at the bottom of Table 4. This query may be expected to retrieve 22 items which is close enough to the stated number of wanted items. It is clear that all addition and deletion operations are controlled only by the estimated document frequencies and idf weights of the terms, and therefore can be carried out automatically without difficulty.

TABLE 4. Refinement of query formulations.

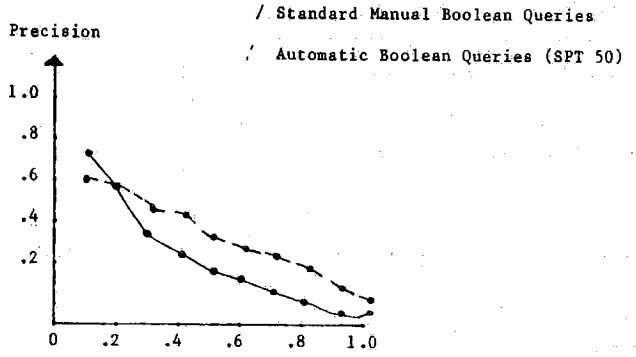
Estimated Number of Retrieved Items	Composition	Query Formulation
1. 100	6P,2S (two singles of lowest frequency and all pairs not involving the two singles)	Pa or Ph or (Ki and Ex) or (Ur and Ex) or (Ho and Ex) or (Ur and Ki) or (Ur and Ho) or (Ki and Ho)
2. 69	10P,1S (replace the highest frequency single)	Pa or [(Ph and Ex) or (Ph and Ki) or (Ph and Ur) or (Ph and Ho)] or (Ki and Ex) or (Ur and Ex) or (Ho or Ex) or (Ur and Ki) or (Ur and Ho) or (Ki and Ho)
3. 50.6	15P (replace next highest frequency single (Pa) by five pairs)	[(Pa and Ph) or (Pa and Ex) or (Pa and Ur) or (Pa and Ki) or (Pa and Ho)] or (Ph and Ex) or (Ph and Ki) or (Ph and Ur) or (Ph and Ho) or (Ki and Ex) or (Ur and Ex) or (Ho and Ex) or (Ur and Ki) or (Ur and Ho) or Ki and Ho)
4. 44.5	14P (remove highest frequency pair)	as above without (Ki and Ho) clause
5. 38.4	13P (remove next highest frequency pair)	as above without (Ur and Ho) clause
6. 33	12P,1T (remove next highest pair, add one nonredundant triple)	as above without (Ur and Ki) clause but with added triple (Ur and Ki and Ho)
7. 28.9	11P,1T (remove next highest frequency pair)	as above without (Ho and Ex) clause
8. 25.3	10P,2T (remove next highest frequency pair, add one nonredundant triple)	as above without (Ki and Ex) clause but with added triple (Ki and Ex and Ho)
9. 22	9P,4T (remove next highest pair (Ur and Ex) and add two triples (Ur and Ex and Ho) and (Ur and Ex and Ki))	(Pa and Ph) or (Pa and Ex) or (Pa and Ur) or (Pa and Ki) or (Pa and Ho) or (Ph and Ex) or (Ph and Ki) or (Ph and Ur) or (Ph and Ho) or (Ur and Ki and Ho) or (Ki and Ex and Ho) or (Ur and Ex and Ho) or (Ur and Ex and Ki)

The automatic Boolean query generation process using term singles, pairs, and triples in disjunctive normal form may be evaluated by comparing the retrieval results obtained for the automatically formulated queries with the results obtained with manually formulated queries. To evaluate the effectiveness of a retrieval system it is customary to compute values of the search recall and search precision following the retrieval of some fixed number of documents. The recall represents the proportion of relevant items retrieved out of the total number of relevant in the whole collection, and the precision represents the proportion of relevant items retrieved out of the total number retrieved. In general, the presumption is that an average user is interested in retrieving most everything 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 such as that shown in Figure 1 [4,5].

Recall Level	Average Precision for 30 Queries	
	Manual Queries Standard Boolean	Automatic Boolean Queries (S,P,T) Set to Retrieve 50 Items
0.1	0.5606	0.4731
0.2	0.4364	0.4261
0.3	0.2917	0.3990
0.4	0.2387	0.3421
0.5	0.1871	0.2890
0.6	0.1562	0.2508
0.7	0.1073	0.1867
0.8	0.0774	0.1656
0.9	0.0385	0.0873
1.0	0.0324	0.0282
Average Improvement		+40.4%

a Typical Recall-Precision Table



b Recall Precision Graph

FIG. 1 Comparison of manual Boolean queries with automatic Boolean formulations (SPT 50) constructed from natural language statements (Medlars 1033, 30 queries).

The evaluation output of Figure 1 compares the effectiveness of a set of automatically generated Boolean queries using the process of Table 1, with manually generated Boolean queries. Thirty queries are used experimentally with a collection of 1033 documents in biomedicine. The automatic method evaluated in Fig. 1 is designated as SPT 50, indicating that single terms (S), term pairs (P), and term triples (T) are used and that the number of wanted retrieved items is set to 50. For the Medlars collection of 1033 documents it may be seen that automatic queries produce better output than the manual queries at recall levels exceeding 0.2. The recall-precision curve for the automatic query formulation process is correspondingly closer to the upper right-hand corner of the graph where both recall and precision values are equal to 1. For the Medlars collection the manual query formulations were evidently not optimal, and the larger number of *anded* term pairs and triples included in the automatic query forms produced better output than the original manual queries.

It is awkward to compare complete tables or graphs with each other when many comparisons are performed. In the remainder of this study, the evaluation results are therefore 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. A complete recall-precision comparison using the composite precision values is shown in Table 5 for the Medlars 1033 collection. The base run listed at the top of Table 5 is the conventional retrieval output obtained with manually formulated Boolean queries. A secondary base run also included in Table 5 is the vector processing run using a cosine coefficient to compare query and document vectors composed of weighted terms automatically derived from natural language statements [4,5]. For the Medlars collection under consideration, the vector processing system using term weights and a large number of natural language terms proves superior to the standard Boolean query system.

The output for the automatic Boolean query generation process is included in the center section of Table 5. Various cases are shown for singles, pairs, and triples, including retrieval thresholds of 20, 30, 50, and 100 documents. Several cases using only single terms with retrieval thresholds of 30, 100, and 500 are also given in Table 5; this last case simulates a vector processing system using a large number of (unweighted) terms with a high retrieval threshold. It can be seen from Table 5 that the automatic query generation process using singles, pairs, and triples produces better results than the conventional Boolean search using manually prepared queries. When singles, pairs, and triples are used, the best results are obtained with a retrieval threshold of 50 documents. All Boolean searches, whether based on manually prepared or automatic queries, are considerably less effective than the vector processing search based on weighted term vectors.

The last result at the bottom of Table 5 corresponds to an upper-bound experiment where the unrealistic assumption is made that the relevance or nonrelevance of each document with respect to each query is known in advance before the searches are actually carried out. In that case it is possible to compute term relevance weights for each term as the ratio of the proportion of relevant documents in which the term occurs to the proportion of nonrelevant documents in which the term occurs [8,9]. Given the actual terms included in the user queries and the terms contained in the document abstracts, the use of term relevance weights represents the best possible performance obtainable in the particular collection environment. The results of Table 5 show that the optimal performance is substantially better than any of the Boolean retrieval runs, and still about 30% better than the vector processing method.

The comparisons carried out in Table 5 for a small collection of documents in biomedicine are repeated in Table 6 for a larger, more heterogeneous collection of over 12,000 items in the areas of electrical engineering and computer science, known as Inspec 12684. It may be

TABLE 5. Automatic generation of conventional Boolean queries from natural language text using singles, pairs, triples (Medlars 1033 collection).

Medlars 1033, 30 Queries

Type of Run	Average Precision at three Recall Points	Percent Difference from Boolean	Percent Difference from Cosine
Basic Boolean Retrieval with Manually Formed Queries (p=∞)	.2065	-	-62%
Vector Processing using Cosine Match and Natural language Queries	.5473	+165%	-
Automatically Formed Conventional Boolean Queries from Original Natural Language Statements Using Singles, Pairs, Triples (S,P,T) unweighted terms			
estimated retrieved 20	.2057	- 0.4%	-62.4%
estimated retrieved 30	.2540	+23.0%	-53.6%
estimated retrieved 50	.2899	+40.4%	-47.0%
estimated retrieved 100	.2735	+32.5%	-50.0%
estimated retrieved 200	.1918	- 7.1%	-64.9%
Automatically Formed Conventional Boolean Queries from Original Natural Language Using Single Terms (s) Only			
estimated retrieved 30	.2310	+11.9%	-57.8%
estimated retrieved 100	.3009	+45.7%	-45.0%
estimated retrieved 500	.1356	-34.3%	-75.2%
Optimal Case Using Retrospective Term Relevance Weights	.7074	+243%	+29%

TABLE 6. Automatic generation of conventional Boolean queries from natural language using singles, pairs, triples (Inspec 12684 collection).

Inspec 12684, 77 Queries

Type of Run	Average Precision at three Recall Points	Percent Difference from Boolean	Percent Different from Cosine
Basic Boolean Retrieval with Manually Formed Queries (p=∞)	.1159	-	-50%
Vector Processing using Cosine Match and Natural Language Queries	.2325	+101%	-
Automatically Formed Boolean Queries from Original Natural Language Statements Using Singles, Pairs, Triples (S,P,T)			
estimated number retrieved 20	.0367	-68.4%	-84.2%
estimated number retrieved 30	.0411	-64.6%	-82.3%
estimated number retrieved 50	.0503	-56.6%	-78.4%
estimated number retrieved 100	.0503	-56.6%	-78.4%
Automatically Formed Boolean Queries Using Singles (S) Only			
estimated number retrieved 30	.0547	-52.8%	-76.5%
estimated number retrieved 100	.0452	-61.0%	-80.5%
estimated number retrieved 500	.0337	-71.0%	-85.5%
Optimal Case Using Retrospective Term Relevance Weights	.2909	+151%	+25%

noted that in this case, the conventional Boolean search with manually prepared queries is closer to the optimal retrospective run than it was for the Medlars collection previously used. This indicates that the conventional Boolean queries were more carefully constructed for Inspec than for Medlars. Correspondingly, the automatically built Boolean queries using singles, pairs, and triples do not now perform as well as the conventional manual queries. Evidently, the natural language statements which were used as a basis for the automatic query construction method were not well suited to the heterogeneous Inspec collection. In that case one cannot expect to construct good Boolean statements without using a substantial amount of intellect which of course is lacking in the query construction method under discussion.

The results of Tables 5 and 6 show that when good natural language statements of user need are available which match the language of the document collection, the simple automatic query construction process outlined in Table 1 is competitive with conventional manual Boolean query formulations. Indeed for the Medlars collection, the automatic Boolean queries proved superior to the manual ones. When the natural language query formulations do not match the document terms additional procedures must be incorporated in the automatic query formulation process. This possibility is further considered in the remainder of this study.

3. The Application of Extended Boolean Logic

The simple query construction process described in the previous section is surprisingly powerful, given that the *anded* term pairs and term triples which are used as a basis are constructed by purely probabilistic, as opposed to semantic, criteria. The results obtained with the Inspec collection (Table 6) show, however, that additional improvements are desirable in the automatic query construction process.

The first possibility consists in using a frequency threshold to control the number of single terms, term pairs, and term triples which enter into the query construction process. Since m single terms will produce of the order of m^2 term pairs and m^3 term triples, the efficiency of the basic query generation method may be improved by using only certain term combinations instead of all possible pairs and triples. It is known that the most appropriate entities for use as content identifiers in a collection of documents exhibit a document frequency which is neither too high or too low [10]. This suggests that high-frequency single terms which occur in a substantial number of documents can safely be eliminated because such terms cannot be used to discriminate the documents from each other. An analogous argument applies to very low-frequency term pairs and term triples because these entities are insufficiently frequent to be of great interest. A high-frequency threshold for the single terms of $N/5$, deleting all terms occurring in more than one fifth of the collection items may be acceptable. For term pairs and triples a low frequency threshold of 0.1 or 0.2 may also be useful. This removes a potentially large number of term phrases with low frequency components whose expected overall document frequency is very small.

The use of frequency thresholds in the generation of term phrases will speed up the automatic query generation process. However, such a threshold does nothing to control the quality of the term phrases that are actually used in an automatic query. The Boolean query construction system of Table 1 may in fact be somewhat inconsistent in the sense that certain semantically questionable term combinations may be created which are later interpreted by using the strict rules of Boolean logic. Thus when an *and* connective is used to relate two phrase components, the respective terms are treated as essential regardless of semantic appropriateness. Similar considerations apply to terms related by *or* connectives.

Since the formal query construction process creates some questionable phrases, improved retrieval results may be obtainable by using a fuzzy type of query document matching in which the interpretation of the Boolean connectives is less unforgiving than in the conventional Boolean logic. Such as extended Boolean system has recently been proposed, using an auxiliary parameter p, $1 \leq p \leq \infty$, attached to the connectives *and* and *or* to control the query-document matching process, and providing thereby a variety of more or less stringent interpretations for the query statements [11,12].

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, $0 \leq d_A, d_B \leq 1$. A term weight of 0 indicates that the corresponding term is not assigned to an item; a weight of 1 represent a fully weighted term, and weights between 0 and 1 are partial term assignments. Given queries (A *and* B) and (A *or* B), it is possible to define the following query-document similarity functions between these queries and a document $D = (d_A, d_B)$.

$$\text{sim}(Q_{(A \text{ and } B)}, D) = 1 - \left[\frac{(1 - d_A)^p + (1 - d_B)^p}{2} \right]^{1/p} \tag{4a}$$

$$\text{sim}(Q_{(A \text{ or } B)}, D) = \left[\frac{d_A^p + d_B^p}{2} \right]^{1/p}. \tag{4b}$$

It is easy to verify that when p is large, the similarity function (4a) 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 is equal to 0. In the same way, the function (4b) 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

TABLE 7. Query-document similarity for unweighted queries $D = (d_A, d_B)$; $Q = (A \text{ and } B)$, $Q = (A \text{ or } B)$.

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

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

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

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

exactly the results of the conventional Boolean logic when large p values are used. The corresponding example is included at the bottom of Table 7.

Consider now the case when p is equal to 1. In that case values of 1 are produced by both expressions (4a) and (4b) when $d_A = d_B = 1$; when $d_A = d_B = 0$ both (4a) and (4b) 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 (4). 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 $Q = (q_1/k, q_2/k, \ldots, q_k/k)$ is evaluated as the usual vector product $\sum_{i=1}^{k} (d_i \cdot q_i/k)$. 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 7.

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 7 shows the case where $p = 2$ and the weights are binary. 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 [11,12],

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

$$\text{sim}(Q_{[(A,a)and(B,b)]}, D) = 1 - \left[\frac{a^p(1-d_A)^p + b^p(1-d_B)^p}{a^p + b^p}\right]^{1/p} \quad (5a)$$

and

$$\text{sim}(Q_{[(A,a)or(B,b)]}, D) = \left[\frac{a^p d_A^p + b^p d_B^p}{a^p + b^p}\right]^{1/p} \quad (5b)$$

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 interpretation of the p-value variations is summarized in the graph of Figure 2.

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.

A typical set of query formulations usable in the extended system is shown in Table 8. A standard Boolean query is shown first in part (a) of Table 8. The corresponding formulation in the extended Boolean system is shown in Table 8(b). A p-value of $p = \infty$ is now attached to each Boolean operator, and each query term receives a value of 1. Weights may also be attached to the query clauses (that is, to each *and*-clause and *or*-clause included in a query). In the illustration of Table 8, the clause weights are underlined and chosen as the averages of the weights of the terms in the respective clauses. For the normal Boolean formulation of Table 8(b), the clause weights are of course also equal to 1. When the matching functions of expressions 4 and 5 are used with the query formulation of Table 8(b), the results are identical with those obtained by a conventional Boolean retrieval system using the formulation of Table 8(a).

The relaxed p-norm formulation for the sample query is shown in Table 8(c). The outer *and* and inner *or* operators are assigned p-values designated as p_1 and p_2, respectively. Each term is weighted using the automatically determined inverse document frequency (idf) value computed as $\log N/n_i$ [see expression (3)]. The clause weight is set as the average of the idf values of the component terms. The last part of Table 9 contains a standard vector formulation for the same query, consisting of a set of weighted terms used without Boolean operators.

The operations of the extended Boolean retrieval system are evaluated in the next section.

4. Automatic Query Formulation in the Extended System Using Single Terms, Term Pairs, and Term Triples

The document collections already used earlier to evaluate the automatic query formulations in the conventional Boolean system are used again to carry out experiments in the extended model. The experimental results cover retrieval thresholds which vary between 20 and 500 documents, respectively; p-values equal to 1, 2, 5, and in-

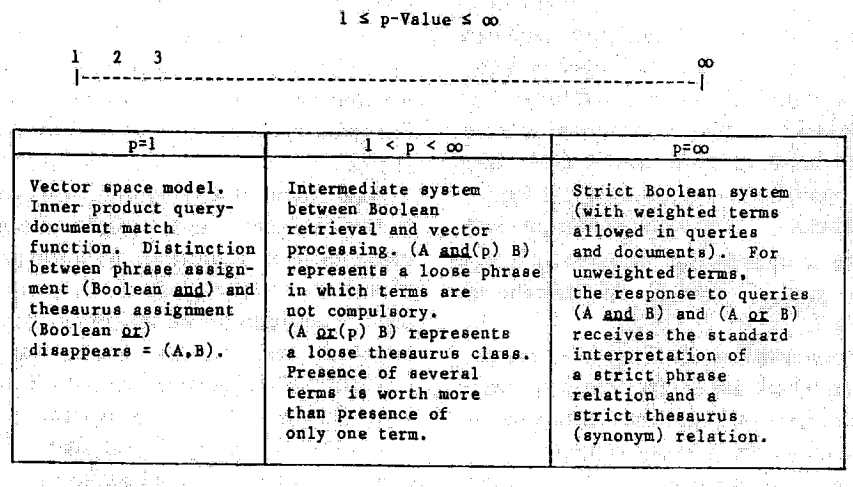


FIG. 2. Interpretation of p-value variations ($1 \leq p \leq \infty$).

TABLE 8. Typical query formulations in extended Boolean retrieval (term weights: log N/n; clause weights: average term weight).

a Unweighted query formulation (Boolean statement)

```
(catalogue or catalog)
   and (mechanization or automation or computerization)
```

b Weighted query formulation (binary weights)

```
and(∞) (< or(∞) (<catalogue,1>,<catalog,1>, 1 >
    or(∞) (<mechanization,1>, <automation,1>,
          <computerization,1>, 1 >)
```

c Weighted query formulation (term wts:idf; clause wts:average idf)

```
and(p1) (<or(p2)(<catalogue,6.358>,<catalog,5.358>), 5.898 >
     <or(p2) (<mechanization,4.036>, <automation,2.657>,
          <computerization,1.603>), 2.765 >)
```

d Vector formulation

```
(<catalogue,6.358>, <catalog,5.358>, <mechanization,4.036>,
     <automation,2.657>, <computerization,1.603>)
```

finity are used in the experiments. A common p-value was assigned to all Boolean query operators within a given experimental run.

Since the extended system makes it possible to assign term weights, the query terms are weighted using the previously mentioned idf values of expression (3). It is known that for the document terms, a good term weighting system favors terms with a large number of occurrences in individual document texts, but a small number of occurrences over the whole collection [10]. A term receiving a large value in such a system will obviously help in distinguishing the document in which it occurs from the remainder of the collection. In the experiments which follow, a combined document term weight is therefore used, defined as the product (tf $\times$ idf) of the frequency of the term, tf, in an individual document multiplied by the inverse document frequency idf of the term in the collection [10,12].

The experimental output for the automatically constructed p-norm queries is shown in Tables 9 and 10 for the Medlars and Inspec collections, respectively. The runs used as a basis for comparison are the same ones used earlier, corresponding to the conventional manually constructed Boolean queries and the automatically generated vector queries using natural language input. The results of Tables 9 and 10 must be compared with the previous output of Tables 5 and 6 for the automatic queries in the conventional Boolean system.

It is obvious that the retrieval effectiveness of the automatic queries in the extended Boolean system is far greater than the effectiveness of these queries in the conventional Boolean system previously illustrated in Tables 5 and 6. For both collections the automatic queries also produce results that are substantially superior to those obtained with the manually constructed Boolean queries. The best results are obtained for low p-values between 1 and 5 where the interpretation of the Boolean connectives is most relaxed. Conversely, the worst values are those for $p = \infty$ where the connectives are interpreted in the strict conventional way.

For the Medlars collection the automatic p-norm queries are comparable in effectiveness also with the automatically constructed vector queries used in a vector processing system. The advantage of the p-norm queries compared with the vector queries is the compatibility with the conventional Boolean processing system which is lacking in the vector processing environment. For the Inspec collection, the automatic query formulations produce results which are 20 to 30% better than the manual Boolean queries, but 30 to 40% worse than the vector pro-

TABLE 9. Automatic generation of p-norm queries from natural language statements using singles, pairs, and triples (Medlars 1033 collection).

Type of Run		Medlars 1033 30 Queries: Average Precision at three Recall Points	Percent Difference from Boolean	Percent Difference from Cosine
Basic Boolean Retrieval with Manually Formed Queries (p=∞)		.2065	-	-62%
Vector Processing using Cosine Match and Natural Language Queries (weighted terms)		.5473	+165%	-
Automatically Formed P-Norm Queries from Original Natural Language Statements Using Singles, Pairs, Triples (weighted document and query terms)				
estimated retrieved 20	p=1	.5588	+170.7%	+ 2.1%
	p=2	.5447	+163.8%	- 0.5%
	p=5	.5179	+150.8%	- 5.4%
	p=∞	.4039	+ 95.6%	-26.2%
estimated retrieved 50	p=1	.5367	+159.9%	- 1.9%
	p=2	.5284	+155.9%	- 3.5%
	p=5	.5062	+145.2%	- 7.5%
	p=∞	.4595	+122.6%	-16.0%
estimated retrieved 100	p=1	.5597	+171.1%	+ 2.3%
	p=2	.5440	+163.5%	- 0.6%
	p=5	.5028	+143.5%	- 8.1%
	p=∞	.4594	+122.5%	-16.1%
estimated retreived 200	p=1	.5331	+158.3%	- 2.6%
	p=2	.5366	+159.9%	- 2.0%
	p=5	.4911	+137.9%	-10.3%
	p=∞	.4500	+118.0%	-17.8%
Automatically Formed P-Norm Queries from Original Natural Language Statements Using Single Terms Only (weighted document and query terms)				
estimated retrieved 30	p=1	.2542	+ 23.1%	-53.6%
	p=2	.2544	+ 23.2%	-53.5%
	p=5	.2552	+ 23.6%	-53.4%
	p=∞	.2553	+ 23.7%	-53.3%
estimated retrieved 100	p=1	.4706	+127.9%	-14.0%
	p=2	.4675	+126.4%	-14.6%
	p=5	.4530	+119.4%	-17.2%
	p=∞	.4394	+112.8%	19.7%
estimated retrieved 500	p=1	.5550	+168.8%	+ 1.4%
	p=2	.5397	+161.4%	- 1.4%
	p=5	.4943	+139.4%	- 9.7%
	p=∞	.4594	+122.5%	-16.1%
Optimal Case Using Retrospective Term Relevance Weights		.7074	+243%	+29%

cessing queries. The automatic queries in the extended system appear good enough to be used as initial queries to be improved by user-system interaction for subsequent searches.

The experiments of Tables 9 and 10 lead to a practical method for the construction of effective quasi-Boolean queries:

(1) conventional Boolean queries are first constructed in disjunctive normal form, using single terms, as well as *anded* term pairs and term triples;
(2) p-values are assigned to the Boolean connectives; if no special information is available about the importance of particular terms, uniformly low values of $p = 1$ or $p = 2$ are preferred;
(3) the query terms are weighted using actual inverse document frequency values for single terms and estimated idf values for term pairs and triples; the clause weights are determined as the average idf values of the component terms;
(4) the matching functions of expression (5) are used to compare queries and documents, and the documents are brought to the users' attention in decreasing order of the query-document similarity.

Judging by the results obtained in laboratory experiments with several document collections, the automatic query construction process based on single terms, term pairs, and term triples will produce retrieval results which are often superior to and in any case not substantially

TABLE 10. Automatic generation of p-norm queries from natural language statements using singles, pairs, and triples (Inspec 12684 collection).

Type of Run		Average Precision at three Recall Points	Percent Difference from Boolean	Percent Difference from Cosine
Inspec 12684 77 Queries				
Basic Boolean Retrieval with Manually Formed Queries (p=∞)		.1159	-	-50%
Vector Processing using Cosine Match and Natural Language Queries (weighted terms)		.2325	+101%	-
Automatically Formed P-Norm Queries from Original Natural Language Statements Using Singles, Pairs, Triples (weighted document and query terms)				
estimated retrieved 20	p=1	.1467	+26.6%	-36.9%
	p=2	.1467	+26.6%	-36.9%
	p=5	.1503	+29.6%	-35.3%
	p=∞	.1229	+ 6.0%	-47.1%
estimated retrieved 30	p=1	.1499	+29.3%	-35.5%
	p=2	.1550	+33.7%	-33.3%
	p=5	.1518	+30.9%	-34.7%
	p=∞	.1168	+ 0.7%	-49.8%
estimated retrieved 50	p=1	.1356	+17.0%	-41.7%
	p=2	.1493	+28.8%	-35.8%
	p=5	.1567	+35.1%	-32.6%
	p=∞	.1266	+ 9.2%	-45.5%
estimated retrieved 100	p=1	.1375	+18.6%	-40.9%
	p=2	.1378	+18.8%	-40.7%
	p=5	.1425	+22.9%	-38.7%
	p=∞	.1175	+ 1.4%	-49.4%
Automatically Formed P-Norm Queries from Original Natural Language Statements Using Single Terms Only (weighted document and query terms)				
estimated retrieved 30	p=1	.0724	-37.6%	-68.9%
	p=2	.0695	-40.1%	-70.1%
	p=5	.0695	-40.1%	-70.1%
	p=∞	.0695	-40.1%	-70.1%
estimated retrieved 100	p=1	.0818	-29.4%	-64.8%
	p=2	.0770	-33.6%	-66.9%
	p=5	.0770	-33.6%	-66.9%
	p=∞	.0758	-34.6%	-67.4%
estimated retrieved 500	p=1	.1110	- 4.2%	-52.2%
	p=2	.0993	-14.4%	-57.3%
	p=5	.0974	-16.0%	-58.1%
	p=∞	.0937	-19.1%	-59.7%
Optimal Case Using Retrospective Term Relevance Weights		.2909	151%	+25%

worse than those obtained with conventional Boolean queries formulated by trained search intermediaries.

5. Automatic Query Formulation in the Extended System Using Frequency Range Methods

In the automatic query construction system described up to now, no clear prescription is given about the actual values of the p-parameters to be attached to the various Boolean operators. In a more refined query construction process, a distinction could, however, be made among connectives by assigning higher p-values when the term relationships appear important enough to be interpreted strictly, while using lower p-values when the interpretation of the operators can be relaxed. The insights provided by the term discrimination theory can be used to make the necessary distinctions [13].

The discrimination value of a term measures the ability of the term to distinguish a document to which the term is assigned from the remaining documents of a collection. It is obvious that very high frequency terms are very poor discriminators since they are assigned to many documents in a collection which then become indistinguishable. It is not so obvious that very low frequency terms are also poor discriminators because their very rarity implies that their presence makes little difference on a collection-wide basis. The graph of Figure 3 provides a summary of

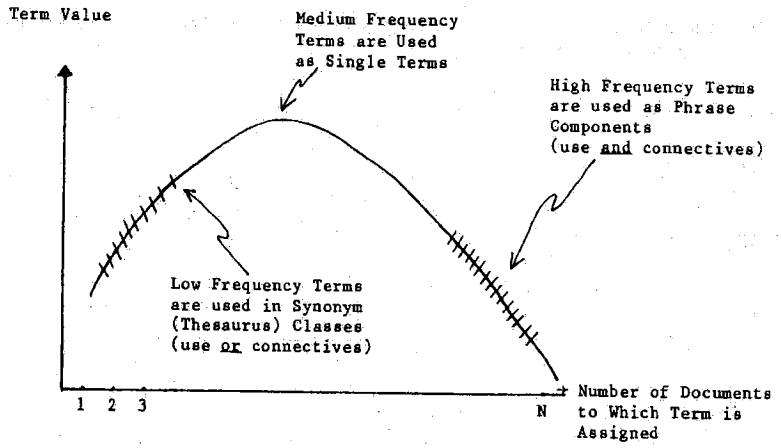


FIG. 3. Term discrimination characterization.

the variations of the term discrimination value with the document frequency of a term [13]

(1) very low document frequency terms exhibit low discrimination values;
(2) as the document frequency of a term rises, the discrimination value of the term improves up to a maximum point reached for medium frequency terms;
(3) as the document frequency increases still further, the discrimination value of a term worsens rapidly;
(4) very high frequency terms exhibit the worst discrimination values.

The term discrimination analysis shows that very low and very high frequency terms do not produce satisfactory content identifiers. This suggests that frequency transformation operators be used to lower the frequency of terms on the high side of frequency spectrum, while increasing the frequency of the terms on the low side of the spectrum. A broad high-frequency term receives a narrower scope and hence a lower assignment frequency when it is incorporated into a *term phrase*; for example, instead of using "computer" one could use "computer science," "computer theory," or "computer program." Analogously, a narrow low frequency term receives a broader interpretation by using it together with other low-frequency synonyms; instead of using the single identifier "microcomputer," one would say "microcomputer *or* minicomputer *or* hand-held calculator."

The term discrimination theory can be used as a basis for a new query formulation system which is compatible with the earlier system based on single terms, term pairs, and term triples, but also provides special treatment for the low frequency terms in addition to the high frequency terms:

(1) low frequency terms should be broadened by incorporating several related low-frequency terms into a single *or*-clause, that is, a clause related by *or* connectives;
(2) medium frequency terms should be used as single terms without further transformation;
(3) high frequency terms should be narrowed by forming *and*-clauses which include several high-frequency terms related by *and* components.

In the *frequency-range* query generation method the terms are grouped into classes according to their respective document frequencies, and within the various groups *or*-operators are used to connect the low frequency terms, while *and*-operators are used for the high-frequency terms. In addition, an *outer operator*—either *and* or *or*—is used to connect the various frequency classes with each other. Consider as an example a typical frequency range query using five different ranges of the frequency. The following formulation can be used:

$$[A_1 \text{ or } (p_1)\, A_2 \text{ or } (p_1) \ldots \text{ or } (p_1)\, A_m] \ \Big\}$$
very low frequency terms
$$\text{or } (p_0)\, [B_1 \text{ or } (p_2)\, B_2 \text{ or } (p_2) \ldots \text{ or } (p_2)\, B_n] \ \Big\}$$
medium low frequency terms
$$\text{or } (p_0)\, C_1 \text{ or } (p_0)\, C_2 \ldots \text{ or } (p_0)\, C_p \ \Big\}$$
medium frequency terms
$$\text{or } (p_0)\, [D_1 \text{ and } (p_3)\, D_2 \text{ and } (p_3) \ldots \text{ and } (p_3)\, D_q] \ \Big\}$$
medium high frequency terms
$$\text{or } (p_0)\, [E_1 \text{ and } (p_4)\, E_2 \text{ and } (p_4) \ldots \text{ and } (p_4)\, E_r)] \ \Big\}$$
very high frequency terms

A greater or smaller number of frequency ranges could be utilized in any given instance and the outer operator [*or* (p_0)] can be replace by *and* (p_0).

It remains to specify a procedure for choosing an appropriate number of frequency ranges and for assigning the p-values to the outer and inner Boolean operators. Since the terms falling into the more extreme frequency ranges (either very low or very high frequency groups) are most in need of being related to other terms, the corresponding p-values should be interpreted more strictly than the p-values for the less extreme frequency ranges. For the example previously used a value of 2 might be used

TABLE 11. Frequency range determination.

Frequency Range	Number of Terms	Percentage of Terms (8 classes)	Percentage of Terms (5 classes)	Percentage of Terms (3 classes)	Percentage of Terms (2 classes)
1	4681	53.5%	53.5%	53.5%	53.5%
2	1099	12.6%	12.6%	26.2%	46.5%
3	577	6.6%			
4-5	613	7.0%	13.6%		
6-8	492	5.6%	10.5%		
9-14	429	4.9%		20.3%	
15-29	434	5.0%	9.8%		
30-357	425	4.8%			

a Frequency Range Subdivision for Medlars 1033 Collection

Frequency Range	Number of Terms	Percentage of Terms (8 classes)	Percentage of Terms (5 classes)	Percentage of Terms (3 classes)
2	5209	35.5%	35.5%	35.5%
3	2103	14.3%	14.3%	31.0%
4	1156	7.8%	16.7%	
5-6	1306	8.9%		
7-10	1232	8.4%	16.9%	
11-20	1245	8.5%		
21-58	1199	8.2%	16.6%	33.5%
59+	1233	8.4%		

b Frequency Range Subdivision for Inspec 12684 Collection

TABLE 12. Automatic query formulation using frequency range method.

a) Automatic Query Formulation Rules

1. very low frequency term : idf > 5 => or(2)
2. medium low frequency term : 3 < idf ≤ 5 => or(1.5)
3. very high frequency term : idf < 1.5 => and(2)
4. medium high frequency term : 1.5 ≤ idf < 3 => and(1.5)

b) Basic Automatic Query

and (1.5) [or(2) (<very low freq. terms>, clause wt)
or(1.5) (<medium low freq. terms>, clause wt)
and(1.5) (<medium high freq. terms>, clause wt)
and(2) (<very high freq. terms>, clause wt)]

Manual Form (Boolean Statement)	Automatic Form
1. (catalogue or catalog) and(mechanization or automatic or computerization)	[(<catalogue,6.36> or(2) <catalog,5.36>), 5.86] and(1.5) [<mechanization,4.04>] and(1.5) [(<automation,2.66> and(1.5) <computerization,1.60>), 2.13].
2. (information and science) and (education or training)	[<information,0.90>] and(1.5) [<science,2.19>] and(1.5) [(<education,4.36> or(1.5) <training,3.78>), 4.06]

for p_1 and p_4, covering the very low and very high frequency terms; correspondingly smaller values of 1.5 or 1.2 should be appropriate for p_2 and p_3. Finally, a very low value of 1 or 1.1 might be suitable for the outer operator p_0. When the outer operator p_0 equals 1, the results are of course identical for *and* and *or* operators.

The frequency ranges can be chosen automatically in such a way that the number of terms in each range is approximately the same. Consider as an example the distribution of document frequencies of Table 11 for the terms used with the Medlars and Inspec collections. (To reduce the term set used for indexing to a manageable size, terms of document frequency 1 have been excluded from consideration for the Inspec collection.) Over 53% of the terms occur with frequency 1 in the Medlars collection; another 12% occur in two documents, and about 7% in three documents. It is not possible to break down the large number of very low frequency terms into several frequency classes. However, the higher frequency terms can be grouped in such a way that each class covers approximately 5% of the terms for Medlars and 8% of the terms for Inspec as shown in column 2 of Table 2. It is obvious that the grouping operation can be performed automatically, given approximate guidelines about group size.

The arrangement in column 2 of Table 12 produces a subdivision into 8 frequency classes. By collecting terms from adjacent frequency classes, additional arrangements into 5 frequency classes or 3 frequency classes can be constructed as shown in columns 3 and 4 of Table 2. The total class frequency of the five-class arrangement varies from 10 to 14% for Medlars and from 14 to 16% for Inspec except for the very low frequency classes, while the three class arrangement covers from 20 to 35% of the collection. Other arrangements into frequency classes are also possible; experimentation will indicate which arrangements are most useful in particular circumstances.

Table 12 contains two sample queries with formulations in the conventional Boolean form and in extended Boolean form using the frequency range method. An arrangement into 4 frequency classes is used in the ex-

TABLE 13. Automatic generation of p-norm queries from natural language statements using frequency range method (outer p-operator equal to 1) (Medlars 1033, 30 queries).

Medlars 1033
30 Queries

Type of Run	Average Precision	Percent Difference from Boolean	Percent Difference from Cosine
Basic Boolean Retrieval with Manually Formed	.2065	-	-62%
Vector Processing Using Cosine Match and Natural Language Queries	.5473	+165%	-
8 ranges starting at frequency of 1,2,3,4,6,9,15,30-357 p=2,1.5,1.3,1,1,1.3,1.5,2; weighted	.5042	+144.2	- 7.9
8 ranges all p=1 weighted	.5022	+143.3	- 8.2
8 ranges all p=∞ unweighted	.0371	- 82.0	-93.2
8 ranges all p=∞ weighted	.0872	- 57.7	-84.1
5 ranges starting at 1,2,3,6,15-357 p=1.5,1.3,1,1.3,1.5; weighted	.4953	+139.9	- 9.5
5 ranges all p=1 weighted	.5005	+142.4	- 8.6
5 ranges all p=∞ unweighted	.0371	- 82.0	-93.2
5 ranges all p=∞ weighted	.1016	- 50.8	-81.4
3 ranges starting at 1,2,5-357 p=1.5,1,1.5; weighted	.5229	+153.3	- 4.5
3 ranges all p=1 weighted	.5329	+158.1	- 2.6
3 ranges all p=∞ unweighted	.0371	- 82.0	-93.2
3 ranges all p=∞ weighted	.1027	- 50.3	-81.2
Optimal Retrospective Term Relevance Weights	.7074	+243	+20

amples of Table 12, with p-values equal to 2 for the very low and very high-frequency terms, and p-values of 1.5 for medium-high and medium-low frequency terms. An *and* connective is used as outer operator in the example of Table 12 with a p-value of 1.5.

Tables 13 and 14 contain evaluation results for the operations of automatically constructed frequency-range queries for the Medlars and Inspec collections, respectively. Once again the automatic frequency range queries are compared with the manually constructed Boolean queries. It is clear that the automatic queries derived from natural language statements are again substantially superior to the manual queries except for the cases when $p = \infty$, entailing a strict interpretation of the Boolean connectives. The best results are obtained when the frequency range is subdivided into a small number of subsets (3 for Medlars and 5 for Inspec). Approximately equal results are obtained when all p-values are set to 1, and when the p-values are varied. A comparison between the results of Tables 13 and 14 covering the frequency range method and those Tables 9 and 10 for the method using singles, pairs, and triples, respectively, shows that the frequency range process produces results comparable to the best output obtained with the earlier SPT procedure. All the procedures for automatic query formulation produce results which substantially outperform the manually constructed conventional Boolean queries.

6. Utilization of Automatic Query Formulation Process

A few words are in order about the methods that could be used to incorporate the foregoing automatic query construction methods in a practical retrieval environment based on Boolean queries and inverted file methodologies. The following sample process requires no alterations of the basic operational bibliographic retrieval system except for the addition of "back-end" programs to carry out the query-document comparisons in the extended p-norm model using the formulas of expressions (4) and (5):

(1) Given a user search statement in natural language form, a broad conventional Boolean query formulation is prepared similar to the first statement in the example of Table 4. Typically, such a statement consists of single terms and possibly a few *anded* phrases all connected by *or*-operators, chosen without any linguistic input in accordance with the probabilistic considerations previously outlined. The initial Boolean statement must be sufficiently broad to retrieve most potentially relevant items.*

*Methods for constructing broad query statements, or for "loosening" narrow Boolean statements have been described in the literature [14].

TABLE 14. Automatic generation of p-norm queries from natural language statements using frequency range method (outer p-operator equal to 1) Inspec 12684, 77 queries.

Inspec 12684
77 Queries

Type of Run	Average Precision at three Recall Points	Percent Difference from Boolean	Percent Difference from Cosine
Basic Boolean Retrieval with Manually Formed Queries (p=∞)	.1159	-	-50%
Vector Processing Using Cosine Match and Natural Language Queries	.2325	+101%	-
Automatic p-Norm Queries Using Frequency Range Method and Weighted Document Terms (outer p-value = 1)			
8 ranges: 2,3,4,5,7,11,21,59 all p=1	.1519	+31%	-34.7%
8 ranges: 2,3,4,5,7,11,21,59 p=2,1.5,1.3,1,1,1.3,1.5,2	.1487	+28.2%	-36%
5 ranges: 2,3,4,7,21 all p=1	.1592	+37.3%	-31.5%
5 ranges: 2,3,4,7,21 p=1.5,1.3,1,1.3,1.5	.1588	+37.0%	-31.7%
Optimal Case Using Retrospective Term Relevance Weights	.2909	+151%	+25%

(2) The initially constructed broad Boolean query is used to retrieve a subcollection of potentially relevant documents by searching the available collections using the conventional inverted file technology.
(3) The initial natural language query formulation is then used as input to the automatic query generation process to produce an automatic formulation in the extended Boolean system using either the process of Table 1 based on singles, pairs, and triples or the frequency range method.
(4) The automatically constructed query is compared with the documents included in the previously retrieved subcollection (see step 2). For each item in this subcollection, a query-document similarity function is produced by using expressions (4) and (5), and the documents are ranked and retrieved in decreasing order of the query-document similarity.

The processes described earlier to construct the automatic quasi-Boolean queries can of course be used also to improve available query statements based on the texts, or text excerpts, of documents retrieved in step 4 which are identified as relevant by the users. Such an automatic *query reformulation* process using *relevance feedback* methods can be carried out by methods similar to those already described for the basic query formulations [15]. However, a detailed consideration of such a relevance feedback process is beyond the scope of this study.

The automatic Boolean query construction methods described in this article are easily included in conventional operational bibliographic retrieval environments, and the costs of the required backend programs should be low. Their utilization may render unnecessary any manual construction of Boolean query formulations and may substantially improve the quality of the retrieval service.

References

1. Barker, F. H.; Veal, D. C.; Wyatt, B. K. "Towards automatic profile construction." *Journal of Documentation*. 28(1):44-55; 1972.
2. Preschel, B. M. "Indexer consistency in perception of concepts and choice of terminology." Final Report, School of Library Science, Columbia University, New York, June 1972.
3. Lufkin, R. C. "Determination and analysis of some parameters affecting the subject indexing process." Master's thesis, Department of Electrical Engineering, M.I.T., Cambridge, MA, September 1968.
4. Salton, G. *Automatic Information Organization and Retrieval.*, McGraw-Hill; 1968.
5. Salton, G., Ed. *The Smart Retrieval System-Experiments in Automatic Document Processing*. Englewood Cliffs, NJ: Prentice-Hall; 1971.
6. Sparck Jones, K. "A statistical interpretation of term specificity and its application in retrieval." *Journal of Documentation*. 28(1):11-21; 1972.
7. Salton, G.; Wu, H.; Yu, C. T. "The Measurement of Term Importance in Automatic Indexing." *Journal of the American Society for Information Science*. 32(3):175-186; 1981.
8. Robertson, S. E.; Sparck Jones, K. "Relevance Weighting of Search Terms." *Journal of the American Society for Information Science*. 27(3):129-146; 1976.
9. Sparck Jones, K. "Experiments in relevance weighting of search terms." *Information Processing and Management*. 15:133-144; 1979.
10. Salton, G. "A theory of indexing." *Regional Conference Series in Applied Mathematics No. 18*. Society for Industrial and Applied Mathematics, Philadelphia, PA, February 1975.
11. Wu, H. "On query formulation in information retrieval." Doctoral disseration, Cornell University, Ithaca, NY, January 1981.
12. Salton, G.; Fox, E. A.; Wu, H. "Extended Boolean information retrieval." Technical Report 82-511, Department of Computer Science, Cornell University, Ithaca, NY, 1982.
13. Salton, G.; Yang, C. S.; Yu, C. T. "A Theory of Term Importance in Automatic Text Analysis." *Journal of the American Society for Information Science*. 26(1):33-44; 1975.
14. Vernimb, C. "Automatic query adjustment in document retrieval." *Information Processing and Management*. 13(6):339-353; 1977.
15. Salton, G.; Fox, E. A.; Buckley, C.; Voorhees, E. "Boolean query formulation with relevance feedback." Technical Report 83-539, Dept. of Computer Science, Cornell University, Ithaca, NY, 1983.