RESEARCH CONTRIBUTIONS

Martha Williams
Guest Editor

Extended Boolean Information Retrieval

GERARD SALTON *Cornell University*

EDWARD A. FOX *International Institute for Tropical Agriculture, Ibadan, Nigeria*

HARRY WU *ITT Programming Technology Center*

Gerard Salton has worked on a number of non-numeric computer applications including automatic information retrieval. He has published a large number of articles and several books on information retrieval, most recently, "Introduction to Modern Information Retrieval," 1983. Edward Fox is currently assistant professor of computer science at VPI. Harry Wu is currently involved in the comparative study of relational database systems. His main interest is in information storage and retrieval.

Authors' Present Addresses: G. Salton, Dept. of Computer Science, Cornell Univ., Ithaca, NY 14853; E. A. Fox, Virginia Polytechnic Institute and State Univ., Blacksburg, VA 24061; H. Wu, ITT—Programming Technology Center, 1000 Oronogue Lane, Stratford, CT 06497.

This study was supported in part by the National Science Foundation under Grant IST-8108696.

0001-0782/83/1100-1022 75¢

ABSTRACT: A new, extended Boolean information-retrieval system is introduced that is intermediate between the Boolean system of query processing and the vector-processing model. The query structure inherent in the Boolean system is preserved, while at the same time weighted terms may be incorporated into both queries and stored documents; the retrieved output can also be ranked in strict similarity order with the user queries. A conventional retrieval system can be modified to make use of the extended system. Laboratory tests indicate that the extended system produces better retrieval output than either the Boolean or the vector-processing system.

1. CONVENTIONAL RETRIEVAL STRATEGIES

In conventional information retrieval, the stored records are normally identified by sets of key words or index terms, and requests for information are expressed by using Boolean combinations of index terms. The retrieval strategy is normally based on an auxiliary inverted-term index that lists the corresponding set of document references for each allowable index term. The Boolean retrieval system is designed to retrieve all stored records exhibiting the precise combination of key words included in the query: when two query terms are related by an **and** connective, both terms must be present in order to retrieve a particular stored record; when an **or** connective is used, at least one of the query terms must be present to retrieve a particular item. In some systems where the natural language text of the documents or the document excerpts is stored, the user queries may be formulated as combinations of text words. In that case, the queries may include location restrictions for the query terms—for example, a requirement that the query terms occur in the same sentence of any retrieved document or within some specified number of words of each other.

Boolean retrieval systems have become popular in operational situations because high standards of performance are achievable. Furthermore, the retrieval technology which is based on list intersections and list unions to implement Boolean conjunction ("*A* **and** *B*") and Boolean disjunction ("*A* **or** *B*"), respectively, is now well understood [1,2].

The conventional Boolean retrieval technology is however also saddled with various disadvantages:

1. The size of the output obtained in response to a given query is difficult to control; depending on the assignment frequency of the query terms and the actual term combinations used in a query formulation, a great deal of output can be obtained or, alternatively, no output might be retrieved at all.
2. The output obtained in response to a query is not ranked in any order of presumed importance to the user; thus, each retrieved item is assumed to be as important as any other retrieved item.
3. No provisions are made for assigning importance factors or weights to the terms attached either to the documents or

to the queries; thus, all terms included in the documents and queries are assumed to carry equal importance.

4. Boolean query formulations may produce counterintuitive results: for example, in response to an **or** query ("*A* **or** *B* **or** . . . **or** *Z*"), an item containing only one query term is deemed just as important as an item containing all query terms; similarly, given an **and** query ("*A* **and** *B* **and** . . . **and** *Z*"), an item containing all but one of the query terms is assumed to be just as useless as an item containing none of the query terms.

Some of the disadvantages of the conventional Boolean retrieval system are eliminated in the *vector-processing retrieval model* [3, 4]. In that case, both the stored records and the information requests are unstructured and expressed simply by sets of key words of varying lengths. In the vector-processing system, both query and document terms can be weighted, 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 suffers from one major disadvantage: the structure inherent in the standard Boolean query formulation is absent. This implies that it is no longer possible to incorporate phrase-like constructs or sets of synonymous terms by using **and** and **or** connectives, respectively. (In a Boolean query formulation, an **and** connective may be used to identify query phrases as in "information **and** retrieval"; similarly, **or** connectives may relate synonymous terms as in "hand-held calculators **or** pocket computers **or** microcomputers.")

Various intermediate retrieval systems have been designed that include features of both the Boolean and the vector-processing models. For example, in the SIRE system (Syracuse Information Retrieval Experiment), a standard Boolean query-processing system may be used first to retrieve items that respond precisely to the Boolean query formulations. An output ranking is then obtained by using the weights attached to the document terms to display the retrieved items in decreasing order according to the sum of the weights of the matching terms in queries and documents [5].

Document weights and output ranking are also incorporated in various extensions of the classical Boolean systems based on *fuzzy-set theory* [6–9]. In the standard fuzzy-set model of retrieval, the document terms may be weighted and the queries are processed as standard Boolean formulations. Given queries "*A* **or** *B*," "*A* **and** *B*," and "**not** *A*," a document *X* with weights $d_A(X)$ and $d_B(X)$ for terms *A* and *B*, respectively, receives the following retrieval values in a fuzzy-set retrieval system:

$\max[d_A(X), d_B(X)]$ for query (*A* **or** *B*)
$\min[d_A(X), d_B(X)]$ for query (*A* **and** *B*)
$1 - d_A(X)$ for query (**not** *A*)

For binary document vectors where the document term weights are restricted to 0 and 1, the fuzzy-set model is compatible with the Boolean system of retrieval. Unfortunately, the fuzzy-set system suffers from a lack of discrimination among the retrieved output nearly to the same extent as the conventional Boolean retrieval system, since the rank of a retrieved item depends only on the lowest or highest weighted document term for **and** and **or** queries, respectively. In addition, it is difficult to extend the fuzzy-set model to situations where term weights are also attached to query terms instead of only to the document terms [6–9].

In the remainder of this study, an extended Boolean retrieval model is introduced that accommodates both weighted query and weighted document terms. The extended model represents a compromise between the strictness of the conventional Boolean system and the lack of structure inherent in the vector-processing system. The extended model thus preserves the query structure inherent in a Boolean system with its distinction between term phrases ("**and**ed" terms) and term synonyms ("**or**ed" terms); at the same time, the model makes it possible to retrieve items that are not retrievable by a conventional Boolean system, and all retrieved items are ranked in decreasing order of query-document similarity.

The basic model is introduced in the remainder of this study and evaluation results are used to illustrate the effectiveness of the system.

2. THE EXTENDED RETRIEVAL MODEL

2.1. Motivation

Consider first the operations of a conventional retrieval system based on Boolean query formulations. Three document classes may be distinguished with respect to two-term queries such as (*A* **or** *B*) and (*A* **and** *B*): those exhibiting both terms, those containing only one of the terms, and those containing neither term. The **or**-type query assigns a value of 0 to the items containing neither term and values of 1 to the remaining items; the **and**-query assigns a value of 1 to the items containing both terms and values of 0 to the remainder. This situation is represented in Table I(*a*).

When only two terms are under consideration, the term assigment can be repesented by a two-dimensional map, as shown in Figure 1, where each term is assigned a different coordinate axis. It is clear that for **and**-queries, the (1, 1) point on the map, representing the situation when both terms are present in an item, is the desirable location; for **or**-queries, on the other hand, the (0, 0) point identifying the situation when both terms are absent from an item is to be avoided. This suggests that a discriminating retrieval system might rank the stored items in order of increasing distance from the (1, 1) point for **and**-queries, and in order of decreasing distance from (0, 0) for the **or**-queries. For a document with term weights d_A and d_B for terms *A* and *B*, respectively, the normal Euclidian distance from the (0, 0) point is $\sqrt{(d_A - 0)^2 + (d_B - 0)^2}$, whereas the distance from the (1, 1) point is $\sqrt{(1 - d_A)^2 + (1 - d_B)^2}$. It is convenient to operate with normalized distance measures by dividing out the maximum distance of $\sqrt{2}$ between the (0, 0) and (1, 1) points. For

TABLE I. Query-Document Similarity Values for Conventional and Extended Systems.

(*a*) Conventional Boolean Retrieval

	Terms		Similarity with Query	
	A	*B*	(*A* or *B*)	(*A and B*)
Document D_1	1	1	1	1
Document D_2	1	0	1	0
Document D_3	0	1	1	0
Document D_4	0	0	0	0

(*b*) Extended Retrieval

Document D_1	1	1	1	1
Document D_2	1	0	$1/\sqrt{2}$	$1\text{-}1/\sqrt{2}$
Document D_3	0	1	$1/\sqrt{2}$	$1\text{-}1/\sqrt{2}$
Document D_4	0	0	0	0

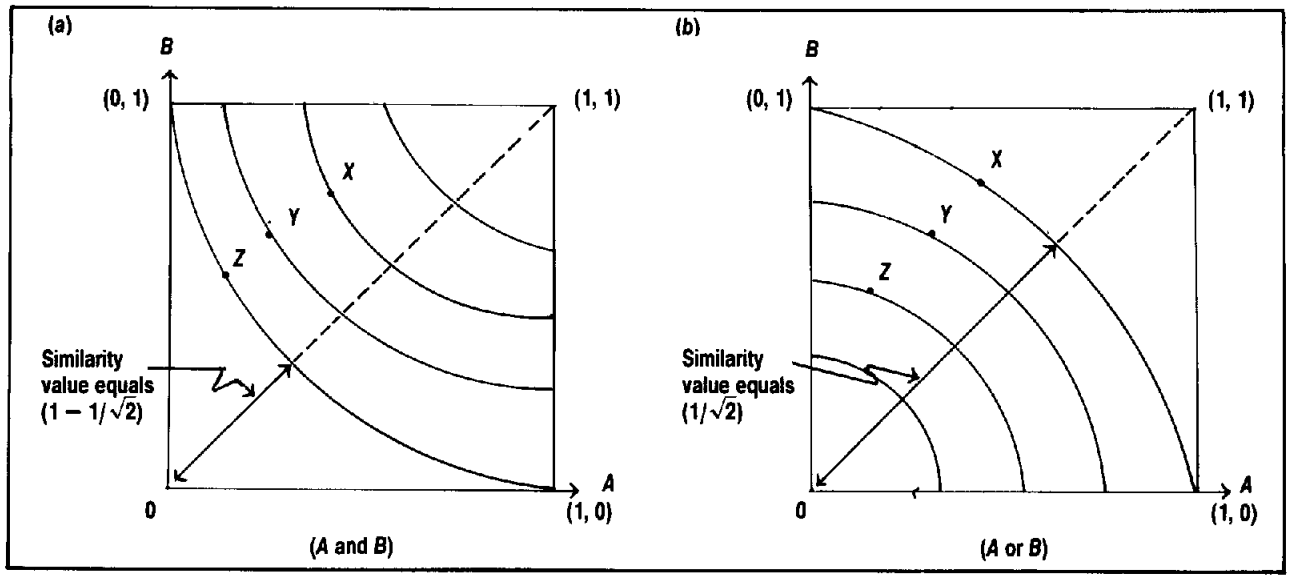

FIGURE 1. Equidistance lines from (1, 1) point for "and" and from (0, 0) point for "or." (a) Lines of equal similarity equidistant from (1, 1). (b) Lines of equal similarity equidistant from (0, 0).

$0 \le d_A, d_B \le 1$, the following similarity measures between queries Q and documents D will then rank the documents in decreasing distance from (0, 0) and increasing distance from (1, 1), respectively.

$$\text{sim}(D, Q_{(A \text{ or } B)}) = \sqrt{\frac{d_A^2 + d_B^2}{2}}$$

$$\text{sim}(D, Q_{(A \text{ and } B)}) = 1 - \sqrt{\frac{(1 - d_A)^2 + (1 - d_B)^2}{2}} \quad (1)$$

Table 1(*b*) shows the similarity values obtained by using formulas (1) for the sample documents previously used in Table 1(*a*). It may be seen that when only one query term is present in a given document, the document receives values of $1/\sqrt{2}$ and $1 - 1/\sqrt{2}$ for **or** and **and** queries, respectively, rather than 1 and 0 as in the Boolean system. That is, the presence of a single term in a document is not worth as much as the presence of both terms for **or**-queries, but is worth more than the absence of both terms for the **and** case.

The formulas of expression (1) are directly applicable to the case of *weighted-document terms* when $0 \le d_A, d_B \le 1$. In that case, each document is represented by a point on the unit square as shown in Figure 1. The locus of equidistant points from (1, 1) and (0, 0) for **and** and **or**, respectively, is shown in Figure 1 by appropriate lines on the graph. Three documents labeled *X*, *Y*, and *Z* are used as examples in Figure 1. It may be seen that the similarity computation of expression (1) produces a ranking where sim(*Q*, *X*) > sim(*Q*, *Y*) > sim(*Q*, *Z*) with respect to both (*A* **and** *B*) and (*A* **or** *B*).

The basic similarity measurements of expression (1) can be extended to the case of *weighted-query terms* based on term weights $0 \le a, b \le 1$ reflecting the importance of the individual terms *A*, *B* in the query. In that case

$$\text{sim}(D, Q_{([A,a] \text{ or } [B,b])}) = \sqrt{\frac{a^2 d_A^2 + b^2 d_B^2}{a^2 + b^2}}$$

$$\text{sim}(D, Q_{([A,a] \text{ and } [B,b])}) \quad (2)$$

$$= 1 - \sqrt{\frac{a^2(1 - d_A)^2 + b^2(1 - d_B)^2}{a^2 + b^2}}$$

When the query terms are fully weighted, that is, $a = b = 1$, the similarity measures of expression (2) reduce to the basic formulas of expression (1).

In the foregoing development, it was assumed that the query-document similarity could be measured by using a normalized Euclidian distance between corresponding points in the document space. The notion of distance between points in a document space can be generalized by introducing the well-known L_p vector norm defined for an *n*-dimensional vector $(d_1, d_2, \ldots, d_n)$ as

$$\|\mathbf{d}\|_p = \|d_1, d_2, \ldots, d_n\|_p = (d_1^p + d_2^p + \cdots + d_n^p)^{1/p} \quad (3)$$

Since normalized distances are to be used to measure query-document similarities, expression (3) can be rewritten in the form

$$\|d\|_p(\text{normalized}) = \left(\frac{1}{n}\right)^{1/p} \|d\|_p = \left[\frac{(d_1^p + d_2^p + \cdots + d_n^p)}{n}\right]^{1/p} \quad (4)$$

Expression (4) exhibits the form previously introduced in (1). The theory of vector norms will now be used as a model for query-document comparisons in a Boolean query environment [10, 11].

2.2 The *p* Norm Model

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

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

where a_i indicates the weight of query term A_i, $0 \le a_i \le 1$ and $1 \le p \le \infty$. Similarly, a generalized **and**-query, Q_{and} is written as

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

The similarities between a given document $D = (d_{A_1} d_{A_2}, \ldots, d_{A_n})$ and $Q_{or(p)}$, $Q_{and(p)}$ may now be defined as[1,2]

$$\text{sim}(D, Q_{or(p)}) = \left[\frac{a_1^p d_{A_1}^p + a_2^p d_{A_2}^p + \cdots + a_n^p d_{A_n}^p}{a_1^p + a_2^p + \cdots + a_n^p}\right]^{1/p} \quad (5)$$

$$\text{sim}(D, Q_{and(p)}) = 1 - \left[\frac{a_1^p(1 - d_{A_1})^p + a_2^p(1 - d_{A_2})^p + \cdots + a_n^p(1 - d_{A_n})^p}{a_1^p + a_2^p + \cdots + a_n^p}\right]^{1/p} \quad (6)$$

It can now be shown that when expressions (5) and (6) are used to compute the similarities between a set of (possible weighted) document terms and a (possibly weighted) **or**-query and **and**-query, respectively, the effect of a standard vector-processing retrieval model is obtained when p is set equal to 1. When $p = \infty$ and the query and document term weights are limited to 0 or 1, a conventional Boolean retrieval model is produced. Finally, for intermediate values of p between 1 and ∞, a retrieval system intermediate between the vector-processing and the Boolean models is obtained [12].

Consider first the situation where $p = 1$. In that case, it is simple to show that $\text{sim}(D, Q_{or}) = \text{sim}(D, Q_{and})$, because

$$\text{sim}(D, Q_{and(1)}) = 1 - \left[\frac{a_1(1 - d_{A_1}) + a_2(1 - d_{A_2}) + \cdots + a_n(1 - d_{A_n})}{a_1 + a_2 + \cdots + a_n}\right]$$

$$= 1 - \left[\frac{(a_1 + a_2 + \cdots + a_n) - (a_1 d_{A_1} + a_2 d_{A_2} + \cdots + a_n d_{A_n})}{a_1 + a_2 + \cdots + a_n}\right]$$

$$= \frac{a_1 d_{A_1} + a_2 d_{A_2} + \cdots + a_n d_{A_n}}{a_1 + a_2 + \cdots + a_n}$$

$$= \text{sim}(D, Q_{or(1)}) \quad (7)$$

Expression (7) represents the inner product between a document

$$D = (d_{A_1}, d_{A_2}, \ldots, d_{A_n}) \text{ and query } Q = \left(\frac{a_1}{a_1 + a_2 + \cdots + a_2}, \ldots, \frac{a_n}{a_1, + \cdots +, a_n}\right)$$

In other words, when $p = 1$, the distinction between the **and** and **or** connectives in a query disappears and a simple vector-processing retrieval model is obtained where the similarities between queries and documents are measured by the inner product between the document term weights d_{A_i} and the normalized query weights w_i, represented as

$$w_i = \left(\frac{a_i}{a_1 + a_2 + \cdots + a_n}\right)$$

[1] The query weights a_i are relative rather than absolute weights used to represent the presumed importance of one query term relative to the other query terms. Because of the normalization inherent in the denominator of expressions (5) and (6), the restriction $0 \leq a_i \leq 1$ is actually unnecessary. Query weights larger than 1 are acceptable, provided the relative importance of the terms is properly reflected by the term weights.

[2] The notations **or**^{p}, **and**^{p} as well as **or**(p), **and**(p) are used interchangeably.

When $p = \infty$, one obtains

$$\text{sim}(D, Q_{and(\infty)}) = \lim_{p \to \infty} 1 - \left[\frac{a_1^p(1 - d_{A_1})^p + \cdots + a_n^p(1 - d_{A_n})^p}{a_1^p + a_2^p + \cdots + a_n^p}\right]^{1/p} = 1 - \frac{\max[a_1(1 - d_{A_1}), \ldots, a_n(1 - d_{A_n})]}{\max[a_1, a_2, \ldots, a_n]}$$

Assuming that the query terms are all equally weighted, such as when $a_1 = a_2 = \cdots = a_n = 1$, one has

$$\text{sim}(D, Q_{and(\infty)}) = 1 - \max[(1 - d_{A_1}), (1 - d_{A_2}), \ldots, (1 - d_{A_n})] = \min[d_{A_1}, d_{A_2}, \ldots, d_{A_n}]$$

Similarly,

$$\text{sim}(D, Q_{or(\infty)}) = \lim_{p \to \infty} \left[\frac{a_1^p d_{A_1}^p + a_2^p d_{A_2}^p + \cdots + a_n^p d_{A_n}^p}{a_1^p + a_2^p + \cdots + a_n^p}\right]^{1/p} = \frac{\max[a_1 d_{A_1}, a_2 d_{A_2}, \ldots, a_n d_{A_n}]}{\max[a_1, a_2, \ldots, a_n]}$$

When all query term weights are equal to 1, that is, $a_1 = a_2 = \cdots = a_n = 1$, one obtains

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

This leads to the conclusion that when $p = \infty$ and the query is unweighted, the query-document matching function is dependent only on the document term of highest weight for Q_{or} and the document term of lowest weight for Q_{and}. This is precisely the situation previously mentioned for the fuzzy-set model of retrieval, and by extension, for the conventional Boolean retrieval system when both query and document terms are unweighted.

By varying the value of p between 1 and ∞ and using the query-document similarity function of expressions (5) and (6), it is then possible to obtain a system intermediate between a pure vector-processing model ($p = 1$) and a conventional Boolean retrieval system ($p = \infty$). The larger the value of p, the more importance is given to the query structure as reflected by the **and** and **or** connections. As the p-value decreases, the distinction between an **and** connection and an **or** connection becomes weaker, until that distinction disappears completely as p reaches a lower bound of 1. The effect of the p-value variations is represented schematically in Figure 2.

The theory of vector norms can be used to show that the query-document similarity values obtained for $1 < p < \infty$ are indeed intermediate between those obtainable for the extreme cases when $p = 1$ and $p = \infty$. Indeed

$$\text{sim}(D, Q_{and(\infty)}) \leq \text{sim}(D, Q_{and(p)}) \leq \text{sim}(D, Q_{and(1)}) = \text{sim}(D, Q_{or(1)}) \leq \text{sim}(D, Q_{or(p)}) \leq \text{sim}(D, Q_{or(\infty)}) \quad (8)$$

The various query-document similarities of expression (8) are strictly equal when all document term weights are equal, that is, when $d_{A_1} = d_{A_2} = \cdots = d_{A_n}$. An evaluation of expressions (5) and (6) for that case shows that the query-document simi-

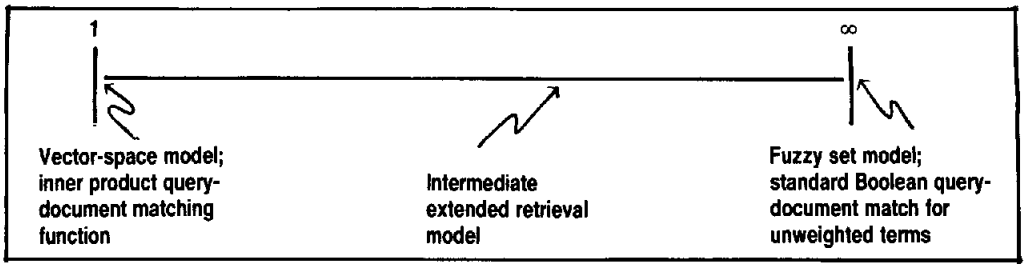

FIGURE 2. Range of p-value variations.

larity values obtained for various values of p all reduce to d_A, the common document term value.

When the document terms are not all equally weighted, the strict inequality signs are valid for the query-document similarities of expression (8), except, of course, between sim(D, $Q_{\text{and}(1)}$) and sim(D, $Q_{\text{or}(1)}$). These two expressions are always strictly equal for any document, as shown earlier in (7). The situation may be illustrated in two dimensions by taking a document $D = (d_A, d_B)$ and queries Q equal to (A **and**(∞) B), (A **and**(p) B), (A **and**(1) B) = (A **or**(1) B), (A **or**(p) B), and (A **or**(∞) B), respectively. In that case, one has

$$\min(d_A, d_B) \leq 1 - \left[\frac{(1 - d_A)^p + (1 - d_B)^p}{2}\right]^{1/p} \leq \frac{d_A + d_B}{2} \leq \left[\frac{d_A^p + d_B^p}{2}\right]^{1/p} \leq \max(d_A, d_B) \quad (9)$$

This is proved by setting $d_B = d_A(1 + \epsilon)$ for $\epsilon > 0$ and $d_B > d_A$ and expanding the formulas in (9).

The two-dimensional case is represented in Figure 3(*a*) where loci of equal similarity are shown for the cases $p = 1$, $p = 2$, and $p = \infty$, respectively. By measuring the distance between each locus and the (0, 0) point along the diagonal (from (0, 0) to (1, 1)], it is clear that

i) sim(D, A **and**(∞) B) = min(d_A, d_B) (proportional to $\overline{OV}$).

ii) $$\text{sim}(D, A \textbf{ and}(2)\ B) = 1 - \left[\frac{(1 - d_A)^2 + (1 - d_B)^2}{2}\right]^{1/2}$$ (proportional to $\overline{OX}$).

iii) sim(D, A **and**(1) B) = average (d_A, d_B) (proportional to $\overline{OY}$).

iv) $$\text{sim}(D, A \textbf{ or}(2)\ B) = \left[\frac{d_A^2 + d_B^2}{2}\right]^{1/2}$$ (proportional to $\overline{OW}$).

v) sim(D, A **or**(∞) B) = max(d_A, d_B) (proportional to $\overline{OZ}$).

More generally, as p changes from 2 to ∞, the similarity values between document D and query (A **and**(p) B) will lie between $\overline{OX}$ and $\overline{OV}$; similarly, the values between D and (A **or**(p) B) lie between $\overline{OW}$ and $\overline{OZ}$. When $d_A = d_B$, a document D lies along the (0, 1) diagonal and all similarity values are proportional to the distance $\overline{OD}$ as shown in Figure 4.

2.3 Computation of Query-Document Similarities in the Extended System

The computation of a particular document value corresponding to a given extended Boolean query statement (that is, of the query-document similarity) follows the model used for standard Boolean query statements in a conventional retrieval system. One possible method generates the final document value recursively by first taking the document value with respect to single query terms, then with respect to two-term clauses each containing two single terms, then with respect to

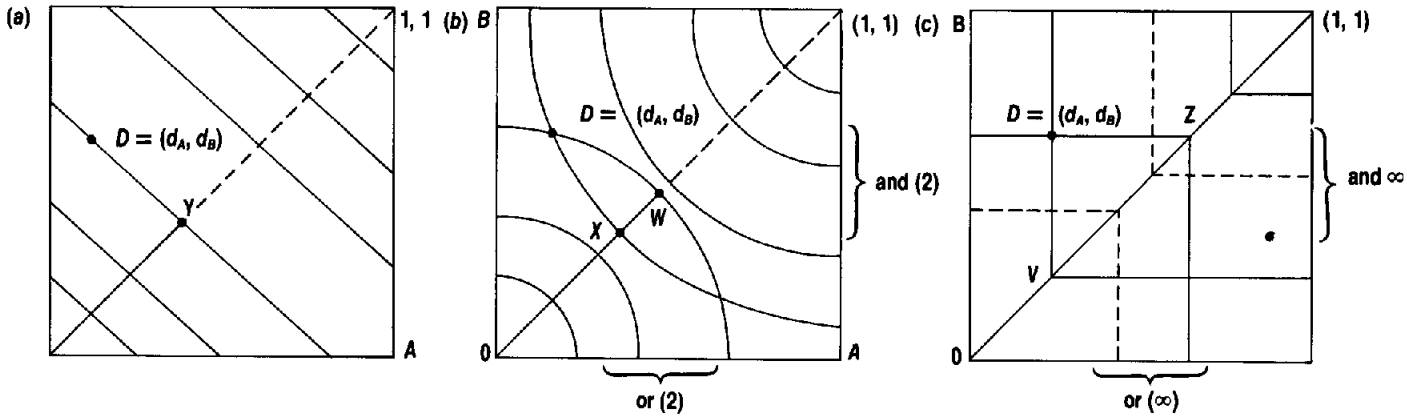

FIGURE 3. Query-document similarities for $D = (d_A, d_B)$ when $d_A \neq d_B$ [and (∞) < and (2) < and (1) = or (1) < or (2) < or (∞)]. (*a*) Equal similarity lines for $p = 1$ (or (1) = and (1)); (*b*) Equal similarity lines for $p = 2$ [$\overline{OW} > \overline{OX}$]; (*c*) Equal similarity line for $p = \infty$ [$\overline{OZ} > \overline{OV}$].

larger clauses containing one or more initial two-term clauses and so on until the complete query is considered. For example, to obtain the document value with respect to a query such as [(*A* **and** *B*) **or** *C*], one first finds the individual values of terms *A*, *B*, and *C* in the document, that is, d_A, d_B, and d_C. In a Boolean system, these values will be equal to 1 for terms present in the document and 0 for terms not included in the document. One next proceeds to find the document value with respect to the clause (*A* **and** *B*), that is, $d_{(A \text{ and } B)}$. This will be equal to 1 if both $d_A = 1$ and $d_B = 1$ and 0 otherwise. Finally one finds the document value for the complete query $d_{[(A \text{ and } B) \text{ or } C]}$, which equals 1 if either $d_{(A \text{ and } B)}$ or d_C are equal to 1. These computations are illustrated in the left-hand portion of Table II.

In the extended Boolean query system, the situation is complicated in two ways: first, the term and clause weights, which are restricted to 0 and 1 only in the conventional Boolean system, now range anywhere from 0 to 1; second, a *p*-value is assigned to the Boolean connectives. Some typical recursive computations for the extended system are illustrated in the right-hand portion of Table II. Each partial computation in the Boolean system produces a document value of 0 or 1; in the extended system, all document values d_ψ are in the range from 0 to 1.

Consider as an example, a document $D = (d_A, d_B, d_C)$ where $d_A = 1$, $d_B = 0$, and $d_C = 0.5$, and a query $Q = (\langle \psi, 0.2 \rangle$ **or** (2) $\langle C, 0.1 \rangle)$ where in turn $\psi = (\langle A, 0.3 \rangle$ **and** (2) $\langle B, 0.4 \rangle)$. The recursive computation might then proceed as follows:

$$d_\psi = 1 - \sqrt{\frac{(0.3)^2(1-1)^2 + (0.4)^2(1-0)^2}{(0.3)^2 + (0.4)^2}}$$

$$= 1 - 4/5 = 0.2$$

$$d_Q = \sqrt{\frac{(0.2)^2(0.2)^2 + (0.1)^2(0.5)^2}{(0.2)^2 + (0.1)^2}}$$

$$= \sqrt{\frac{41}{500}} = 0.286$$

In extended retrieval, the query terms and query clauses are weighted, and a normalized distance is used to measure the document values with respect to the extended Boolean queries. As a result, most of the equivalence properties that hold for conventional Boolean query statements must be modified by addition of appropriate scale factors in the extended system. The following equivalences may be proved in the extended system [12].

quasi-idempotency

(A, a_1) **and** (p) $(A, a_2) = A$
(A, a_1) **or** (p) $(A, a_2) = A$
(A, a_1) **and** (p) (A, a_2) **and** (p) (B, b)
$= [A, (a_1^p + a_2^p)^{1/p}]$ **and** (p) (B, b)
(A, a_1) **or** (p) (A, a_2) **or** (p) $(B, b) = (A, (a_1^p + a_2^p)^{1/p})$ **or** (p) (B, b)

commutativity

(A_1, a_1) **and** (p) $(A_2, a_2) = (A_2, a_2)$ **and** (p) (A_1, a_1)
(A_1, a_1) **or** (p) $(A_2, a_2) = (A_2, a_2)$ **or** (p) (A_1, a_1)

duality

$\neg [(A_1, a_1)$ **and** (p) $(A_2, a_2)] = \neg (A_1, a_1)$ **or** (p) $\neg (A_2, a_2)$
$\neg [(A_1, a_1)$ **or** (p) $(A_2, a_2)] = \neg (A_1, a_1)$ **and** (p) $\neg (A_2, a_2)$

involution

$\neg \neg [(A_1, a_1)$ **and** (p) $(A_2, a_2)] = (A_1, a_1)$ **and** (p) (A_2, a_2)
$\neg \neg [(A_1, a_1)$ **or** (p) $(A_2, a_2)] = (A_1, a_1)$ **or** (p) (A_2, a_2)

quasi-associativity

$\{[(A_1, a_1)$ **or** (p) $(A_2, a_2)], a\}$ **or** (p) (B, b)
$= [(A_1, aa_1)$ **or** (p) $(A_2, aa_2)]$ **or** (p) $[B, b(a_1^p + a_2^p)^{1/p}]$
$\{[(A_1, a_1)$ **and** (p) $(A_2, a_2)], a\}$ **and** (p) (B, b)
$= [(A_1, aa_1)$ **and** (p) $(A_2, aa_2)]$ **and** (p) $(B, b(a_1^p + a_2^p)^{1/p})$

In the extended retrieval system, the distributive laws that hold for conventional Boolean expressions are not valid. That is, it is not generally the case that

$[(A, a)$ **and** (p) $(B, b)]$ **or** (p) (C, c)
$= [(A, a)$ **or** (p) $(C, c)]$ **and** (p) $[(B, b,)$ **or** (p) $(C, c)]$

and

$[(A, a)$ **or** (p) $(B, b)]$ **and** (p) (C, c)
$= [(A, a)$ **and** (p) $(C, c)]$ **or** (p) $[(B, b)$
and (p) $(C, c)]$ (10)

In conventional Boolean retrieval, a Boolean **and** represents an absolute requirement for the presence of several terms (a phrase of individual terms), whereas a Boolean **or** identifies a set of quasisynonyms for which each term is worth as much as all other terms. In these circumstances, the distributive laws are easy to rationalize. Indeed the query ((*A* **or** *B*) **and** *C*)

TABLE II. Sample Recursive Document Value Computation

Boolean Query	Document Value (Query-Document Similarity)	Extended Boolean Query	Document Value (Query-Document Similarity)
$\psi = A = [(A, 1)]$ $\psi' = \neg A = [(\neg A, 1)]$	$d_\psi = 1$ if $d_A = 1$; 0 otherwise $d_{\psi'} = 0$ if $d_A = 1$; 0 otherwise	$\psi = [(A, a)]$ $\psi' = \neg[(A, a)]$	$d_\psi = ad_A$ $d_{\psi'} = 1 - d_\psi$
$\psi_1 = (A \text{ or } B)$ $= [(A, 1) \text{ or } (B, 1)]$	$d_{\psi_1} = \begin{matrix}1 \text{ if } d_A = 1 \text{ or } d_B = 1\\ 0 \text{ otherwise}\end{matrix}$	$\psi_1 = [(A, a) \text{ or } (p) (B, b)]$	$d_{\psi_1} = \left[\frac{a^p d_A^p + b^p d_B^p}{a^p + b^p}\right]^{1/p}$
$\psi_2 = (A \text{ and } B)$ $= [(A, 1) \text{ and } (B, 1)]$	$d_{\psi_2} = \begin{matrix}1 \text{ if } d_A = 1 \text{ and } d_B = 1\\ 0 \text{ otherwise}\end{matrix}$	$\psi_2 = [(A, a) \text{ and } (p) (B, b)]$	$d_{\psi_2} = 1 - \left[\frac{a^p(1-d_A)^p + b^p(1-d_B)^p}{a^p + b^p}\right]^{1/p}$
$X_1 = (\psi_1 \text{ and } C)$ $= [(\psi_1, 1) \text{ or } (C, 1)]$	$dX_1 = \begin{matrix}1 \text{ if } d_{\psi_1} = 1 \text{ or } d_C = 1\\ 0 \text{ otherwise}\end{matrix}$	$X_1 = [(\psi_1, d) \text{ or } (p) (C, c)]$	$dX_1 = \left[\frac{d^p d_{\psi_1}^p + c^p d_c^p}{d^p + c^p}\right]^{1/p}$
$X_2 = (\psi_1 \text{ and } C)$ $= [(\psi_1, 1) \text{ and } (C, 1)]$	$dX_2 = \begin{matrix}1 \text{ if } d_{\psi_1} = 1 \text{ and } d_C = 1\\ 0 \text{ otherwise}\end{matrix}$	$X_2 = [(\psi_1, d) \text{ and } (p) (C, c)]$	$dX_2 = 1 - \left[\frac{d^p(1-d_{\psi_1})^p + c^p(1-d_c)^p}{d^p + c^p}\right]^{1/p}$
		$e^\psi \ (0 \le e \le 1)$	$e \cdot d_\psi \ (0 \le e \cdot d_\psi \le 1)$

says that one of either *A* or *B* is needed in addition to *C*, and quite clearly this implies either *A* and *C* or else *B* and *C*. In the extended system, the phrase relation may hold to some extent depending on the p-value attached to the **and** connective, and the synonym (thesaurus) relationship holds to some extent depending on the p value used for **or**. The interaction between phrase and thesaurus relations is complex and there is no obvious reason why the two parts of expression (10) should then be strictly equivalent. In practice, any differences in document values obtained for the two formulas of expressions (10) should be very small.

3. EXPERIMENTAL DESIGN FOR EXTENDED RETRIEVAL

3.1 Main Properties of Extended Retrieval

The extended retrieval system exhibits the following practical advantages for information retrieval:

1. Structured queries can be processed, including the usual **and**, **or**, and **not** operators.
2. Term weights can be used to reflect relative term importance for terms assigned to documents and queries; in addition, the individual query components and clauses can be weighted separately. This leads to sophisticated query formulations and to the ranking of retrieved items in decreasing order of query-document similarity.
3. The interpretation of the query structure can be altered by using different p-values to compute the query-document similarity. As the p-value changes from ∞ to 1, the Boolean connectives are interpreted more and more loosely. Eventually, when *p* reaches 1, the distinction between compulsory phrase bonding (using **and**) and between alternative synonym specification (using **or**) is completely lost, and only the presence or absence of the terms is taken into account.

The use of the p-norm metric opens a number of attractive possibilities for the enhancement of information retrieval operations. The most obvious application is the use of the *p*-values to simulate flexible phrase and thesaurus assignment processes. The use of term phrases and thesaurus classes constitutes a principal tool in text analysis and automatic indexing [13–16]:

1. Phrases are useful to render broad, high-frequency terms more specific. Broad terms with high document frequencies must be incorporated into term phrases to obtain adequate search precision.
2. Thesaurus classes are most useful as a means for broadening low frequency terms that by themselves are too rare and too narrow. By grouping related low-frequency terms into common classes and assigning these thesaurus classes to the bibliographic items instead of the individual terms, the search recall can be appropriately enhanced.

The use of different *p*-values leads to varying interpretations of thesaurus and phrase relations as shown in Table III. Values of *p* close to infinity produce strict interpretations of phrase relationships and synonym classes. Low values of *p* are used for coordinate-level-type query-document comparisons where the terms are assumed to be independent of each other, and the number and weight of matching query and document terms determines the retrieval ranks. Intermediate p-values produce a compromise between the complete absence of terms relations and the strict Boolean interpretation of the term relationships.

Table III. *p*-Value Interpretation

p Value	Connective	Resulting Semantic Relation
∞	*and*	Strict phrase assignment; item not retrievable unless all phrase components are present;
∞	*or*	Strict thesaurus feature; terms related by "**or**" are substitutable one for another; only one of each group of related terms is required;
3	*and*	Loose phrase; the presence of all phrase components is worth more than the presence of only some of the components; terms are not compulsory;
3	*or*	Loose thesaurus class; the presence of several terms from a given class is worth more than the presence of only one term;
1	*or, and*	Terms are independent of each other; distinction between phrase and thesaurus assignment disappears.

One attractive concept is the use of mixed *p*-values to distinguish compulsory terms or term relations from optional ones. This makes it possible to include in a single query-statement tentative content identifiers using low p-values, as well as compulsory, objective terms using high *p*-values. For example, bibliographic items in computer science authored by "Smith" may be specified as

[(computer **and** (2) science) **or** (2)
(computer **and** (2) engineering)]
and (∞) (author 'Smith')

The first part of the query statement is used to specify items in the computer area, using loose phrase and synonym specifications. The high *p*-value attached to the final **and** will restrict the output to the specified author.

The match with mixed *p*-values thus embodies aspects of bibliographic as well as of database retrieval. When **and** (∞) and **or** (∞) operators are used, a standard database retrieval environment is produced. The term weights are then usable to rank the output items in decreasing order of query-document similarity. Alternatively, when low p-values are utilized, a vector-type matching system is produced. The extended system is thus usable to simulate a variety of systems incorporating database as well as bibliographic retrieval aspects. For example, in the SIRE system [5], a strict Boolean match is used, followed by the additional ranking of the output in order according to the sum of the weights of matching query and document terms. This type of retrieval is obtainable directly in the metric system by first constructing standard Boolean queries $Q_{(\infty)}$, which are then "flattened," that is, turned into vector-type queries for output ranking purposes. This is achieved by replacing **and** (∞) and **or** (∞) by **and** (1) and **or** (1) operators. For each $Q_{(\infty)}$, a new query statement $Q_{(1)}$ is then obtained which may be processed in one operation as "$Q_{(\infty)}$ **and** (∞) $Q_{(1)}$." For the query [(computer **and** science) **or** (computer and engineering)], the final formulation will be

[(computer **and** (∞) science) **or** (∞) (computer **and** (∞) engineering)] **and** (∞) [(computer **and** (1) science) **or** (1) (computer **and** (1) engineering)]

The mixed query system is also useful in a *relevance feedback* environment, where relevance information obtained from users in response to previous retrieval activities is used to reformulate the initial query statements in preparation for further search activities [17, 18]. In that case, probabilistic term *relevance* weights may be generated leading to increas-

ingly useful query specifications [19–21]. In an extended retrieval environment, high p-values could be used for terms whose occurrence characteristics in the relevant documents are well specified, whereas low p-values would serve for terms that are less well-specified. The extended system then serves as a model for both the standard interactive relevance-feedback system using vector-type queries, as well as for the reformulation of Boolean query statements in conventional retrieval environments. [22]

3.2 Query and Document Processing

The processing environment used in this study may be illustrated by using a typical two-clause query as an example. The basic two-clause query takes the form

$$\begin{matrix}\textbf{and}\\ \textbf{or}\end{matrix}\left[\begin{matrix}\textbf{and}\\ \textbf{or}\end{matrix}(\text{term } A, \text{ term } B, \ldots), \begin{matrix}\textbf{and}\\ \textbf{or}\end{matrix}(\text{term } C, \text{ term } D, \ldots)\right]$$

where the terms included in a given clause are connected by a common Boolean operation and an outer clause operator is used to relate the clauses. In each case, one of the two operators (**and, or**) is inserted as a term or clause connector. In the extended retrieval system, the query terms are individually weighted and additional weights can be assigned to the clauses as previously discussed.

A sample query is shown in Table IV in its original Boolean form in part(*a*). Part(*b*) shows the equivalent expanded form in the extended query system. Each Boolean operator is now assigned a p-value [equal to infinity in Table IV(*b*) to correspond with the standard interpretation of the Boolean query of Table IV(*a*)], and each term and clause receives a term weight of 1. The clause weights are underlined for easy identification. The formulation of Table IV(*b*) is precisely the same as that of Table IV(*a*).

The formulation of Table IV(*c*) corresponds to that used in Table IV(*b*) except for the addition of relative term and clause weights and of variable p-values attached to the connectives. The outer clause operator is identified as p_1, $1 \leq p_1 \leq \infty$, and the inner operators by p_2, $1 \leq p_2 \leq \infty$. It is known that in the absence of specific information about the actual importance of the terms, the *inverse document frequency* represents a useful indication of term importance [23]. The inverse document frequency weight arranges the terms in inverse order according to the number of documents in a collection in which the term occurs. That is, terms occurring in many documents receive low weights, whereas terms occurring in few documents are highly weighted. Specifically, if N is the number of documents in a collection and n_k is the number of documents in which term k occurs, then the inverse document frequency of term k, idf_k, is defined as

$$idf_k = \log N/n_k \tag{11}$$

In the query formulation of Table IV(*c*), an inverse document frequency weight, idf_k, is attached to each term, and the clause weight is computed as the average weight of terms in the corresponding clause (ave. *idf*). A vector formulation for the sample query is shown in Table IV(*d*). In that case, the term weights are maintained, but the distinction between **and** and **or** operators disappears since all p-values are then fixed at 1.

The weights attached to document terms can in principle be determined in the same way as the query term weights. That is, binary weights equal to 1 can be used for terms that are present in a given document and weights equal to 0 are attached to terms not included in a document. Alternatively, the inverse document frequency weights of expression (11) can be used. A third possibility consists in defining the term weight as a combination of two different values [13–16]:

1. A value based on the occurrence characteristics of the term in the whole collection, typically the inverse document frequency weight of expression (11).
2. A value based on the occurrence characteristics of the terms in the individual documents; typically, this factor could be computed as the frequency of occurrence or *term frequency tf* of a term in a given document.

The combined weight w_{ik} of term k in document i can then be defined as

$$w_{ik} = \text{idf}_k \cdot tf_{ik} \tag{12}$$

According to formula (12) a term is highly weighted when it exhibits a high frequency in an individual document but a low overall number of occurrences in the whole collection. In the experiments to be described in the sequel, a normalized form of expression (12) is used, designated *tf·idf*, and defined as

$$(tf \cdot idf)_{ik} = \left[\frac{tf_{ik}}{\max\limits_{\text{term } h \text{ in doc } i}(tf_{ih})}\right]\left[\frac{idf_k}{\max\limits_{\text{term } h \text{ in doc } i}(idf_h)}\right] \tag{13}$$

Experimental results are described in the remainder of this study in which various combinations of document and query term weights including binary and combined *tf·idf* values are used for illustration.

TABLE IV. Typical Query Formulations

(*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** (p_1) ((**or** (p_2) ((catalogue, 6.358), (catalog, 5.358)), 5.898)
> **or** (p_2) ((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))

3.3. Typical Retrieval Results

The retrieval results obtainable in the extended system are illustrated in Table V for the query previously used as an example in Table IV. Five documents, numbered 1, 11, 36, 47, and 51, are relevant to the query in a collection of 82 documents. The query term frequencies in the five relevant documents are shown in Table V(*b*). It may be seen that at least one term from each of the two query clauses appears in documents 1, 11, and 36. On the other hand, neither the term *catalog* nor *catalogue* appear in documents 47 and 51. These

TABLE V. Typical Query Operations

(a) Query Formulation (Boolean form)
(catalogue or catalog) **and** (mechanization **or** automation **or** computerization)

(b) Query Term Frequencies in Relevant Documents
(relevant document numbers **1, 11**, 36, 47, 51)

Document Number	Query Clause 1		Query Clause 2		
	Catalogue	Catalog	Automation	Computerization	Mechanization
Doc 1	—	1	—	1	4
Doc 11	—	2	2	—	4
Doc 36	4	—	—	2	—
Doc 47	—	—	2	—	1
Doc 51	—	—	—	2	2

(c) Rank of Retrieved Relevant Documents

Boolean form, $p = \infty$ binary weights			Extended form, $p = 2$ binary weights			Vector formulation, cosine tf · idf weights		
Similarity with Query	Document Rank	Document Number	Similarity with Query	Document Rank	Document Number	Similarity with Query	Document Rank	Document Number
1	1	11	0.756	1	11	0.384	1	36
1	2	36	0.756	2	1	0.374	2	11
1	3	1	0.636	3	36	0.186	4	1
			0.281	7	47	0.156	5	51
			0.281	9	51	0.125	6	47

last two documents are therefore not retrievable by the conventional Boolean query formulation.

The actual retrieval ranks achieved by the five relevant documents are shown in Table V(c) for the conventional Boolean form as well as for the extended matching system using a p-value of 2 for **and** and **or** operators and binary term weighting, and finally for a vector query-formulation using p-values of 1 and a query-document matching system based on the cosine similarity coefficient [3, 4]. The Boolean formulation retrieves the first three documents (numbers 11, 36, and 1) with a perfect query-document similarity of 1. These same three documents are also retrieved with ranks 1, 2, and 3 by the extended system where the similarity coefficients are equal to 0.756, 0.756, and 0.636, respectively. In addition, the extended system also retrieves the other two relevant documents (numbers 47 and 51) with ranks 7 and 9, respectively, and much lower query-document similarities of 0.281. Assuming that the user looks at the top 10 retrieved documents (the ten items with the highest query-document similarity), better retrieval results are then obtainable with the extended matching than with conventional Boolean comparisons. The two items that do not match the original Boolean formulations do, however, exhibit rather low query-document similarities in the extended system.

In the vector-matching system illustrated on the right-hand side of Table V(c), the five relevant documents are retrieved even more effectively with ranks 1, 2, 4, 5, and 6, but the distinction between the items matching the Boolean query and those not matching the query has disappeared. In the vector-matching system, all retrieved items exhibit fairly low query-document similarities ranging from 0.384 for the highest retrieved document to 0.125 for the sixth retrieved item. Actual retrieval results are exhibited in the next section.

4. EXTENDED RETRIEVAL EXPERIMENTS

4.1 Experimental Collections and Evaluation Process

Four different document collections are used to evaluate the extended retrieval model covering items in biomedicine, library science and related areas, electrical engineering and related areas, and computing and related areas, respectively. The biomedical collection, designated as Medlars 1033, represents a subset of items obtained from the National Library of Medicine. The library science items, designated as ISI 1460, cover highly cited items extracted from the Social Science Citation Index and obtained with the help of the Institute for Scientific Information. The Inspec 12684 collection covers items in electrical engineering and computer science extracted from Computer and Control Abstracts and obtained with the help of the Institution of Electrical Engineers. Finally, the CACM 3204 collection covers articles published between 1959 and 1979 in the *Communications of the ACM*.

For each of these collections, queries were formulated first in natural language form, and later in Boolean form by graduate students and staff of Cornell University (for Medlars and ISI) and by graduate students of Syracuse University (for Inspec). Because the Medlars, ISI, and CACM collections are relatively small, consisting of 1033, 1460 and 3204 documents, respectively, it was possible to perform exhaustive relevance assessments of each document with respect to each query. The relatively large size of the Inspec collection (12684 documents) made it impractical to obtain complete relevance assessments. However, as part of a study dealing with different document representations, each query was processed in seven different ways at Syracuse University producing seven different sets of retrieved documents [24]. Two additional searches were conducted at Cornell including one vector-processing and one p-norm run using the extended retrieval model. All the documents retrieved by the nine different search methods were examined for relevance, and the assumption is that the vast majority of the relevant items in the complete collection were actually retrieved by one of the nine searches. The collection environment is summarized in Table VI.

To evaluate the effectiveness of a retrieval system, it is customary to compute values of the search recall and the 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

TABLE VI. Document Collections

Collection Name	Number of Documents	Number of Queries	Subject Area
Medlars	1033	30	Documents in biomedicine received from the National Library of Medicine. Boolean queries constructed by Cornell staff.
ISI	1460	35	Documents in library science and related areas extracted from *Social Science Citation Index* by the Institute for Scientific Information. Boolean queries constructed by Cornell staff.
Inspec	12684	77	Documents in electrical engineering and computing extracted from *Computer and Control Abstracts by the Institution of Electrical Engineers.* Queries constructed by graduate students at Syracuse University.
CACM	3204	52	Documents in computer science published between 1959 and 1979 in *Communications of the ACM.* Queries constructed by graduate students in computer science at Cornell University

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 and the extended *p*-norm retrieval system, the documents are retrieved in decreasing order of a given document-query similarity measure, that is, the most important

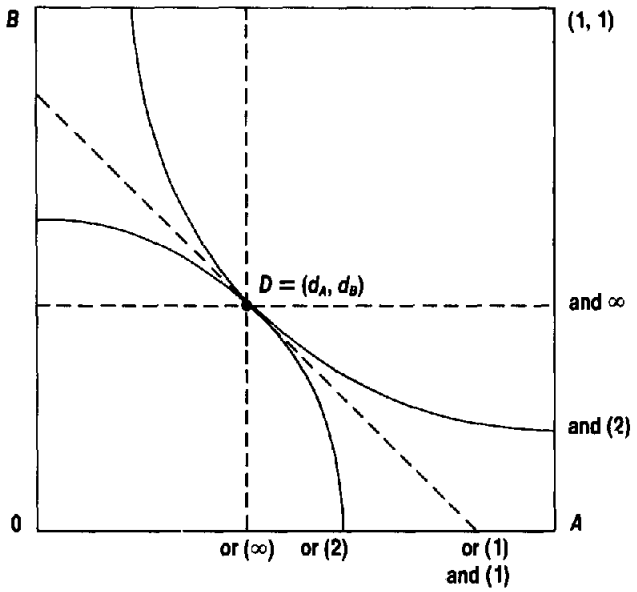

FIGURE 4. Query Document Similarities for $D = (d_A, d_B)$ when $d_A = d_B$ and (∞) = and (2) = and (1) = or (1) = or (2) = or (∞).

items are obtained first. The ranked output then makes it possible to compute a recall and a precision value after the retrieval of each item. 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 as shown in Figure 5 [3, 4]. The retrieval system exhibiting the highest recall-precision values, corresponding to the graph located closest to the upper right-hand corner where both recall and precision are equal to 1, represents the preferred method. For the two retrieval methods used in Figure 5, the *p*-norm run with *p*-values set equal to 1 and binary term weights proves much superior to the standard Boolean retrieval system.

Since it is awkward to compare complete tables or graphs with each other when many comparisons are performed, the

TABLE VIIa. Evaluation Results—Extended Retrieval Medlars 1033 Collection

Type of Run	*p* Value	Average Precision at Three Recall Points	Percent Difference from Boolean	Percent Difference from Cosine
Basic Boolean Retrieval binary weights	∞	0.2065	—	− 62%
Extended (*p*-norm) Retrieval				
Binary Weights for queries and documents	1	0.4710	+ 128%	− 14%
	2	0.4767	+ 131%	− 13%
	5	0.4720	+ 129%	− 14%
	9	0.4599	+ 123%	− 16%
Extended (*p*-norm) Retrieval				
Document Weights tf · idf binary query weights	1	0.5505	+ 167%	+ 1%
	2	0.5573	+ 170%	+ 2%
	5	0.5442	+ 163%	− 1%
	9	0.5417	+ 162%	− 1%
	∞	0.2368	+ 15%	− 57%
Vector Processing using cosine match and natural language query	—	0.5473	+ 165%	—
Optimal Case using retrospective term relevance weights	—	0.7175	+ 247%	+ 40%

TABLE VIIb. Evaluation Results—Extended Retrieval ISI 1460 Collection

Type of Run	p-Value	Average Precision at Three Recall Points	Percent Difference from Boolean	Percent Difference from Cosine
Basic Boolean Retrieval binary weights	∞	0.1118	—	− 29%
Extended (p-norm) Retrieval				
Binary Weights for queries and documents	1	0.1687	+ 51%	+ 8%
	2	0.1692	+ 51%	+ 8%
	5	0.1691	+ 51%	+ 8%
	9	0.1586	+ 42%	+ 1%
Extended (p-norm) Retrieval				
Document Weights tf · idf binary query weights	1	0.1835	+ 64%	+ 17%
	2	0.1806	+ 62%	+ 15%
	5	0.1710	+ 53%	+ 9%
	9	0.1674	+ 50%	+ 7%
	∞	0.1000	− 11%	− 36%
Vector Processing using cosine match and natural language query	—	0.1569	+ 40%	—
Optimal Case using retrospective term relevance weights	—	0.3798	+ 240%	+ 142%

TABLE VIIc. Evaluation Results—Extended Retrieval Inspec 12684 Collection

Type of Run.	p-Value	Average Precision at Three Recall Points	Percent Difference from Boolean	Percent Difference from Cosine
Basic Boolean Retrieval binary weights	∞	0.1159	—	− 50%
Extended (p-norm) Retrieval				
Binary Weights for queries and documents	1	0.1911	+ 65%	− 18%
	2	0.2069	+ 79%	− 11%
	5	0.2084	+ 80%	− 10%
	9	0.2027	+ 75%	− 13%
Extended (p-norm) Retrieval				
Document Weights tf · idf binary query weights	1	0.2747	+ 136%	+ 18%
	2	0.2700	+ 133%	+ 16%
	5	0.2691	+ 132%	+ 16%
	9	0.2641	+ 128%	+ 14%
	∞	0.1314	+ 13%	− 43%
Vector Processing using cosine match and natural language query	—	0.2325	+ 101%	—
Optimal Case using retrospective term relevance weights	—	0.3134	+ 170%	+ 35%

TABLE VIId. Evaluation Results—Extended Retrieval CACM 3204 Collection

Type of Run	p Value	Average Precision at Three Recall Points	Percent Difference from Boolean	Percent Difference from Cosine
Basic Boolean Retrieval binary weights	∞	0.1789	—	− 40.6%
Extended (p-norm) Retrieval				
Binary Weights for queries and documents	1	0.2604	+ 44.8%	− 14.0%
	2	0.2624	+ 45.9%	− 13.3%
	5	0.2624	+ 45.9%	− 13.3%
	9	0.2489	+ 38.4%	− 17.8%
Extended (p-norm) Retrieval				
Document Weights tf · idf binary query weights	1	0.3090	+ 72%	+ 2.1%
	2	0.3314	+ 84%	+ 9.5%
	5	0.3113	+ 73%	+ 2.8%
	9	0.3052	+ 70%	+ 0.8%
	∞	0.1551	− 14%	− 48.8%
Vector Processing using cosine match and natural language query	—	0.3027	+ 68%	—
Optimal Case using retrospective term relevance weights	—	0.4659	+159%	+ 54%

(a)

Recall Level	Average Precision for 30 queries	
	Base Case Standard Boolean	p-Norm Run Binary Weight $p = 1$
0.1	0.5606	0.7383
0.2	0.4364	0.6635
0.3	0.2917	0.5956
0.4	0.2387	0.5263
0.5	0.1871	0.4680
0.6	0.1562	0.4294
0.7	0.1073	0.3563
0.8	0.0774	0.2853
0.9	0.0385	0.1696
1.0	0.0324	0.1116
Average Improvement		+ 172%

(b)

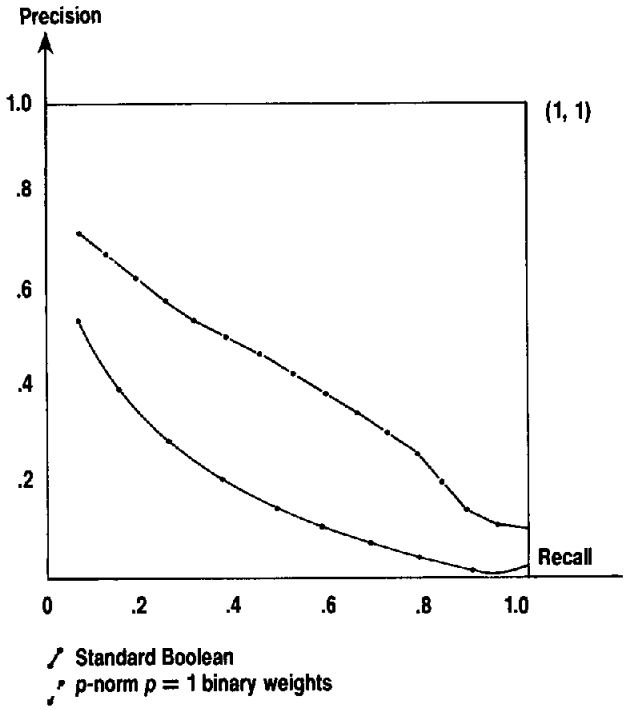

FIGURE 5. Typical Retrieval Results (Medlars 1033 collection; average over 30 queries. (*a*) Typical Recall-Precision Table; (*b*) Corresponding Recall-Precision Graph.

evaluation results included in the next section are stated in terms of a single precision value, representing the average precision 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 are then given for these composite precision measures.

4.2 Evaluation Results

The evaluation output for the four document collections is included in Tables VII(*a*), VII(*b*), VII(*c*), and VII(*d*), respectively. In each case, the base run listed at the top of each table is assumed to be the standard Boolean system (with binary term weights and p values = ∞). A secondary base run listed at the bottom of each table is the vector-processing run using a cosine coefficient to compare query and document vectors. The corresponding vector forms were derived from the original natural language query statements and from the document abstracts using a standard automatic-indexing process based on word-stem extraction and on an automatic assignment of term weights using the *tf·idf* weighting system [13–16]. A comparison of the results for the standard Boolean and the automatic vector-processing systems shows that the latter produces improved retrieval results for all three collections, ranging from a 40 percent improvement for ISI and CACM to 165 percent for Medlars. This agrees with earlier test results showing that properly chosen automatic-indexing methods with weighted term assignments will outperform the conventional Boolean retrieval methodologies [25, 26].

The results for the extended Boolean (p-norm) retrieval model are shown in the middle section of Tables VII(*a*)–VII(*d*). The p-norm query formulations corresponding to the search results of Table VII were constructed fully automatically from the original Boolean query formulations by assigning constant p values to all connectives in the original queries. Sample values of p = 1, 2, 5, and 9 were used experimentally.

Two main cases arise: the extended model can be used with unweighted (binary) query and document terms; alternatively, the query and/or document terms can be automatically weighted in accordance with the formulas of expressions (11), (12), or (13). The methods used for experimental purposes are the unweighted binary case listed in the upper halves of Tables VII(*a*)–VII(*d*), and a weighted case where the documents terms only are weighted using the *tf·idf* formula of expression (13) shown in the lower part of the tables.

The results of Tables VII(*a*)–VII(*d*) indicate that substantial improvements in retrieval effectiveness are obtainable in the extended system by simply replacing the normal strict interpretation of the Boolean connectives (corresponding to *p* values equal to infinity) by lower p-values. For the binary *p*-norm system the optimum values appear to occur for *p* values somewhere between 2 and 5; as the *p*-values grow larger, the amount of improvement over the standard Boolean base case decreases. The extended retrieval system used in a binary mode provides improvements over the Boolean system ranging from about 50 percent for ISI to about 130 percent for Medlars. When the unweighted p-norm system is compared with the vector-processing system, an additional advantage is obtained only for the ISI collection.

Consider now the weighted *p*-norm model. It may be seen that when the document terms are automatically weighted using the *tf·idf* function, additional improvements are produced for all collections over the unweighted case. The p values tried for the weighted system are identical with those used earlier for the binary case, except that an infinite p value is added. Once again, the infinite *p*, corresponding to a strict interpretation of the Boolean connectives, gives a much worse performance than any of the smaller *p*-values. For low *p*-values the weighted system affords improvements over the unweighted case ranging from ten percent for ISI to over 40 percent for Inspec. Compared with the pure Boolean basic

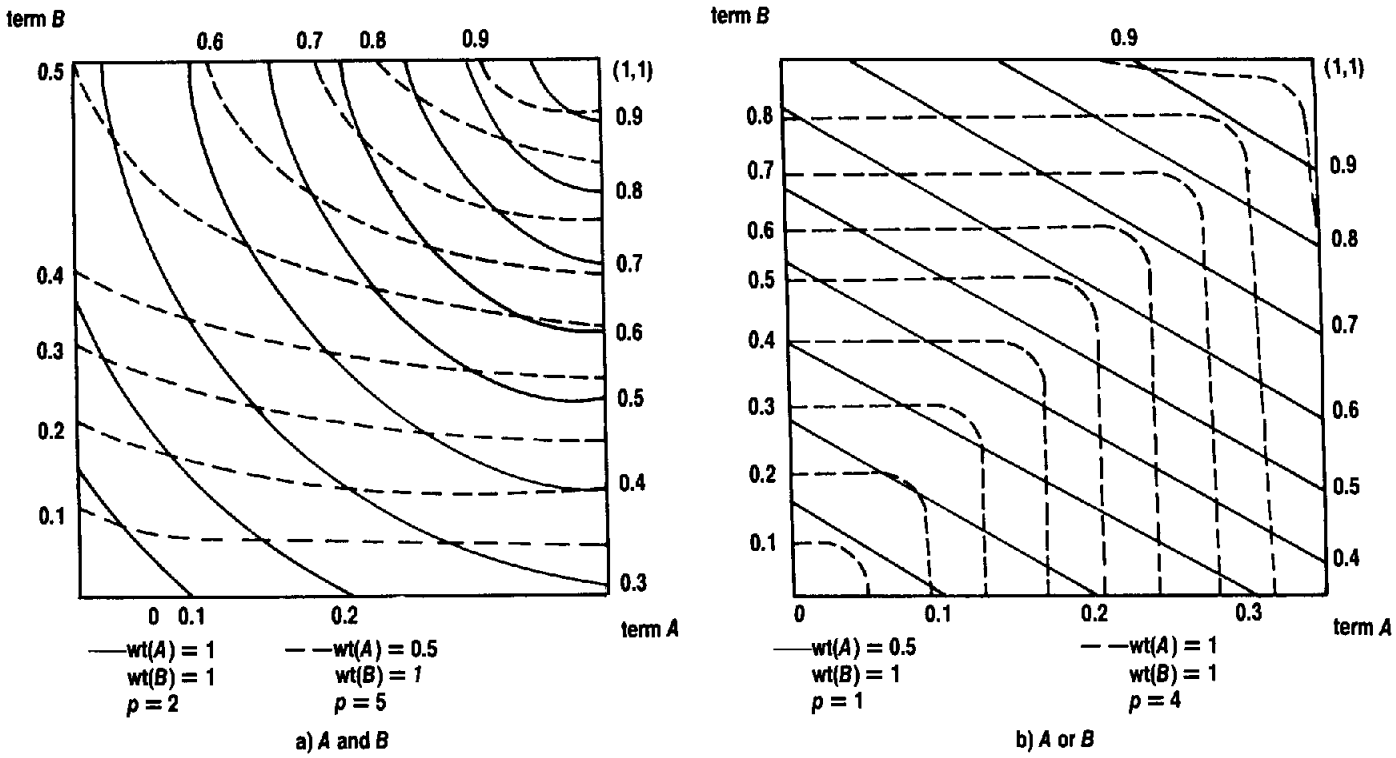

FIGURE 6. Interaction Between Term Weighting and *p*-Value Alterations

case, the weighted p-norm system brings improvements of at least 60 percent for ISI and up to 170 percent for Medlars. For the unweighted p-norm system, the best p-values were in the range from 2 to 5. For the weighted case, lower p-values between one and two appear most useful.

The difference in behavior between the weighted and unweighted cases may be explained by studying the effect of term weighting on the shape of the document space. Consider for this purpose the differences between the query-document spaces of Figures 1 and 6, respectively, covering the situations when the query terms are equally weighted as in Figure 1, or when one query term is more highly weighted than another as in Figure 6. In the latter case, the equal similarity lines, representing the points of equal distance from the (1,1) point for **and** and from the (0,0) point for **or**, become skewed, and complex interactions occur between the skewness due to the unequal weighting and the distortions of the distance computations due to the higher p-values. These distortions adversely affect the query-document similarity computations for higher p-values.

Each of the graphs of Figure 6 covers two queries representing [(*A*, 1) **and**[2] (*B*, 1)] and [(*A*, 0.5) **and**[5] (*B*, 1)] for Figure 6(*a*) and [(*A*, 0.5) **or**[1] (*B*, 1)] and [(*A*, 1) **or**[4] (*B*, 1)] for Figure 6(*b*), respectively. The solid lines of Figure 6(*a*) represent the lines of equal similarity (equal distance from the (1, 1) point) when both terms are equally weighted and p is equal to 2. The dashed lines of Figure 6(*a*) represent equal similarity lines for an example where term *B* is weighted twice as heavily as term *A* and the p-value is raised to 5. It may be seen that the similarity lines and the similarity values listed along the edges of the graph are very different for the two cases. For example, for a particular solid similarity line equal to 0.3, the dashed similarity values may vary from about 0.5 down to about 0.1 depending on the values of terms *A* and *B* in a document. This same effect is visible for the **or**-case in Figure 6(*b*) where $p = 4$ is used for the symmetric case of equal term weights (the dashed lines) and $p = 1$ covers the case where term *B* is weighted more heavily than term *A*.

In summary, when all query terms are equally weighted, the higher p-values, which impose a more definite structure on the query statements, are useful. On the other hand, when variable term weights appear, space distortions are caused as a result of the term weight differences and the p-norm distance measurements. In that case, lower values of p provide the best results. In either case, the pure Boolean approach, which is used conventionally in operating information retrieval situations, appears to be the worst possible choice.

The performance values actually attained by the p-norm system may be compared with the corresponding values for the optimal case, included at the bottom of Tables VII(*a*)–VII(*d*). This run represents an upper-bound experiment for queries using the exact vocabulary specified in the experimental query set 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 a ratio of the proportion of relevant documents in which the term occurs to the proportion of nonrelevant items in which the term occurs [19–21]. Given the actual terms included in the user queries and the terms contained in the abstracts, the use of term relevance weights produces the best possible performance obtainable in the particular collection environment.

5. CONCLUDING REMARKS

An extended query-document matching system is described in this study that relaxes the stringent requirements of the conventional Boolean retrieval operations. The new extended system furnishes a bridge between the strict interpretation of term connectives inherent in the use of Boolean logic and the total absence of query structure characterizing the vector-processing system. Experimental evidence obtained with various sample collections indicates that the extended matching system is more powerful than either the conventional Boolean system or the vector-processing system.

The adjustments needed in the extended retrieval system for document and query representations are simple to implement in practical retrieval situations. The document indexing is adjusted by attaching $tf \cdot idf$ weights to each document term [expression (13)]. The inverse document frequency values are easily computed by using the collection size (N) and the postings frequency (n) for each term (that is, the number of documents to which a term is attached). Each incoming Boolean query is then adjusted by replacing the implicit $p = \infty$ values attached to the connectives by lower p values. If no specific information is available from the users about the special importance of particular query terms, it appears simplest to choose uniformly low p-values—say, $p = 1$ or $p = 2$—for all Boolean operators. Thus, an initial query such as (A **and**(∞) B) **or**(∞) (C **and**(∞) D) becomes (A **and**(1) B) **or**(1) (C **and**(1) D). If the user provides specific information about the importance of individual query terms, then higher p-values (say, $p = 5$) may be used for the connectives attached to terms deemed to be most important.

It remains to specify how the extended p-norm queries may be compared with the document collection. Since most operational retrieval systems use inverted file organizations, it is possible to use a standard inverted file process to isolate a subset of potentially relevant documents in response to each query. This may be done by broadening each initial query and processing it in the standard inverted file mode against the complete collection. A query could be broadened, for example, by replacing all **and** by **or** connectives. The extended p-norm matching system can then be applied in a second query-document comparison operation by computing the p-norm distance between each document contained in the initially retrieved subcollection and the corresponding p-valued queries. The output of this second search operation produces ranked output that can then be used in a relevance feedback mode to construct improved query formulations usable in a subsequent search [17,18].

A complete iterative search procedure using the extended Boolean retrieval system, but compatible with the established conventional retrieval environments based on inverted file systems may then be specified in the following terms: [27]

1. Given a user search statement in natural language form, a broad *conventional* 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 initially available Boolean query formulation is now used to produce a p-valued formulation in the extended Boolean system. 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 and query 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 (5) and (6), 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. Such an improved query contains weighted terms included in the original query formulation as well as terms obtained from previously retrieved documents identified as relevant, together with appropriate p-valued Boolean operators [27].

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

The extended Boolean retrieval system can also be applied to retrieval environments where the initial query statements are available as natural language formulations of user needs rather than as conventional Boolean forms. In that case, completely automatic procedures can be used to produce extended Boolean formulations from the original natural language query statements [28]. In particular, the p-value interpretations previously outlined in Table III may then lead to a system where high-frequency terms that are normally included in term phrases are supplied with **and** connectives, and low-frequency terms that are normally used together with synonymous constructions are supplied with **or** connectives. A detailed consideration of automatic Boolean query formulation systems is beyond the scope of the present study. Experiments that apply the extended retrieval strategies to conventional operational Boolean retrieval systems based on inverted file organizations are currently being planned.

REFERENCES

1. Lancaster, F.W. and Fayen, E.G. *Information Retrieval On-Line*. Melville Publishing Co., Los Angeles, California, 1973.
2. Lancaster, F.W. *Information Retrieval Systems: Characteristics, Testing and Evaluation*. 2nd Ed., John Wiley and Sons, New York, 1979.
3. Salton, G. *Automatic Information Organization and Retrieval*. McGraw-Hill Book Co., New York, 1968.
4. Salton, G. Ed. *The Smart Retrieval System—Experiments in Automatic Document Processing*. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1971.
5. Noreault, T., Koll, M., and McGill, M.J. Automatic ranked output from Boolean searches in SIRE. *J ASIS. 28*, 6, (Nov. 1977), 333–339.
6. Bookstein, A. A comparison of two systems of weighted Boolean retrieval. *J. ASIS. 32*, 4, (July 1981), 275–279.
7. Bookstein, A. Fuzzy requests: An approach to weighted Boolean searches. *J. ASIS. 31*, 4, (July 1980), 240–247.
8. Waller, W.G., and Kraft, D.H. A mathematical model for a weighted Boolean retrieval system. *Information Processing and Management, 15*, 5, (1979), 235–245.
9. Buell, D.A., and Kraft, D.H. Threshold values and Boolean retrieval systems. *Information Processing and Management, 17*, 3, (1981), 127–136.

10. Ortega, J.M. *Numerical Analysis*. Academic Press, New York, 1977.
11. Bartle, R.G. *The Elements of Integration*. J. Wiley and Sons, New York, 1966.
12. Wu, H. On query formulation in information retrieval. Doctoral Dissertation, Cornell University, Ithaca, New York, January 1981.
13. Salton, G. A Theory of Indexing. Regional Conference Series in Applied Mathematics, No. 18, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, February 1975.
14. Salton, G. On the role of words and phrases in automatic text analysis. *Computers and the Humanities, 10*, 2, (March–April 1976), 69–87.
15. Salton, G., Wong, A., and Yu, C.T. Automatic indexing using term discrimination and term precision measurements. *Information Processing and Management. 12*, 1, (1976), 43–51.
16. Salton, G., Wu, H., and Yu, C.T. The measurement of term importance in automatic indexing. *J ASIS, 32*, 3, (May 1981), 175–186.
17. Rocchio, Jr., J.J. Relevance feedback in information retrieval. In *The Smart System—Experiments in Automatic Document Processing*. G. Salton, Ed. Prentice Hall, Englewood Cliffs, New Jersey, 1971.
18. Ide, E., and Salton, G. Interactive search strategies and dynamic file organization in information retrieval. In *The Smart System—Experiments in Automatic Document Processing*. G. Salton, Ed., Prentice Hall, Englewood Cliffs, New Jersey, 1971.
19. Robertson, S.E., and Sparck Jones, K. Relevance weighting of search terms. *J ASIS. 27*, 3, (May–June 1976), 129–146.
20. Yu, C.T., and Salton, G. Precision weighting—An effective automatic indexing method. *J ACM. 23*, 1, (January 1976), 76–88.
21. Sparck Jones, K. Experiments in relevance weighting of search terms. *Information Processing and Management. 15*, 3, (1979), 133–144.
22. Dillon, M., and Desper, J. The use of automatic relevance feedback in Boolean retrieval systems. *Journal of Documentation. 36*, 3, (Sept. 1980), 197–208.
23. Sparck Jones, K. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation. 28*, 1, (March 1972), 11–21.
24. Katzer, J., McGill, M.J., Tessier, J.A., Frakes, W., and DasGupta, P. A Study of the Overlap Among Document Representations. Technical Report, School of Information Studies, Syracuse University, 1982.
25. Salton, G. A comparison between manual and automatic indexing methods. *American Documentation. 20*, 1, (Jan. 1969), 61–71.
26. Salton, G. A new comparison between conventional indexing (Medlars) and automatic text processing (Smart). *J. ASIS. 23*, 2, (March–April 1972), 75–84.
27. Salton, G., Buckley, C., Fox, E.A., and Voorhees, E. Boolean Query Formulation with Relevance Feedback. Technical Report TR 83-539, Department of Computer Science, Cornell University, Ithaca, New York, January 1983.
28. Salton, G., Buckley, C., and Fox, E.A. Automatic Query Formulations in Information Retrieval. *Journal of the ASIS. 34*, 4, (July 1983), 262–280.

CR Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.4 [**Information Storage and Retrieval**]: Systems and Software
General Terms: Measurement
Additional Key Words and Phrases: information retrieval, query formulation, Lp-vector norm, generalized distance measurement, online retrieval methods

Received 8/82; revised 1/83; accepted 4/83