Extended Boolean Information Retrieval

G. Salton[†], E.Fox[†] and H. Wu[††]

August 1982

TR 82-511

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

[††]ITT - Programming Technology Center
1000 Oronogue Lane
Stratford, Connecticut 06497

Extended Boolean Information Retrieval

G. Salton[†], E.A. Fox[†], and H. Wu[††]

Abstract

In conventional information retrieval Boolean combinations of index terms are used to formulate the users' information requests. While any document is in principle retrievable by a Boolean query, the amount of output obtainable by Boolean processing is difficult to control, and the retrieved items are ..ot ranked in any presumed order of importance to the user population. In the vector processing model of retrieval, the retrieved items are easily ranked in decreasing order of the query-record similarity, but the queries themselves are unstructured and expressed as simple sets of weighted index terms.

A new, extended Boolean information retrieval system is introduced which 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 systems.

1. Conventional Retrieval Strategies

In conventional information retrieval, the stored records are normally identified by sets of keywords, 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 which lists for each allowable index term the corresponding set of document references indexed by that term. The Boolean retrieval system is designed to retrieve all stored records exhibiting the precise combination of keywords 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 of 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 in many cases 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 sad-
dled with various disadvantages:

a) 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 formu-
    lation, a great deal of output could be obtained, or alternatively
    no output might be retrieved at all.

b) The output obtained in response to a query is not ranked in any
    order of presumed importance to the user; thus each retrieved item
    is assumed to be as important as any other retrieved item.

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

d) Boolean query formulations may produce counter intuitive 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 con-
    taining 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 keywords 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 is first used 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:

$$\max\ (d_A(X), d_B(X))\ \text{for query (A or B)}$$
$$\min\ (d_A(X), d_B(X))\ \text{for query (A and B)}$$
$$1 - d_A(X)\ \text{for query (not A)}.$$

It is easy to see that 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 lack of discrimination among the retrieved output nearly to the same extent as the conventional Boolean retrieval system, since the rank of a retrieved item depends only on the lowest or highest weighted document term for "and" and "or" queries, respectively. In addition, it is difficult to extend the fuzzy set model to situations where term weights are attached also to query terms instead of only to the document terms. [6-9]

In the remainder of this study, an extended Boolean retrieval model is introduced which easily 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 ("anded" terms) and term synonyms ("ored" 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

A) 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 1(a).

When only two terms are under consideration, the term assignment can be represented by a two dimensional map as shown in Fig. 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 $1 - \sqrt{(1-d_A)^2 + (1-d_B)^2}$. 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 $0 \leq d_A, d_B \leq 1$, 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.

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

(1)

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

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 $0 \leq d_A, d_B \leq 1$. In that case, each document is represented by a point on the unit square as shown in the illustration of Fig. 1. The locus of equidistant points from (1,1) and (0,0) for "_and_" and "_or_" respectively is shown in Fig. 1 by appropriate lines on the graph. Three documents labelled X, Y, and Z are used as examples in Fig. 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 $0 \leq a, b \leq 1$ reflecting the importance of the individual terms A,B in the query. In that case

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

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

(2)

It is easy to see that 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

$$||\underline{d}||_p = ||d_1, d_2, \ldots, d_n||_p = (d_1^2 + d_2^P + \ldots + d_n^P)^{1/P} \qquad (3)$$

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

$$||d||_p \text{(normalized)} = (\frac{1}{n})^{1/P} \; ||d||_p = \left[\frac{(d_1^P + d_2^P + \ldots + d_n^P)]}{n} \right]^{1/P} \qquad (4)$$

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]

B) 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 "$\underline{or}$"-query, $Q_{\underline{or}}$, can now be written as

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

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 "$\underline{and}$"-query, $Q_{\underline{and}}$ is written as

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

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[*]

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

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

It is now easy to show 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 is equal to infinity 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 infinity, 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 easy to see that $\text{sim}(D, Q_{or}) = \text{sim}(D, Q_{and})$, because

----

[*]The query weights, a_i are relative rather than absolute weights used to represent the presumed importance of one query term relation 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.

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

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

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

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

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

Expression (7) represents the inner product between a document $D = (d_{A_1}, d_{A_2}, \ldots, d_{A_n})$ and query $Q = (\frac{a_1}{a_1 + a_2 + \ldots + a_2}, \ldots, \frac{a_n}{a_1, + \ldots +, a_n})$. 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

$$w_i = (\frac{a_i}{a_1 + a_2 + \ldots + a_n}).$$

When p = ∞, one obtains

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

$$= 1 - \frac{\max \{a_1(1-d_{A_1}), \ldots, a_n(1-d_{A_1})\}}{\max \{a_1, a_2, \ldots, a_n\}}$$

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

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

$$= min\ \{\ d_{A_1},d_{A_2},\ldots,d_{A_n}\ \}$$

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

$$= \frac{max\ \{a_1 d_{A_1},a_2 d_{A_2},\ldots,a_n d_{A_n}\}}{max\ \{a_1,a_2,\ldots,a_n\}}$$

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

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

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 $Q_{\underline{or}}$ and the document term of lowest weight for $Q_{\underline{and}}$. 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 infinity 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 = ∞). 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

"$\underline{or}$" connection becomes weaker, until that distinction disappears com-
pletely as p reaches a lower bound of 1. The effect of the p-value
variations is represented schematically in Fig. 2.

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

$$\text{sim} (D, Q_{\underline{and}(\infty)}) \leq \text{sim}(D, Q_{\underline{and}(p)}) \leq \text{sim}(D, Q_{\underline{and}(1)}) \tag{8}$$

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

The various query-document similarities of expression (8) are strictly
equal when all document term weights are equal, that is when
$d_{A_1} = d_{A_2} = ... = d_{A_n}$. An evaluation of expressions (5) and (6) for that
case shows that the query-document similarity values obtained for vari-
ous values of p all reduce to d_A, the common document term value.

When the document terms are not all equally weighted, a strict ine-
quality obtains for the query-document similarities of expression (8),
except of course between $\text{sim}(D, Q_{\underline{and}(1)})$ and $\text{sim} (D, Q_{\underline{or}(1)})$. 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 $\underline{and}(\infty)$ B), (A $\underline{and}(p)$ B),
(A $\underline{and}(1)$ B) = (A $\underline{or}(1)$ B), (A $\underline{or}(p)$ B), and (A $\underline{or}(\infty)$ B), respectively.
In that case one has

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

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

This is easily 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 Fig. 3 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) $\text{Sim} (D, A \text{ and}(\infty) B) = \min (d_A, d_B) = \overline{OV}$

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

iii) $\text{Sim} (D, A \text{ and}(1) B) = \text{average} (d_A, d_B) = \overline{OY}$

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

v) $\text{Sim} (D, A \text{ or}(\infty) B) = \max (d_A, d_B) = \overline{OZ}$.

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 reduce to $\overline{OD}$ as shown in Fig. 3(b).

C) 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 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 2.

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 2. It is clear that 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 $Q = (<\psi, 0.2> \underline{or}(2) <C, 0.1>)$ where in turn $\psi = (<A, 0.3> \underline{and}(2) <B, 0.4>)$. 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 which hold for conventional Boolean query statements must be modified by addition of appropriate scale factors in the extended system. The following equivalences are easy to prove in the extended system [12].

a)   quasi idempotency

$$(A,a_1) \ \underline{and}(p) \ (A,a_2) = A$$

$$(A_1,a_1) \ \underline{or}(p) \ (A,a_2) = A$$

b)   commutativity

$$(A_1,a_1) \ \underline{and}(p) \ (A_2,a_2) = (A_2,a_2) \ \underline{and}(p) \ (A_1,a_1)$$

$$(A_1,a_1) \ \underline{or}(p) \ (A_2,a_2) = (A_2,a_2) \ \underline{or}(p) \ (A_1,a_1)$$

c)   duality

$$\neg \ ((A_1,a_1) \ \underline{and}(p) \ (A_2,a_2)) = \neg \ (A_1,a_1) \ \underline{or}(p) \ \neg (A_2,a_2)$$

$$\neg \ ((A_1,a_1) \ \underline{or}(p) \ (A_2,a_2)) = \neg \ (A_1,a_1) \ \underline{and}(p) \ \neg (A_2,a_2)$$

d)   involution

$$(\neg\neg \ ((A_1,a_1) \ \underline{and}(p) \ (A_2,a_2)) = (A_1,a_1) \ \underline{and}(p) \ (A_2,a_2)$$

$$(\neg\neg \ ((A_1,a_1) \ \underline{or}(p) \ (A_2,a_2)) = (A_1,a_1) \ \underline{or}(p) \ (A_2,a_2)$$

e)   quasi-associativity

$$<(<A_1,a_1> \ \underline{or}(p) \ <A_2,a_2>),a> \ \underline{or}(p) \ <B,b>$$

$$= (<A_1,aa_1> \ \underline{or}(p) \ <A_2,aa_2>) \ \underline{or}(p) \ <B,b(a_1^p+a_2^p)^{1/p}>$$

$$<(<A_1,a_1> \ \underline{and}(p) \ <A_1,a_2>),a> \ \underline{and}(p) \ <B,b>$$

$$= (<A_1,aa_1> \ \underline{and}(p) \ <A_2,aa_2>) \ \underline{and}(p) \ <B,b(a_1^p+a_2^p)^{1/p}>$$

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

$$((A,a) \text{ and}(p) \ (B,b)) \text{ or}(p) \ (C,c)$$

$$= ((A,a) \text{ or}(p) \ (C,c)) \text{ and}(p) \ ((B,b,) \text{ or}(p) \ (C,c))$$

and $\hspace{9cm}$ (10)

$$((A,a) \text{ or}(p) \ (B,b)) \text{ and}(p) \ (C,c)$$

$$= ((A,a) \text{ and}(p) \ (C,c)) \text{ or}(p) \ ((B,b) \text{ and}(p) \ (C,c)).$$

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 quasi-synonyms 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) 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

A) Main Properties of Extended Retrieval

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

a)   Structured queries can be processed, including the usual "and", "or", and "not" operators.

b)   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.

c)   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 infinity to one, 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]:

a)   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.

b) Thesaurus classes are most useful as a means for broadening low frequency terms which 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 3. 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.

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 mixed metric matching thus embodies aspects of bibliographic as well as of data base retrieval. When "and"(∞) and "or"(or) operators are used, a standard data base 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 data base 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 increasingly 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]

B) Query and Document Processing

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

$$\frac{and}{or} \left\{ \frac{and}{or} (term\ A,\ term\ B,...),\ \frac{and}{or} (term\ C,\ term\ D,...) \right\}$$

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 4 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 4(b) to correspond with the standard interpretation of the Boolean query of Table 4(a)), and each term and clause receives a term weight of 1. The clause weights are underlined for easy identification. The formulation of Table 4(b) is precisely the same as that of Table 4(a).

The formulation of Table 4(c) corresponds to that used in Table 4(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

$$idf_k = \log \frac{N}{n_k} \qquad (11)$$

In the query formulation of Table 4(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 4(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]

    a)   a value based on the occurrence characteristics of the term in the whole collection, typically the inverse document frequency weight of expression (11);

    b)   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 <u>term frequency</u>, tf, of a term in a given document.

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

$$w_{ik} = idf_k \cdot tf_{ik}. \tag{12}$$

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

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

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.

C) Typical Retrieval Results

The retrieval results obtainable in the extended system are illustrated in Table 5 for the query previously used as an example in Table 4. Five documents, numbered 1, 11, 36, 47 and 51 are relevant to the query in the collection of 82 documents. The query term frequencies in the five relevant documents are shown in Table 5(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 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 5(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, respec-

tively. 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 retrieves the top 10 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 5(c), the five relevant documents are retrieved even more effec-
tively 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

A) Experimental Collections and Evaluation Process

Three different document collections are used to evaluate the
extended retrieval model covering items in biomedicine, library science
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, covers highly cited items extracted from the Social Science Citation Index and obtained with the help of the Institute for Scientific Information. Finally, 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.

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 and ISI collections are relatively small, consisting of 1033 and 1460 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 6.

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

ing 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 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 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 Fig. 4. [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 Fig. 4, 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 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.

B) Evaluation Results

The evaluation output for the three document collections is included in Tables 7(a), 7(b) and 7(c), 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 equal to infinity). 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 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 7(a), (b) and (c). The p-norm query formulations corresponding to the search results of Table 7 were

constructed fully automatically from the original Boolean query formula-
tions 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
half of the Tables 7, 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 7 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 per-
cent 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 func-
tion, 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 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 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 explanation for the difference in behavior may be found in studying the effect of the term weighting on the shape of the document space. When each term is equally weighted, the lines of equal query-document similarity appear symmetric with respect to the center of the space as previously illustrated for the two-dimensional example in Fig. 1. On the other hand, when one term is more highly weighted than another, the equal similarity lines 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.

Consider as an example, the graphs of Fig. 5. The example of Fig. 5(a) covers an "and"-query; the "or"-case is covered in Fig. 5(b). The solid lines of Fig. 5(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 Fig. 5(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 edge 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 Fig. 5(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 Table 7(a). This run represents an upper-bound experiment where the unrealistic assumption is made that the relevance or non-relevance 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. Summary

An extended query-document matching system is described in this study which 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 extended retrieval system should be simple to implement in practical retrieval environments. The document indexing is adjusted by attaching tf × 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 special information is available from the users about the special importance of partic-

ular query terms, it appears simplest to choose uniformly low p-values -- say p=1 or p=2 -- for all connectives. 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 are 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 to the subcollections retrieved by the initially broadened queries.

If natural language, rather than Boolean queries are initially available, the extended retrieval model can be used to formulate initial Boolean statements completely automatically. In that case the p-value interpretations previously outlined in Table 3 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 consideration of automatic Boolean query formulation systems is beyond the scope of the present study.

Experiments which apply the extended retrieval strategies to

conventional Boolean retrieval systems based on inverted file organizations are currently being planned.

References

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

[2] F.W. Lancaster, Information Retrieval Systems: Characteristics, Testing and Evaluation, 2nd Edition, John Wiley and Sons, New York, 1979.

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

[4] G. Salton, editor, The Smart Retrieval System--Experiments in Automatic Document Processing, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[5] T. Noreault, M. Koll and M.J. McGill, Automatic Ranked Output from Boolean Searches in SIRE, Journal of the ASIS, Vol. 28, No. 6, November 1977, p. 333-339.

[6] A. Bookstein, A Comparison of Two Systems of Weighted Boolean Retrieval, Journal of the ASIS, Vol. 32, No. 4, July 1981, p. 275-279.

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

[8] W.G. Waller and D.H. Kraft, A Mathematical Model for a Weighted Boolean Retrieval System, Information Processing and Management, Vol. 15, No. 5, 1979, p. 235-245.

[9] D.A. Buell and D.H. Kraft, Threshold Values and Boolean Retrieval Systems, Information Processing and Management, Vol. 17, No. 3, 1981, p. 127-136.

[10] J.M. Ortega, Numerical Analysis, Academic Press, New York, 1977.

[11] R.G. Bartle, The Elements of Integration, J. Wiley and Sons, New York, 1966.

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

[13] G. Salton, A Theory of Indexing, Regional Conference Series in Applied Mathematics, No. 18, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, February 1975.

[14] G. Salton, On the Role of Words and Phrases in Automatic Text Analysis, Computers and the Humanities, Vol. 10, No. 2, March-April 1976, p. 69-87.

[15] G. Salton, A. Wong, and C.T. Yu, Automatic Indexing Using Term Discrimination and Term Precision Measurements, Information Processing and Management, Vol. 12, 1976, p. 43-51.

[16] G. Salton, H. Wu and C.T. Yu, The Measurement of Term Importance in Automatic Indexing, Journal of the ASIS, Vol. 32, No. 3, May 1981, p. 175-186.

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

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

[19] S.E. Robertson and K. Sparck Jones, Relevance Weighting of Search Terms, Journal of the ASIS, Vol. 27, No. 3, 1976, p. 129-14f.

[20] C.T. Yu and G. Salton, Precision Weighting - An Effective Automatic Indexing Method, Journal of the ACM, Vol. 23, No. 1, 1976, p. 76-88.

[21] K. Sparck Jones, Experiments in Relevance Weighting of Search Terms, Information Processing and Management, Vol. 15, 1979, p. 133-144.

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

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

[24] J. Katzer, M.J. McGill, J.A. Tessier, W. Frakes and P. DasGupta, A Study of the Overlap Among Document Representations, Technical Report, School of Information Studies, Syracuse University, 1982.

[25] G. Salton, A Comparison Between Manual and Automatic Indexing Methods, American Documentation, Vol. 20, No. 1, January 1969, p. 61-71.

[26] G. Salton, A New Comparison Between Conventional Indexing (Medlars) and Automatic Text Processing (Smart), Journal of the ASIS, Vol. 23, No. 2, March-April 1972, p. 75-84.

	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

a) Conventional Boolean Retrieval

	A	B	(A or B)		(A and B)
Document D_1	1	1	1	|	1
Document D_2	1	0	$1/\sqrt{2}$	|	$1-1/\sqrt{2}$
Document D_3	0	1	$1/\sqrt{2}$	|	$1-1/\sqrt{2}$
Document D_4	0	0	0	|	0

b) Extended Retrieval

Query-Document Similarity Values for Conventional

and Extended Systems

Table 1

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

Sample Recursive Document Value Computation
Table 2

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.

p-Value Interpretation
Table 3

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>)
```

Typical Query Formulations
Table 4

b) <u>Query Term Frequencies in Relevant Documents</u>

(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

<u>Rank of Retrieved Relevant Documents</u>

Boolean form, p=∞ 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	.756	1	11	.384	1	36
1	2	36	.756	2	1	.374	2	11
1	3	1	.636	3	36	.186	4	1
			.281	7	47	.156	5	51
			.281	9	51	.125	6	47

Typical Query Operations
Table 5

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.

Document Collections

Table 6

Medlars 1033				
Type of Run	p-Value	Average Precision at three Recall Points	Percent Difference from Boolean	Percent Difference from Cosine
Basic Boolean Retrieval binary weights	∞	.2065	-	- 56%
Extended (p-norm) Retrieval Binary Weights for queries and documents	1	.4710	+ 128%	- 1%
	2	.4767	+ 131%	0
	5	.4720	+ 129%	0
	9	.4599	+ 123%	- 3%
Extended (p-norm) Retrieval Document Weights tf × idf binary query weights	1	.5505	+ 167%	+ 16%
	2	.5573	+ 170%	+ 18%
	5	.5442	+ 163%	+ 15%
	9	.5417	+ 162%	+ 14%
	∞	.2368	+ 15%	- 50%
Vector Processing using cosine match and natural language query	-	.5473	+ 165%	-
Optimal Case using retrospective term relevance weights	-	.7675	+ 272%	+ 62%

Evaluation Results - Extended Retrieval

Medlars 1033 Collection

Table 7(a)

Type of Run	p-Value	Average Precision at three Recall Points	Percent Difference from Boolean	Percent Difference from Cosine
Basic Boolean Retrieval binary weights	∞	.1118	-	- 29%
Extended (p-norm) Retrieval Binary Weights for queries and documents	1	.1687	+ 51%	+ 8%
	2	.1692	+ 51%	+ 8%
	5	.1691	+ 51%	+ 8%
	9	.1586	+ 42%	+ 1%
Extended (p-norm) Retrieval Document Weights tf × idf binary query weights	1	.1835	+ 64%	+ 17%
	2	.1806	+ 62%	+ 15%
	5	.1710	+ 53%	+ 9%
	9	.1674	+ 50%	+ 7%
	∞	.1000	- 11%	- 36%
Vector Processing using cosine match and natural language query	-	.1569	+ 40%	-

ISI 1460

Evaluation Results - Extended Retrieval

ISI 1460 Collection

Table 7(b)

| Inspec 12684				
Type of Run	p-Value	Average Precision at three Recall Points	Percent Difference from Boolean	Percent Difference from Cosine
Basic Boolean Retrieval binary weights	∞	.1159	-	- 50%
Extended (p-norm) Retrieval Binary Weights for queries and documents	1	.1911	+ 65%	- 18%
	2	.2069	+ 79%	- 11%
	5	.2084	+ 80%	- 10%
	9	.2027	+ 75%	- 13%
Extended (p-norm) Retrieval Document Weights tf × idf binary query weights	1	.2747	+ 136%	+ 18%
	2	.2700	+ 133%	+ 16%
	5	.2691	+ 132%	+ 16%
	9	.2641	+ 128%	+ 14%
	∞	.1314	+ 13%	- 43%
Vector Processing using cosine match and natural language query	-	.2325	+ 101%	-

Evaluation Results - Extended Retrieval

Inspec 12684 Collection

Table 7(c)
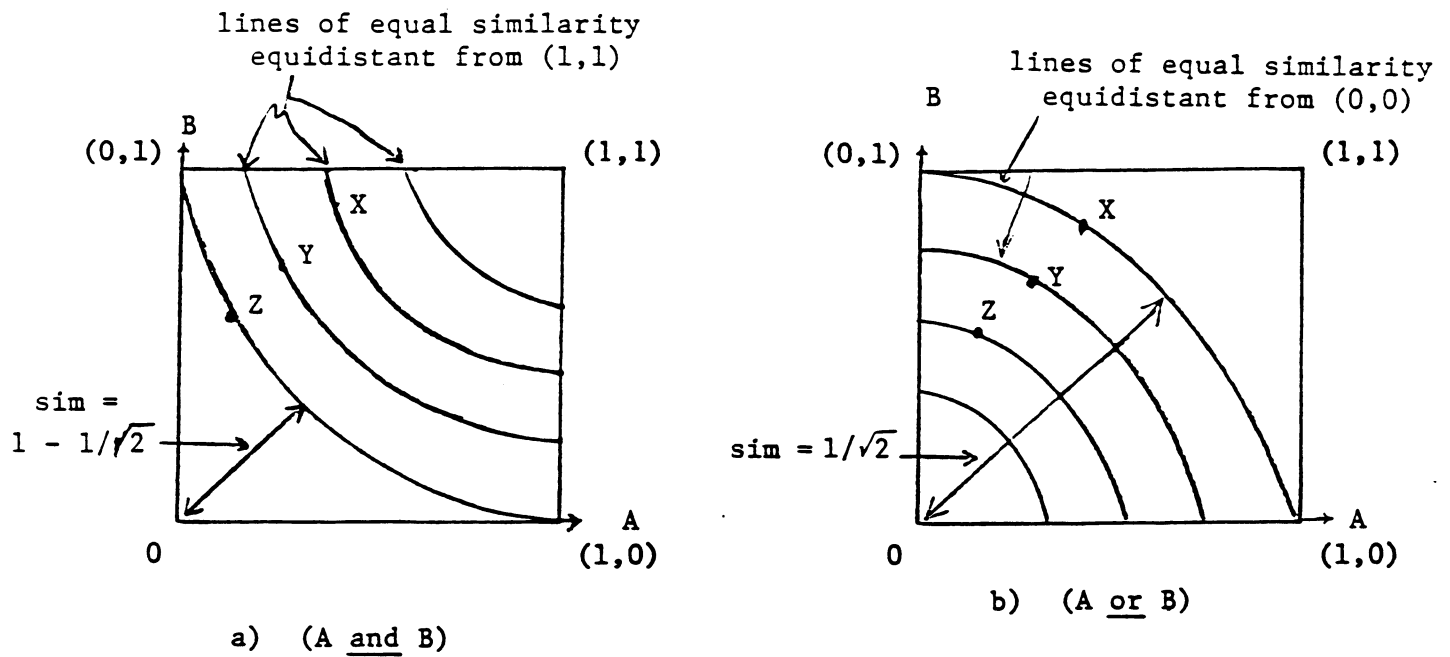
Equi-Distance Lines from (1,1) Point for "and"
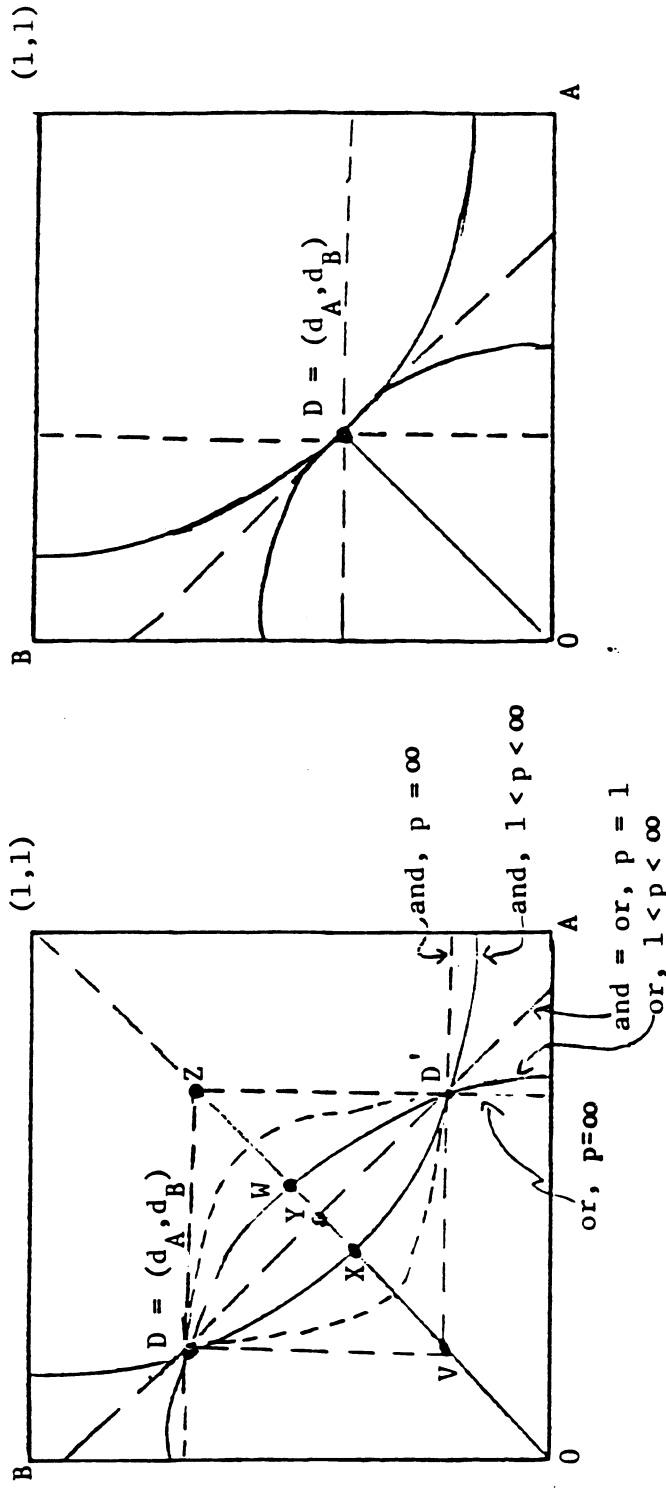and from (0,0) Point for "or"

Fig. 1
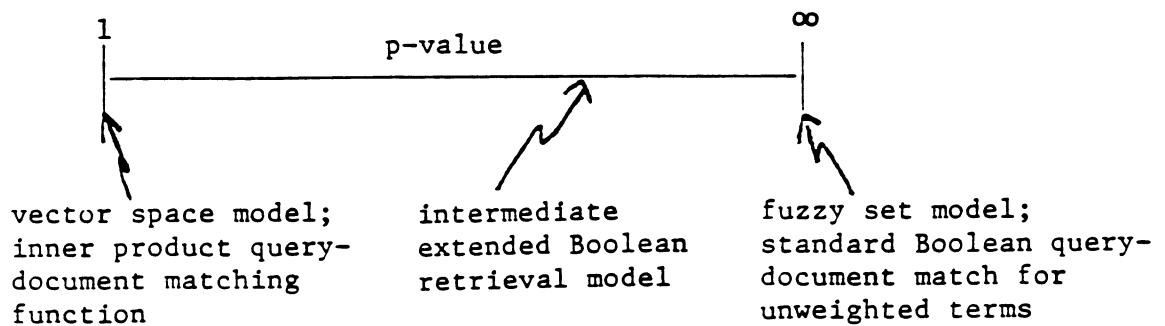
Range of p-Value Variations

Fig. 2

a) $d_A \neq d_B$

b) $d_{A.} = d_B$

Document-Query Similarities for $D = (d_A, d_B)$

and Queries (A and(p) B), (A or(p) B)

Fig. 3

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%

a) Typical Recall-Precision Table

/ Standard Boolean

/ P-Norm p=1 binary weights
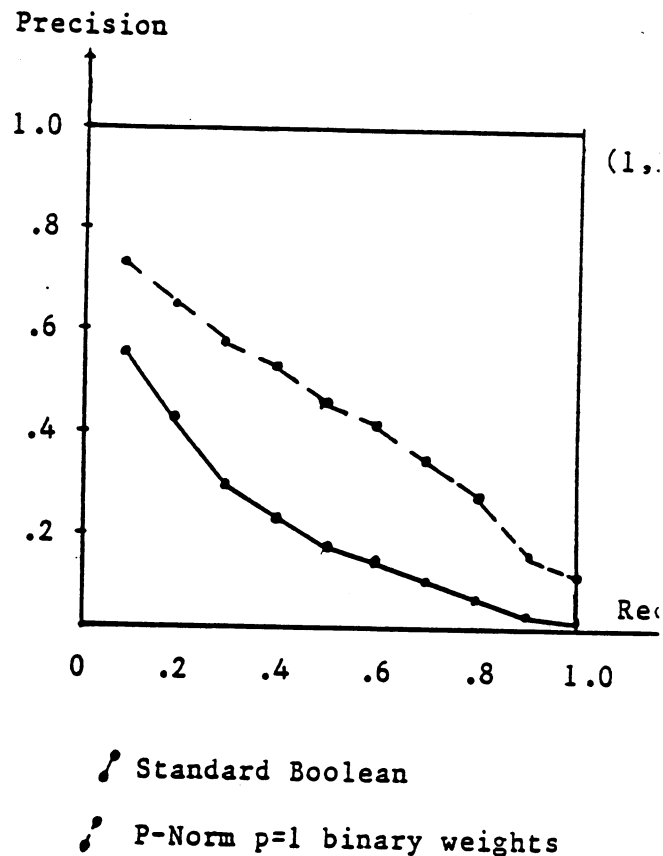
b) Corresponding Recall-Precision Gra

Typical Retrieval Results

(Medlars 1033 Collection;
average over 30 queries)

Fig. 4