Soft evaluation of Boolean search queries in information retrieval systems

C D Paice discusses the rationale for soft matching

Waller and Kraft[1] have presented a function permitting 'soft' matching between weighted Boolean search queries and weighted document representations. This paper discusses the rationale for soft matching more fully, generalizes Waller and Kraft's function to handle multiple arguments, and examines the rankings produced by soft, fuzzy and vector matching functions for two simple examples. The results, though tentative, suggest that discrepancies in ranking between equivalent Boolean query forms may be much less than the differences compared with non-soft matching functions. It is concluded, therefore, that further work in this area is fully justified.

Keywords: information retrieval, Boolean search queries, Waller and Kraft's function, soft matching

The commercially available information retrieval systems, almost without exception, rely on the use of Boolean logic for the formulation of search queries. Such systems appear to offer a clean and precise way of expressing a client's need. In practice, however, it is found that, for the retrieval of bibliographic references in particular, such systems leave a lot to be desired; the retrieval of some 'irrelevant' items, and the failure to retrieve some of the 'relevant' ones, are inescapable features of these systems. It is plain that the reason for this deficiency, or at least one important reason, is the categorical nature of the Boolean values *true* and *false*, which is in sharp conflict with the rather tentative status of the terms used both for representing the subject matter of a document, and for constructing a search query.

Department of Computer Studies, University of Lancaster, Bailrigg, Lancaster, UK
Received 29 July 1983

What is clearly required is a means of computing how closely a given document conforms to a given query. Assigning a value in the real interval [0, 1], such that the higher the value the greater the conformity, means that documents can be ranked in decreasing order of closeness to the query. In an interactive environment, details of documents can be displayed in rank order, so that the 'best' items should be presented first. In a batch system, the ranked output list can be terminated after a certain number of items, thus controlling the amount of output (it is of course a feature of Boolean systems that some queries will produce vast numbers of 'hits').

Systems producing ranked retrievals can be set up quite readily using queries comprising simple lists of search terms, known as query vectors. A variety of formulae are available for computing the similarity between a query vector and the 'document vectors' that are used to represent the documents in a collection. Moreover, it is easy to arrange for such systems to accept non-binary vectors, whose contents reflect the various statuses of the terms used in a query or assigned to a document. Feedback from users of such systems can be used to 'tune up' the values in query vectors, and possibly also in document vectors.

Despite these advantages, vector matching systems still produce results which are far from ideal (in the sense that rank positions and relevance judgements show rather poor correlation), and it is hard to resist the thought that the absence of Boolean relationships in the queries represents a serious lack. A few attempts have been made to produce a synthesis, but so far without any clear success.

BOOLEAN QUERIES AND POST HOC RANKING

It is helpful to distinguish two kinds of approach to producing ranked responses from Boolean queries. In the first approach, a Boolean query is first compared with each of a set of document representations in the normal way, to produce a set of unranked retrievals; the items in the retrieved set are then partially ordered using some convenient criterion. The alternative approach is to handle non-binary values right from the start; in other words, to replace the standard Boolean evaluations by some other form of logic, which will handle and produce values in the interval [0, 1].

A method of the first kind was described by Noreault *et al.*[2], who used a vector similarity measure (the 'cosine coefficient') for ranking the retrieved documents against the search terms. Their approach was subsequently criticized by Bookstein[3], who discussed a number of sources of logical inconsistency, mainly relating to the use of the *not* operator in queries. After describing some of the problems, Bookstein comments that 'there is no simple way to determine whether any given index term is negated in a Boolean statement'. This is not actually true, since it is quite easy to convert any Boolean expression into a tree structure, with the search terms appearing as the leaves, and the operators *and, or* and *not* as non-terminal nodes. Any leaf whose path from the root contains an odd number of *not* nodes represents a negated term. A problem certainly does remain if a particular term appears in both negated and non-negated positions within the same tree, as in the example (B *and* A) *or* (C *and not* A). Taking the easy way out, and ignoring any term of this kind, will indeed give inconsistent results, but it is perhaps unlikely that examples of this kind will occur very often in practice.

A more sophisticated approach of the same kind is outlined by Radecki[4]. This involves converting a query into its *disjunctive normal form,* consisting of a set of *reduced atomic index terms* (RAIT). Each RAIT contains all of the term in the query, and represents one possible way in which a document may satisfy the query. Thus, the query

(A *or* B) *and not* C

can be converted into a disjunctive normal form containing three RAITS:

(A *and* B *and not* C)
or (*not* A *and* B *and not* C)
or (A *and not* B *and not* C).

Every document which matches the query will match exactly one of the constituent RAITS. Each RAIT therefore represents a hypothetical document, for which an appropriate rank position may be determined. In the above example it seems appropriate that documents matching the first RAIT, which contain both terms A and B, should be ranked above documents containing only one of A and B. The relative ranking of the second and third RAITs, however, can only be decided by recourse to supplementary information Radecki proposes the use of probabilistic methods for determining the rank order of the hypothetical documents. In this case, estimated probability values would need to be assigned to all the terms allowed by the system: a substantial overhead compared with most existing IR systems.

Whilst an approach such as Radecki's may achieve a satisfactory level of consistency, one serious problem remains. Any Boolean query which is over-stringent — perhaps due to excessive use of the *and* operation — may fail to pass any documents at all to the ranking stage. In this case there is no alternative but to reformulate the query and try again. It is one of the attractive features of simple vector matching that (provided the query is a reasonable one) such failures cannot occur.

BOOLEAN QUERIES WITH INTEGRAL RANKING

In the second and more radical approach to producing ranked results from Boolean queries, non-binary logical values may be handled at any stage in the matching of a query to a document. The values concerned are usually restricted to the real interval [0, 1], where a value of 0 corresponds to the Boolean value *false,* and 1 corresponds to *true;* the higher the value within this interval, the nearer the value is to *true.*

Non-binary values may be attached either to the index terms in a document representation, or to the terms in a query, or to both. However, the meaning to be ascribed to these non-binary values is open to debate. It is often stated that such values indicate the relative *importance* of the various terms concerned[1, 5–7]: for it is clear that in any representation of a subject area, some concepts will be of crucial importance, whereas others merely provide supplementary information. Assignment of 'relative importance' values to *both* query terms and index terms should enable differences between the points of view of author and searcher to be handled (it being presumed that the indexer has constructed a faithful representation of the author's point of view).

It is obvious that another factor limiting the performance of IR systems relates to the more or less tentative nature of the choice of terms to denote particular concepts. This results from

- the existence and possible use of alternative terms for denoting a particular concept, which can result in loss of recall, and
- the possible use of a particular term with other meanings in other contexts, which can result in loss of precision.

Traditionally, thesauri have been used to control the assignment of index terms to documents, but little has been done to prevent users from incorporating unsuitable search clues into their queries. Another source of tentativeness appears if partial matching between search terms and index terms is possible: thus there is a strong but imperfect resemblance between the words 'conductance' and 'conductivity' and between the numbers 17.419 and 17.433.

It thus appears that a non-binary truth value in an IR system may be required to register *both* the importance of a term, relative to the subject areas as perceived by author and searcher, *and* the tentativeness of that term, relative to other terms in the document collection. It might therefore be argued that each term in a document should be assigned two non-binary values, to represent these two distinct features, the importance factor being derived from the assessments of the searcher and the indexer, and the tentativeness factor from the statistical properties of the vocabulary. An evaluative logic able to maintain the separation between these factors might be an interesting prospect, but the idea will not be pursued here.

A widely-canvassed approach to the handling of non-binary truth values in IR systems is that of *fuzzy set theory* and *fuzzy logic*[5, 7–11]. Each member of a fuzzy set has associated with it a value in the interval [0, 1] representing its *grade of membership* in that set; an object whose grade of membership is zero may be regarded as absent from the set, but all other objects are members of the set to some extent. Suppose that the grade of membership of a particular object in a fuzzy set A is g_A and in a fuzzy set B is g_B, then its grade of membership in $A \cup B$ is given by

$$g_{A \cup B} = \max (g_A, g_B)$$

and its grade of membership in $A \cap B$ by

$$g_{A \cap B} = \min (g_A, g_B).$$

Also, its grade of membership in the complement of A (denoted by $\bar{A}$) is

$$g_{\bar{A}} = 1 - g_A.$$

It will be noted that if the grades of membership are restricted to the values 0 and 1, these definitions reduce to those of standard Boolean logic.

It is clear that the above definitions are in some sense arbitrary. The possibility of alternative definitions was pointed out by Zadeh in his original paper, and has been repeated by others since[8, 12, 13]. However, there has been little inclination to pursue the alternatives, at least in the IR field. The appeal of the definitions above apparently derives from the fact that they preserve most of the usual axioms of set theory. More particularly, they appear to offer the only way of satisfying the criteria for acceptability enumerated by Bellman and Giertz[14] and by Bookstein[6]. Moreover, they provide a simple and clean method for evaluating 'Boolean' expressions containing non-binary truth values.

Radecki[7, 11] describes and illustrates the possibility of using fuzzy logic for matching 'Boolean' search queries against document representations. Due to the nature of the basic operations used, this approach can only produce ranked results if some at least of the initial values are non-binary; if the initial values are all 0 or 1, the results produced will always be those of standard Boolean logic. But Radecki expresses reservations about the possibility of assigning appropriate non-binary values to index terms:

> The main obstacle to implementing the information retrieval methods based on the theory of fuzzy sets is the fact that their effective use requires an index term weighting scheme which would need to be consistent with their underlying philosophy. Unfortunately, such an index term weighting methodology has not yet been developed, and it is not very likely that this will happen in the near future[4].

In the field of information retrieval, the possibilities of fuzzy logic have perhaps been seized on with more enthusiasm than reflection. Such workers as Bookstein[5, 6] and Radecki[7, 11] clearly take the view that fuzzy logic is an appropriate tool for handling relative term importance values. But the expositions by Zadeh[8], Bellman and Giertz[14], Gaines[12] and Tahani[10] present fuzzy logic in terms which appear to correspond much more closely with tentativeness in term assignment and in term–term matching (see above). Robertson[13] has strongly challenged the appropriateness of fuzzy logic for information retrieval purposes, and has suggested that research into applying probability theory would be more useful. In the present paper, the view is taken that it is proper to maintain a variety of approaches to IR research. Clearly, fuzzy logic has not provided an instant solution to the problem of IR system performance, and for this two interlinked reasons can be seen:

- the meaning of the values handled in the fuzzy logic approach is not clearly understood, and
- the choice of the initial values matters a great deal.

The purpose of this paper is to suggest that judicious generalization of standard fuzzy logic may permit the second of these problems to be eased.

We must accept that if fuzzy logic is to be generalized, the resulting system will be less 'clean' than what we have now. In particular, it may be necessary to re-examine the desiderata given by Bellman and Giertz[14] and by Bookstein[6]. In the final analysis, of course, our aim is a pragmatic one; we will be satisfied with any procedure which consistently succeeds in ranking a set of documents in descending order of their perceived relevance to a query. The advantages of a 'clean' model are that it is likely to be easier to formulate and to understand, and that inconsistencies and special cases may be avoided. In practice, however, the important thing is that any such pitfalls are known and can be allowed for.

THE ARGUMENT FOR SOFT LOGICAL OPERATORS

In what follows, the values handled during matching of a Boolean query will be referred to simply as *scores.* Values of 0 and 1, corresponding to the values of standard Boolean logic, will be referred to as *categorical* scores, whereas all values between these limits will be termed *non-categorical.* Effective ranking of retrieved items can only take place if non-categorical scores can be computed.

Values attached to the index terms of a document, and to the search terms of a query, will be referred to as *weights* (if weights are omitted, they may be assumed to have a default value, perhaps 1). The first stage in matching a query to a document is to compute an *initial score* for each term in the query; this is a function of the weight of the query term, of the weight of any corresponding index term that the document may contain, and of the degree of resemblance between the two matched terms. The design of a suitable function is at present a matter for debate (see Waller and Kraft). In the previous section it was implied that document weights and query weights played much the same rôle (or set of rôles), and accordingly, the examples discussed in the section on the evaluation of equivalent query forms use a simple formulation for initial score as the product of the corresponding document and query weights. Buell and Kraft[15], however, argue that it is better to regard query weights as thresholds, and this radically affects the forms of function which they propose. This matter is not, however, discussed further here, since the aim of this paper is to de-emphasize the importance of the initial scores in the overall evaluation process.

An important desideratum is that our method should determine a sensible rank order for documents irrespective of whether the initial scores are categorical or not. This means that the 'Boolean' operators *and* and *or* must be able to produce a non-categorical result even when both arguments are categorical. In order to understand why this is sensible, let us consider the following Boolean query:

FIND *A and B and C and D and E*

The standard view is that a record should be retrieved if and only if it contains a match for each of the search terms A to E. Irrespective of the user's view of the correctness of this search formulation, however, the vagaries of index-term assignment and of search-term selection are such that many relevant items might contain only three or four of the stated terms. Thus, many relevant items may be missed simply because the *and* operation, as normally understood, is too *categorical.* The query may therefore be taken to mean not that it is necessary for all the stated terms to be present in the record, but rather that as many as possible should be present. This widens the scope of retrieval, and provides a basis for partial ranking[16]. By an analogous argument, we can regard the disjunctive query

FIND P *or* Q *or* R *or* S *or* T

as meaning not just that any matched term on its own is sufficient for retrieval, but also that matches of two or more terms are even better. Taking this 'soft' view, the *and* operation indicates a strong desire that the arguments should occur together, whilst the *or* operation indicates that the desire for co-occurrence is weak.

The argument for using soft logical operators does not rest purely on uncertainties in the choice of index terms and search terms, but also on uncertainties regarding the choice of the Boolean operators themselves. The more involved the Boolean query, the greater the risk that the enquirer will put it together wrongly. Moreover, an enquirer who understands the uncertainties of term occurrence may find it impossible to construct the kind of query he or she wants except by making the query very long and repetitive. An answer to this last problem is to allow the construction of level-of-coordination queries ('retrieve all items which contain at least N index terms from the following list...'), but this prevents the inclusion of Boolean operators. The proper course, surely, is to allow enquirers to express their perceptions about desired patterns of term occurrence by means of Boolean operators, but to soften the way that those operators are treated.

A FORMULATION FOR SOFT LOGICAL OPERATORS

During the matching of a Boolean query to a document, each initial score may be taken as one indication of the appropriateness of that document for retrieval. The objective is to obtain an overall score which is a compromise between a number of (generally conflicting) indications. The Boolean operators govern the way in which the compromise score is to be computed. Presumably, if two indications are identical, the compromise must be the same as the original values, irrespective of what operator is specified. This leads to the rule that

$$(S_a \sim S_b) = S_a \,|\, S_a = S_b$$

where S_a and S_b are scores, and $\sim$ denotes any 'compromise operator'. More generally, it seems inescapable that $(S_a \sim S_b) \,|\, \; S_a \leq S_b$ must yield a result in the interval $[S_a, S_b]$, making it very clear that compromise scores are not probabilities. A simple way of producing such a result is to use a function of the kind suggested by Waller and Kraft[1]:

$$S_a \sim S_b = z{\cdot}\min(S_a, S_b) + (1 - z){\cdot}\max(S_a, S_b) \quad (1)$$

where z is a value in the interval $[0, 1]$. Specifically, if $z = 1$ then function (1) reduces to the fuzzy (or standard Boolean) *and* operation

$$S_a \cap S_b = \min(S_a, S_b),$$

whereas $z = 0$ yields the fuzzy (or standard Boolean) *or* operation

$$S_a \cup S_b = \max(S_a, S_b).$$

We see moreover that a 'soft *and*' operation (which we will denote by $\overset{\sim}{\cap}$) must have a coefficient $z_\cap$ somewhat less than 1, and a 'soft *or*' (denoted by $\overset{\sim}{\cup}$) must have a coefficient $z_\cup$ somewhat greater than 0. The proper values for $z_\cap$ and $z_\cup$ must be a matter for debate and experimentation, but the following (equivalent) conditions must necessarily hold:

$$z_\cap > z_\cup$$

and

$$(S_a \overset{\sim}{\cap} S_b) < (S_a \overset{\sim}{\cup} S_b) \,|\, S_a \neq S_b.$$

It is clear from the above discussion that the *and* and *or* operators need not be regarded as distinct species, but merely as two members chosen from a family of possible operators, differing in their relative degrees of 'and-ness' and 'or-ness'. At the half-way point, where $z = ½$, there exists a 'neutral' operation, corresponding to the arithmetic mean of the two operands. This suggests the introduction of two further rules, $z_\cap > ½$ and $z_\cup < ½$. However, whilst plausible, there appears to be no absolute reason why *and* and *or* in Boolean search queries must respect this dividing line.

When we consider a Boolean query containing several *and* and *or* operators, we have to admit that some of these operators might ideally need to be softer than others. There is clearly no way that an ordinary user can be expected to specify the proper degree of softness for each operator, and so the system must apply across-the-board z-values. However, there is no reason to suppose that the *and* and *or* operations should in practice be 'mirror images' of one another; in other words, the tempting condition $z_\cap + z_\cup = 1$ has no real justification. Suitable across-the-board values for $z_\cap$ and $z_\cup$ need to be determined by experimentation. We must not overlook the possibility, of course, that individual z-values might be adjusted in the light of relevance feedback from previous runs.

Turning now to the *not* operation, we have two questions to consider: should this operator be softened, and what formulation should be used? Regarding the first question, we may observe that the significance of attaching a *not* operator to a term in a query is somewhat ambiguous: it might mean that the absence of the term is a positive indication of relevance, or it might mean that the presence of the term is a negative indication. In fact, the same kind of ambiguity, but in reverse, applies to search terms which are not negated: is it the presence of a term which is most significant, or its absence? Of course, a term is likely to have significance in both senses, but there will generally be a bias in one direction or the other.

It appears that, in a Boolean search query, the 'sense of significance' of a term is implied by its environment: in short, by whether it enters into an *and* relationship with a neighbouring term or terms, or into an *or* relationship. In an '*and* environment' it is a negative indication (presence of a negated term, or absence of a non-negated one) which is most significant: in strict Boolean logic, one negative indication causes the environment to fail no matter how many positive indications the environment may contain. In an '*or* environment' a positive indication is most significant, since in Boolean logic a single positive indication guarantees the success of its environment. The realization that many search terms may carry some significance in the opposite sense to the dominant one provides a further argument for softening the logical operations in IR systems.

Returning to the question of the *not* operation, we may observe that in most actual queries *not* will not occur at the outermost level, but will appear within an environment controlled by *and* or *or.* We may therefore judge that any necessary softening will be done by the environmental operator. We may observe here that *not* operators usually appear in *and* environments, in agreement with the commonsense perception that negated terms are most often used as negative indicators. An 'outermost' *not* may be used for re-scanning a set of records that have already been retrieved from the main file, but this situation implies an *and* operation between the 'final' scores for the previous retrieved records and the scores for the negative supplementary query. Such a supplementary *and* does not, however, have the same effect as an 'integral *and*', since it is only applied to those records which have been retained from the previous search.

Standard fuzzy logic uses the simple function

$$g_{\bar{A}} = 1 - g_A$$

for complementation, and this is widely accepted (see Bellman and Giertz[14]). In the information retrieval context, however, evaluation of *not* in Boolean queries leads to various complications, and these are discussed by Waller and Kraft[1] and by Buell[17]. Nonetheless, the formula

$$S_{\bar{a}} = 1 - S_a \qquad (2)$$

(where S_a may be either an initial or an interim score) has been used for the simple example in the next section, and appears to give satisfactory results.

EXAMPLE OF THE SOFT PROCESSING OF A QUERY

By way of example, we will consider the query

FIND ((A *or* B) *and* (*not* C *and not* D)) *or* E.

We shall assume that neither index terms nor query terms are weighted, and that term matching is binary: hence the initial scores for the individual terms will all be 0 or 1. With respect to these five terms, therefore, a total of 2^5 (= 32) distinct cases can occur. Our soft operators will be governed by the coefficients $z_\cap = 2/3$, $z_\cup = 1/3$, giving the functions

$$S_a \tilde{\cap} S_b = 2/3 \min(S_a, S_b) + 1/3 \max(S_a, S_b), \qquad (3a)$$

$$S_a \tilde{\cup} S_b = 1/3 \min(S_a, S_b) + 2/3 \max(S_a, S_b), \qquad (3b)$$

Table 1 shows, in rank order, the socres obtained for the 32 distinct cases.

A careful examination of Table 1 shows that the rankings are generally in line with what might be expected. The table is divided into two parts by a horizontal line, the cases above the line being those which would have been retrieved by a strict Boolean treatment of the query. The final column in the table shows values obtained by using the cosine coefficient in the form

$$S_{\cos} = \frac{n_{qd}}{\sqrt{(n_q n_d)}}$$

where n_{qd} is the number of positive indications, $n_q = 5$ is the number of query terms, and n_d is the number of index terms, *assumed always equal to 10 for this example.* In the middle of the table, the cosine coefficient shows some striking disagreements with the ranking produced by our 'soft' approach, most particularly in relation to the two cases which are marked with an asterisk. According to Boolean logic, the presence of term E is sufficient to justify retrieval of a document, no matter how unfavourable the other indications may be: E is a powerful positive indicator, since it inhabits an *or*-environment at the outermost level. *A, B, not C* and *not D* are only weak indicators in the positive sense, since they are dependent on one another for support (though of course *A* and *B* are not mutually dependent). We observe also that the cosine coefficient gives the *'C,D,E,'* case a lower value than several patently inferior below-the-line cases. For these reasons we contend that a soft approach is superior to the use of the cosine coefficient, and probably also to any other measure which disregards the Boolean structure of a query.

The values in Table 2 illustrate the effects of choosing various combinations of values of $z_\cap$ and $z_\cup$. In column (1) the balanced situation $z_\cap + z_\cup = 1$ is retained, but the z-values are less 'soft' than those used in Table 1: the values are accordingly pushed towards the ends of the range. In column (2), $z_\cap + z_\cup < 1$, implying that *and* is softer than *or;* this has the effect that the scores are increased — that is, they are shifted towards the upper end of the scale. The values in column (3) illustrate what would seem to be a nonsense situation, where $z_\cup > 0.5$. However, although the scores are pushed strongly towards the bottom end of the scale, the rankings for the 15 cases are unchanged. Such examples as these suggest that a consistent ranking may be achieved so long as

$$0 < z_\cup < z_\cap < 1.$$

although this has not been formally proved.

Column (4) of Table 2 shows what happens when *and* and *or* become indistinguishable. None of the rankings obtained previously are reversed, but certain neighbouring scores degenerate to a common value. Most noteworthy, the scores immediately above and below the line are merged, indicating that no threshold can be found which will yield the same result as a strict Boolean search. Nonetheless, these results still seem intuitively better than those shown in Table 1 for the cosine coefficient. What has happened in setting $z_\cap = z_\cup = 0.5$ is that the Boolean operations *and* and *or* have been softened out of existence, but the differential weightings that were implicit in the original query have survived. The final *or* in the query has two operands which are treated equally: the first operand ((*A or B*) *and* (*not C and not D*)) is a

Table 1. Ranked scores for $((S_a \tilde{\cup} S_b) \tilde{\cap} (\neg S_c \tilde{\cap} \neg S_d)) \tilde{\cup} S_e$ with $z_\cup = \frac{1}{3}$, $z_\cap = \frac{2}{3}$ and categorical initial scores Cosine coefficient for comparison

	Positive indications		Negative indications			
Search terms found	Posited terms found	Negated terms avoided	Posited terms missing	Negated terms found	'Soft' scores	Cosine coeff.
A,B,E	A,B,E	C,D	. . .	. . .	1.000	0.707
A,E	A,E	C,D	B	. . .	0.926	0.566
B,E	B,E	C,D	A	. . .	0.926	0.566
A,B,C,E	A,B,E	D	. . .	C	0.852	0.566
A,B,D,E	A,B,E	C	. . .	D	0.852	0.566
A,C,E	A,E	D	B	C	0.815	0.424
A,D,E	A,E	C	B	D	0.815	0.424
B,C,E	B,E	D	A	C	0.815	0.424
B,D,E	B,E	C	A	D	0.815	0.424
A,B,C,D,E	A,B,E	. . .	. . .	C,D	0.778	0.424
E	E	C,D	A,B	. . .	0.778	0.424
A,C,D,E	A,E	. . .	B	C,D	0.741	0.283
B,C,D,E	B,E	. . .	A	C,D	0.741	0.283
C,E	E	D	A,B	C	0.704	0.283
D,E	E	C	A,B	D	0.704	0.283
A,B	A,B	C,D	E	. . .	0.667	0.566*
C,D,E	E	. . .	A,B	C,D	0.667	0.141*
A	A	C,D	B,E	. . .	0.519	0.424
B	B	C,D	A,E	. . .	0.519	0.424
A,B,C	A,B	D	E	C	0.370	0.424
A,B,D	A,B	C	E	D	0.370	0.424
A,C	A	D	B,E	C	0.296	0.283
A,D	A	C	B,E	D	0.296	0.283
B,C	B	D	A,E	C	0.296	0.283
B,D	B	C	A,E	D	0.296	0.283
A,B,C,D	A,B	. . .	E	C,D	0.222	0.283
none	. . .	C,D	A,B,E	. . .	0.222	0.283
A,C,D	A	. . .	B,E	C,D	0.148	0.141
B,C,D	B	. . .	A,E	C,D	0.148	0.141
C	. . .	D	A,B,E	C	0.074	0.141
D	. . .	C	A,B,E	D	0.074	0.141
C,D	. . .	. . .	A,B,E	C,D	0.000	0.000

Table 2. Ranked scores for $((S_a \tilde{\cup} S_b) \tilde{\cap} (\neg S_c \tilde{\cap} \neg S_d)) \tilde{\cup} S_e$ for various soft operators, with categorical initial scores. Weighted cosine coefficient for comparison

Cases	(1) $z_\cap = 0.8$ $z_\cup = 0.2$	(2) $z_\cap = 0.6$ $z_\cup = 0.2$	(3) $z_\cap = 0.9$ $z_\cup = 0.8$	(4) $z_\cap = 0.5$ $z_\cup = 0.5$	(5) Weighted cosine
A,B,E	1.000	1.000	1.000	1.000	0.894
A,E; B,E	0.968	0.976	0.424	0.875	0.783
A,B,C,E; A,B,D,E	0.872	0.928	0.352	0.875	0.783
A,C,E; A,D,E; B,C,E; B,D,E	0.864	0.912	0.288	0.750	0.671
A,B,C,D,E; E	0.840	0.880	0.280	0.750	0.671
A,C,D,E; B,C,D,E	0.832	0.864	0.216	0.625	0.559
C,E; D,E	0.808	0.832	0.208	0.625	0.559
A,B; C,D,E	0.800	0.800	0.200	0.500	0.447
A; B	0.672	0.704	0.056	0.375	0.335
A,B,C; A,B,D	0.288	0.512	0.038	0.375	0.335
A,C; A,D; B,C; B,D	0.256	0.448	0.022	0.250	0.224
A,B,C,D; none	0.160	0.320	0.020	0.250	0.224
A,C,D; B,C,D	0.128	0.256	0.004	0.125	0.112
C; D	0.032	0.128	0.002	0.125	0.112
C,D	0.000	0.000	0.000	0.000	0.000

function of four initial scores, each carrying just one quarter the weight of the single initial score (E) which alone forms the other operand.

The cosine coefficient (4) may be converted to a more general form

$$S_{\cos} = \frac{W_{qd}}{\sqrt{(W_q W_d)}} \tag{5}$$

where W_{qd} is the total weight of query term/index term matches, W_q is the total weight of the query terms, and W_d is the total weight of the index terms of the document in question. Assuming that the query terms are weighted 1:1:1:1:4 in order, and that $W_d = 10$ for every document, the formula becomes $S_{\cos} = W_{qd}/\sqrt{80}$, giving the values in column (5) of Table 2. It can be seen that these values are simple linear functions of those in column (4); in fact, the values in column (4) correspond to the weighted conformity measure W_{qd}/W_q.

GENERALIZATION TO MULTIPLE ARGUMENTS

It is easy to show that the formula used for the soft *and* and *or* operation does not obey the associative law. Thus, if we consider the query

FIND (J *and* K *and* L)

where $S_j = 1$, $S_k = 1$ and $S_l = 0$, application of formula (3a) gives 1/3 (= 0.3333), whereas the query

FIND (K *and* L *and* J)

evaluates to 5/9 (= 0.5556). This problem can easily be overcome if the diadic operators $\tilde{\cap}$ and $\tilde{\cup}$ are replaced by corresponding variadic functions $\xi_\cap$ and $\xi_\cup$. A ξ-function operates on a variable number of arguments held in a uniform environment. A uniform environment is one in which n arguments are connected by $n-1$ diadic operators, all of these operators being at the same softness level; the $n-1$ operators may then be represented by a single ξ-function, thus

$$(S_a \sim S_b \sim S_c \sim \ldots \sim S_p) = \xi(S_a, S_b, S_c, \ldots S_p).$$

Evaluation is achieved by first ordering the scores according to their (ascending) magnitude, to give a series of scores ($S_1, S_2, S_3, \ldots S_n$), and then assigning to the first score a weight equal to unity. Each subsequent score then receives a weight which is a fixed ratio r of the previous weight, leading to the formula

$$\xi_r(S_1, S_2, \ldots S_n) = \frac{\Sigma r^{i-1} S_i}{\Sigma r^{i-1}}, \tag{6}$$

where summation is from 1 to n. In the case where $n = 2$ and $r = 1/2$,

$$\xi_{½}(S_1, S_2) = \frac{S_1 + ½S_2}{1½} = \frac{2}{3}S_1 + \frac{1}{3}S_2$$

$$= \frac{2}{3}\min(S_1, S_2) + \frac{1}{3}\max(S_1, S_2)$$

which corresponds to the previous formulation of $\tilde{\cap}$ (Eq. 3a). Similarly, $n = 2$ and $r = 2$ give the previous form of $\tilde{\cup}$ (Eq. 3b). Hence, small values of r (generally, $r < 1$) represent soft *and* operations, whilst large values (generally, $r < 1$) represent soft *or* operations; $r = 1$, of course, represents a straight average of the scores.

In order that Eq. (6) shall give the results of standard fuzzy logic, it is necessary to set r equal to 0 for the *and* operation, and to ∞ for *or*. It is therefore simpler in practice to sort the scores into *descending* order for an *or* operation, so that $r = 0$ for fuzzy *or* as well as fuzzy *and*, and to handle $r = 0$ as a special case

$$\xi_r v(S_1, S_2, \ldots S_n) = \begin{cases} S_1 & \text{if } r = 0 \\ S_1 + \dfrac{\Sigma r^i S_i}{\Sigma r^i} & \text{otherwise,} \end{cases} \tag{7}$$

with summation from 2 to n.

THE EVALUATION OF EQUIVALENT QUERY FORMS

The example discussed in the section on soft logical operators concerned a rather simple query operating on categorical initial scores. If further examples are tried, it quickly emerges that the distributive law does not generally hold (as was appreciated by Waller and Kraft[1]). We might hope that two equivalent forms of a Boolean query, such as

FIND A *and* (B *or* C) (8a)

and

FIND (A *and* B) *or* (A *and* C) (8b)

would evaluate to the same result, but they do not.

In order to examine the significance of this problem, a file was generated containing 50 records each comprising three pseudo-random numbers in the interval [0, 1]. In each record d, the three values represented weights d_t for the terms A, B and C for a particular hypothetical document. The query terms were also assigned weights q_t, and the initial scores were obtained by multiplication, $S_t = d_t q_t$. The two Boolean queries above were applied to the file using Eq. (7) with $r = 0.25$ for both *and* and *or*, and the records were ranked according to the results from the conjunctive query (8a). Table 3 shows the results for the first seven, middle eight, and last seven items in the rank list. Discrepancies between the fourth and fifth columns affect not only the magnitudes of the final scores but, more seriously, the rank orders for the two forms of query.

The last column of Table 3 shows results obtained by applying the cosine coefficient in the form

$$\cos(d, q) = \frac{\Sigma d_t q_t}{\sqrt{\Sigma(d_t)^2 \ \Sigma(q_t)^2}} \tag{9}$$

The summations here cover all terms t in the index vocabulary, but the only terms which have any effect are those which occur (i.e., have an above-zero weight) in either the document, or the query, or both. This means that the term $\Sigma(d_t)^2$ is influenced by document terms which are not represented in the query. Assuming that there are no such terms causes (9) to give results which are very close to 1.0. This has been avoided by adding a plausible makeweight of 2.0 to the value of $\Sigma(d_t)^2$ for the

Table 3. Soft evaluation of equivalent query forms and of cosine coefficient. Contribution ratio : 0.250; query weights : 0.70 0.90 0.50; makeweight for C.C. : 2.000

A	B	C	A and (B or C)	(A and B) or (A and C)	Cosine co-efficient
0.95	0.97	0.71	0.6859	0.6487	0.7292
0.88	0.84	0.91	0.6320	0.6126	0.7070
0.81	0.93	0.05	0.5885	0.5235	0.6115
0.80	0.82	0.18	0.5697	0.5133	0.6096
0.96	0.73	0.09	0.5621	0.5621	0.5931
0.94	0.62	0.78	0.5511	0.5511	0.6552
0.72	0.88	0.80	0.5459	0.5334	0.6869
0.62	0.54	0.15	0.4098	0.3849	0.4865
0.65	0.39	0.80	0.4032	0.4032	0.5403
0.70	0.08	0.86	0.3847	0.3847	0.4429
0.85	0.39	0.30	0.3676	0.3676	0.5113
0.43	0.77	0.50	0.3617	0.3556	0.5742
0.55	0.42	0.44	0.3541	0.3541	0.4830
0.42	0.68	0.98	0.3527	0.3527	0.5910
0.93	0.31	0.22	0.3264	0.3264	0.4815
0.50	0.03	0.31	0.1735	0.1735	0.2789
0.18	0.00	0.61	0.1496	0.1345	0.2233
0.25	0.08	0.22	0.1169	0.1169	0.1971
0.05	0.57	0.06	0.1113	0.1107	0.3041
0.03	0.04	0.98	0.0966	0.0966	0.2552
0.07	0.28	0.37	0.0869	0.0869	0.2620
0.06	0.20	0.20	0.0664	0.0664	0.1792

query terms; 2.0 is the value which would result for a document having eight non-query terms each weighted 0.5. Generating random makeweights would of course be a still more arbitrary exercise, serving only to cloud the comparison with the soft Boolean evaluations. It can be seen that the differences in rank order between the cosine coefficient and the conjunctive query (8a) are much greater than the differences between the two forms of the Boolean query.

In all, six evaluation functions were applied to the file, and the rank orders were compared for each pair of functions. The functions were:

conj: the conjunctive query (8a), with $r = 0.25$;
disj: the disjunctive query (8b), with $r = 0.25$;
cos: the cosine coefficient, as described above;
conf: the conformity measure $\Sigma d_t q_t / \Sigma q_t$;
fuzz: the Boolean query with $r = 0$ (fuzzy logic);
ornd: the Boolean query with $r = 1$ (*and* and *or* not distinct).

The last two functions evaluate to the same results whichever Boolean form is used. For each comparison, the 'mean rank discrepancy' $\overline{\Delta}$ was computed using the formula

$$\overline{\Delta} = \frac{1}{N} \sum_{i=1}^{N} \left| \text{rank}\,(E, d, D) - \text{rank}\,(E', d, D) \right| \quad (10)$$

where N is the number of records in the file (50 here) and 'rank' is the rank position assigned to record d within file D by the evaluation function E or E'. The results shown in Table 4 suggest a partial ordering of the functions (i.e., fuzz, conj, desj, ornd, cos/conf), this apparently corresponding to the order of decreasing structural stringency in the queries. The fairly close agreement between cos and conf is not surprising, since these functions differ only in their denominators.

The most significant figure in Table 4 is the value $\overline{\Delta} = 1.12$ for soft evaluation of the two forms of the Boolean query. This value is less than half that for any other comparison except for cos:conf, suggesting that problems caused by the failure of the distributive law may be small compared with the differences between the soft approach and other methods of evaluation. It might well have been possible to reduce this figure still further by trying other values of r in Eq. (7), but in view of the artificial nature of the file used in these tests this did not seem a worthwhile exercise.

CONCLUSIONS

The file used in the above tests was entirely artificial, with a patently unrealistic distribution of document weights. Nonetheless the results, such as they are, suggest that soft evaluation of Boolean queries may produce rankings significantly different from other methods, and that discrepancies in ranking between equivalent Boolean query forms may be fairly small. We believe that these tentative conclusions fully justify further investigations.

In addition to designing and executing substantial experiments, various other lines of investigation are required. For example, the rationale of the soft matching approach needs to be considered further, particularly as regards the nature of the scores which are handled. The method does not conform to certain of the desiderata presented by Bellman and Giertz[14], Waller and Kraft[1] and Bookstein[6]. It is the impression of the present author that thinking about these desiderata tends to be influenced by more-or-less unconscious assumptions about the nature of the values being handled: in particular, perhaps, by a tendency to regard them as probabilities of some kind. Perhaps the presentation of the soft approach as a kind of 'compromise logic' may help to clarify matters. At all events, the desiderata certainly need close re-examination.

Consideration needs to be given also to the handling of negated terms in queries, and to the method for assigning weights and computing initial scores. It could be argued, for example, that since the choice of index terms and search terms is always to some extent tentative, it is never appropriate to assign categorical weights to these terms:

Table 4. Mean rank discrepancies $\overline{\Delta}$ for various evaluation functions (For explanation see text)

	fuzz	conj	disj	ornd	conf
cos	5.80	4.12	3.88	3.44	1.32
conf	5.88	4.18	3.88	2.88	
ornd	5.08	3.56	2.88		
disj	2.46	1.12			
conj	2.26				

there should in effect be a 'floor' and a 'ceiling' on the permitted weights.

A further task is to see whether the evaluation functions discussed above can be bettered. Ideally, we would like a method which always produced the same values for equivalent query forms. If this is unachievable, then we would happily settle for something which always produced identical rankings for equivalent queries. Failing this, we could at least hope to discover a method to reduce the rank discrepancies produced by the present functions.

An alternative approach to the equivalent form problem is to convert every query into a canonical representation before matching; the disjunctive and conjunctive normal forms have been suggested by Waller and Kraft[1]. Whilst this may achieve consistency, however, it serves no real practical purpose unless it can be shown that one form will result in consistently better retrieval performance than another.

Assuming that further experimentation confirms our optimism about the use of soft logic, then thought will have to be given to the question of algorithmic efficiency. The full evaluation of a soft score for every record in a large file will lead to unacceptably long search times. An adaptation of Radecki's 'λ-level' approach might permit many records to be discarded quickly[11], particularly if some form of inverted file could be organized. We may also ask whether the soft approach could be adapted for searching in clustered files, and whether it might be feasible and advantageous to adjust the individual softness levels in a query by feedback from users.

Finally, it is important to consider whether soft matching algorithms could be incorporated into existing information retrieval systems, where typically the initial scores are all categorical. If this could be done, it would become possible to assess whether the change actually improved the service to users.

REFERENCES

1 **Waller, W G and Kraft, D H** 'A mathematical model of a weighted Boolean retrieval system' *Information Processing and Management* Vol 15 (1979) pp 235–245

2 **Noreault, T, Koll, M and McGill, M J** 'Automatic ranked output from Boolean searches in SIRE' *J. Am. Soc. Inf. Sci. (USA)* Vol 28 (1977) pp 333–341

3 **Bookstein, A** 'On the perils of merging Boolean and weighted retrieval systems' *J. Am. Soc. Inf. Sci. (USA)* Vol 29 (1978) pp 156–158

4 **Radecki, T** 'A probabilistic approach to information retrieval in systems with Boolean search request formulations' *J. Am. Soc. Inf. Sci. (USA)* Vol 33 (1982) pp 365–370

5 **Bookstein, A** 'Fuzzy requests: an approach to weighted Boolean searches' *J. Am. Soc. Inf. Sci. (USA)* Vol 31 (1980) pp 240–247

6 **Bookstein, A** 'A comparison of two weighting schemes for Boolean retrieval' *Information Retrieval Research* (R N Oddy, S E Robertson, C J van Rijsbergen and P W Williams, eds.) (1981) London: Butterworths

7 **Radecki, T** 'Outline of a fuzzy logic approach to information retrieval' *Int. J. Man–Mach. Stud. (GB)* Vol 14 (1981) pp 169–178

8 **Zadeh, L** 'Fuzzy sets' *Inf. & Control (USA)* Vol 8 (1965) pp 338–353

9 **Negoita, C V and Flondor, P** 'On fuzziness in information retrieval' *Int. J. Man–Mach. Stud. (GB)* Vol 8 (1976) pp 711–716

10 **Tahani, V** 'A fuzzy model of document retrieval systems' *Information Processing and Management* Vol 12 (1976) pp 177–188

11 **Radecki, T** 'Fuzzy set theoretical approach to information retrieval' *Information Processing and Management* Vol 15 (1979) pp 247–260

12 **Gaines, B R** 'Foundations of fuzzy reasoning' *Int. J. Man–Mach Stud. (GB)* Vol 8 (1976) pp 623–668

13 **Robertson, S E** 'On the nature of fuzz: a diatribe' *J. Am. Soc. Inf. Sci. (USA)* Vol 29 (1978) pp 304–307

14 **Bellman, R and Giertz, M** 'On the analytic formalism of the theory of fuzzy sets' *Inf. Sciences (USA)* Vol 5 (1973) pp 149–156

15 **Buell, D A and Kraft, D H** 'Threshold values and Boolean retreival systems' *Information Processing and Management* Vol 17 (1981) pp 127–136

16 **Buell, D A** 'A general model of query processing in information retrieval systems' *Information Processing and Management* Vol 17 (1981) pp 249–262

17 **Buell, D A** 'An analysis of some fuzzy subset applications to information retrieval systems' *Fuzzy Sets and Systems* Vol 7 (1982) pp 35–42