Introduction to Modern Information Retrieval

Gerard Salton
Professor of Computer Science
Cornell University

Michael J. McGill
Associate Professor of Information Studies
Syracuse University

McGraw-Hill Book Company

New York St. Louis San Francisco Auckland Bogotá Hamburg
Johannesburg London Madrid Mexico Montreal New Delhi
Panama Paris São Paulo Singapore Sydney Tokyo Toronto

*2 VECTOR SIMILARITY FUNCTIONS

Consider a collection of objects in which each object is characterized by one or more properties associated with the objects. In information retrieval, the objects might be documents and the properties could be the index terms assigned to the documents. Alternatively, the objects could be index terms, and the properties could be the document identifiers to which the terms are assigned. Each property attached to a given object could be weighted to reflect its importance in the representation of the given object. Alternatively, a property characterizing an item may be considered to carry a weight of 1 when it is actually assigned to an item, or a weight of 0 when the property is not assigned. In the former case one speaks of *weighted* indexing; in the latter case the indexing is *binary*.

The similarity between two objects is normally computed as a function of the number of properties that are assigned to both objects; in addition the number of properties that are jointly absent from both objects may be taken into account. Furthermore, when weighted indexing is used, the weight of the properties appearing in the two vectors may be used instead of only the number of properties.

Consider as an example, two particular objects, say DOC_i and DOC_j, and let $TERM_{ik}$ represent the weight of property (term) k assigned to document i. In binary systems the value of $TERM_{ik}$ is restricted to either 0 or 1. Otherwise, one may assume that the weights vary from some lower limit such as 0 to some predetermined maximum weight for a given collection environment. If t properties are used to characterize the objects, the following property vectors may be defined for two sample objects:

$$DOC_i = (TERM_{i1}, TERM_{i2}, \ldots, TERM_{it})$$
$$DOC_j = (TERM_{j1}, TERM_{j2}, \ldots, TERM_{jt})$$

To compute the similarity between two given vectors, the following vector functions are of principal importance:

1 $\sum_{k=1}^{t} TERM_{ik}$, that is, the sum of the weights of all properties included in a given vector (in this case, the vector for DOC_i).

2 $\sum_{k=1}^{t} TERM_{ik} \cdot TERM_{jk}$, that is, the component-by-component vector product, consisting of the sum of the products of corresponding term weights for two vectors. For binary vectors this reduces to the number of *matching* properties for two vectors (the number of properties with weight equal to 1 in the two vectors).

3 $\sum_{k=1}^{t} \min(TERM_{ik}, TERM_{jk})$, that is, the sum of the minimum component weights of the components of the two vectors.

4 $\sqrt{\sum_{k=1}^{t} (\text{TERM}_{ik})^2}$, that is, the length of the property vector (in this case, the one for DOC_i) when the property vectors are consider as ordinary vectors.

Consider the following two vectors defined for a system using eight properties:

$$\text{DOC}_i = (3,2,1,0,0,0,1,1)$$
$$\text{DOC}_j = (1,1,1,0,0,1,0,0)$$

The four vector functions introduced earlier are then equal, respectively, to

1 $\sum_{k=1}^{t} \text{TERM}_{ik} = (3 + 2 + 1 + 0 + 0 + 0 + 1 + 1) = 8$

2 $\sum_{k=1}^{t} \text{TERM}_{ik} \cdot \text{TERM}_{jk} = [(3 \cdot 1) + (2 \cdot 1) + (1 \cdot 1) + (0 \cdot 0) + (0 \cdot 0) + (0 \cdot 1) + (1 \cdot 0) + (1 \cdot 0)] = 6$

3 $\sum_{k=1}^{t} \min(\text{TERM}_{ik},\text{TERM}_{jk}) = (\min(3,1) + \min(2,1) + \min(1,1) + \min(0,0) + \min(0,0) + \min(0,1) + \min(1,0) + \min(1,0)$
$= (1 + 1 + 1 + 0 + 0 + 0 + 0 + 0) = 3$

4 $\sqrt{\sum_{k=1}^{t} (\text{TERM}_{ik})^2}$

$$= \sqrt{(3 \cdot 3) + (2 \cdot 2) + (1 \cdot 1) + (0 \cdot 0) + (0 \cdot 0) + (0 \cdot 0) + (1 \cdot 1) + (1 \cdot 1)}$$
$$= \sqrt{9 + 4 + 1 + 0 + 0 + 0 + 1 + 1} = 4$$

The expressions under 2 and 3 are based on the manipulation of a particular vector pair; expressions 1 and 4 are single vector functions only. Hence the ordinary vector product (expression 2) and the sum of the minimum components (expression 3) could be used directly to measure the similarity between the vectors. In practice, it is customary to include normalizing factors when computing vector similarities. These factors ensure that the similarity coefficients remain within certain bounds, say between 0 and 1 or between -1 and $+1$. The following similarity measures are all relatively easy to generate and have been used in operational or experimental situations to compute term or document similarities [1,2]:

$$\text{SIM}_1(\text{DOC}_i,\text{DOC}_j) = \frac{2\left[\sum_{k=1}^{t} (\text{TERM}_{ik} \cdot \text{TERM}_{jk})\right]}{\sum_{k=1}^{t} \text{TERM}_{ik} + \sum_{k=1}^{t} \text{TERM}_{jk}} \quad (1)$$

$$\mathrm{SIM}_2(\mathrm{DOC}_i,\mathrm{DOC}_j) = \frac{\sum_{k=1}^{t}(\mathrm{TERM}_{ik}\cdot\mathrm{TERM}_{jk})}{\sum_{k=1}^{t}\mathrm{TERM}_{ik} + \sum_{k=1}^{t}\mathrm{TERM}_{jk} - \sum_{k=1}^{t}(\mathrm{TERM}_{ik}\cdot\mathrm{TERM}_{jk})} \tag{2}$$

$$\mathrm{SIM}_3(\mathrm{DOC}_i,\mathrm{DOC}_j) = \frac{\sum_{k=1}^{t}(\mathrm{TERM}_{ik}\cdot\mathrm{TERM}_{jk})}{\sqrt{\sum_{k=1}^{t}(\mathrm{TERM}_{ik})^2\cdot\sum_{k=1}^{t}(\mathrm{TERM}_{jk})^2}} \tag{3}$$

$$\mathrm{SIM}_4(\mathrm{DOC}_i,\mathrm{DOC}_j) = \frac{\sum_{k-1}^{t}(\mathrm{TERM}_{ik}\cdot\mathrm{TERM}_{jk})}{\min\left(\sum_{k=1}^{t}\mathrm{TERM}_{ik},\ \sum_{k=1}^{t}\mathrm{TERM}_{jk}\right)} \tag{4}$$

$$\mathrm{SIM}_5(\mathrm{DOC}_i,\mathrm{DOC}_j) = \frac{\sum_{k=1}^{t}\min(\mathrm{TERM}_{ik},\mathrm{TERM}_{jk})}{\sum_{k=1}^{t}\mathrm{TERM}_{ik}} \tag{5}$$

For the two sample vectors previously used as an illustration, the similarity functions SIM_1 to SIM_5 produce the following results:

$$\mathrm{SIM}_1(\mathrm{DOC}_i,\mathrm{DOC}_j) = \frac{2\cdot(6)}{8+4} = 1$$

$$\mathrm{SIM}_2(\mathrm{DOC}_i,\mathrm{DOC}_j) = \frac{6}{8+4-6} = 1$$

$$\mathrm{SIM}_3(\mathrm{DOC}_i,\mathrm{DOC}_j) = \frac{6}{\sqrt{16\cdot 4}} = \frac{6}{\sqrt{64}} = \frac{6}{8} = 0.75$$

$$\mathrm{SIM}_4(\mathrm{DOC}_i,\mathrm{DOC}_j) = \frac{6}{4} = 1.5$$

$$\mathrm{SIM}_5(\mathrm{DOC}_i,\mathrm{DOC}_j) = \frac{3}{8} = 0.375$$

The first two coefficients, SIM_1 and SIM_2, are known respectively as the *Dice* and *Jaccard* coefficients. They are widely used in the literature to measure vector similarities. The third coefficient, SIM_3, is the *cosine* coefficient that was introduced earlier in this volume. The cosine is a measure of the angle between two t-dimensional object vectors when the vectors are considered as ordinary vectors in a space of t dimensions. Since the numerator in the cosine expression must be divided by the product of the lengths of the two vectors, long vectors with many terms and hence great length normally produce small cosine similarities. The *overlap* measure, SIM_4, does not have this property because its denominator consists of the lower-weighted terms from the two vectors. In a retrieval environment, the query usually contains low-weighted terms; hence the query-document correlations using the overlap measure SIM_4 will be larger in magnitude than those of the cosine SIM_3.

The last coefficient, SIM_5, is an asymmetric measure; that is the similarity between DOC_i and DOC_j is not in general equal to the similarity between DOC_j and DOC_i. Indeed $SIM_5(DOC_i,DOC_j) = 3/8$ whereas $SIM_5(DOC_j,DOC_i) = 3/4$. Asymmetric measures are useful to capture the inclusion relations between vectors (vector B is included in vector A if all properties assigned to B are also present in A). The inclusion properties between vectors can be used for the generation of hierarchical arrangements of objects (for example, hierarchical term displays) and for the comparison of queries with documents in retrieval [1].

A great many different similarity measures are discussed in the literature designed to represent the associations between different property vectors. Some of the functions reflect statistical theories, being designed to measure the agreement between two vectors over and above the coincidences that would be expected if the properties were randomly assigned to the vectors [3,4]. In some similarity functions the absence of properties from a vector may be taken into account as well as the presence. For instance the joint absence of a property in two vectors may be weighted differently from the joint presence of a property [5].

All similarity measures exhibit one common property, namely that their value increases when the number of common properties (or the weight of the common properties) in two vectors increases. Measures of vector dissimilarity which are sometimes used instead of similarity measures have the opposite effect. Various evaluation studies exist in which the effect of different similarity measures has been compared in a retrieval environment [1]. The Jaccard [expression (2)] and the cosine measures [expression (3)] have similar characteristics, ranging from a minimum of 0 to a maximum of 1 for nonnegative vector elements. These measures are easy to compute and they appear to be as effective in retrieval as other more complicated functions. Both these measures have been widely used for the evaluation of retrieval functions.

3 TERM WEIGHTING SYSTEMS

A Principal Weighting Strategies

In principle, the retrieval environment is simplest when the information items are characterized by unweighted properties and the indexing operation is binary. In this case, the degree to which a given property (term) may be useful to represent the content of an item is not a consideration. Any property that appears relevant is assigned to the information item and rejected when it appears extraneous. While the indexing is simplified, the task of evaluating the output of a search operation may be complicated because distinctions among the retrieved items, or for that matter among the items that are not retrieved, are more difficult to make for binary than for weighted vectors. When weighted properties are used, a similarity computation between the query and document vectors makes it possible to retrieve the items in strictly *ranked order* according to the magnitude of the query-document similarity coefficients. This should improve the retrieval effectiveness and lighten the user effort required to generate a useful query.

When users, indexers, or search intermediaries manually assign term weights to document and query vectors, the weighting operation is difficult to control. A satisfactory assignment of weights requires a great deal of know-how about the collection and the operation of the retrieval system. For this reason, an effective term weighting operation is probably best conducted by using objective term characteristics automatically to generate term weights.

Several automatic term weighting systems were introduced in Chapter 3 in the discussion on automatic indexing. They are summarized here for convenience:

1 The term frequency (TF) weighting system is based on the notion that constructs (words, phrases, word groups) that frequently occur in the text of documents have some bearing on the content of the texts. Hence the weight of term k in document i, WEIGHT_{ik} might be set equal to the frequency of occurrence of word construct k in document i:

$$\text{WEIGHT}_{ik} = \text{FREQ}_{ik} \tag{6}$$

2 The term frequency system makes no distinction between terms that occur in every document of a collection and those that occur in only a few items. Experience indicates that the usefulness of a term for content representation increases with the frequency of the term in the document but decreases with the number of documents DOCFREQ_k to which the term is assigned. This produces the inverse document frequency (IDF) weighting system:

$$\text{WEIGHT}_{ik} = \frac{\text{FREQ}_{ik}}{\text{DOCFREQ}_k} \tag{7}$$

3 The term discrimination theory depends on the degree to which the assignment of a term to the documents of a collection is capable of decreasing the density of the document space (the average distance between documents). The discrimination value of term k, DISCVALUE_k is obtained as the difference between two measurements of document space density, corresponding to the densities before and after assignment of term k. A typical weighting function for term k in document i is then obtained as

$$\text{WEIGHT}_{ik} = \text{FREQ}_{ik} \cdot \text{DISCVALUE}_k \tag{8}$$

4 The probabilistic indexing theory states that the best index terms are those that tend to occur in the relevant documents with respect to some query. When the terms are assigned to the documents independently of each other, a measure of term value is obtained from the term relevance TERMREL_k. This is the ratio of the proportion of relevant items in which term k occurs to the proportion of nonrelevant items in which the term occurs [expression (23) of Chapter 3]. A weighting system based on the term relevance is thus

$$\text{WEIGHT}_{ik} = \text{FREQ}_{ik} \cdot \text{TERMREL}_k \tag{9}$$

The term frequency, document frequency, and term discrimination theories were previously examined in the discussion dealing with automatic indexing. The term relevance weighting system is theoretically optimal given certain well-specified conditions. However the term relevance factor

$$\text{TERMREL}_k = \frac{r_k/(R - r_k)}{s_k/(I - s_k)}$$

cannot be computed unless relevance assessments are available of the documents with respect to certain queries. In particular, the number of relevant documents (r_k) containing term k, the number of nonrelevant documents (s_k) containing term k, as well as the total number of relevant documents (R) and nonrelevant documents (I) in the collection for some particular query sets must be known in advance.

A similar situation arises for a weighting function based on the *utility* value introduced in the discussion on system evaluation [expression (19) of Chapter 5]. The utility of a search is defined simply as the sum of the values achieved by retrieving relevant items and rejecting nonrelevant ones plus the sum of the costs incurred by retrieving nonrelevant and rejecting relevant items. One may assume that each relevant item that is correctly retrieved increases the usefulness of retrieval by a specified value equal to v_1; similarly each nonrelevant item that is properly rejected increases the system usefulness by a constant value of v_2. Analogously, a constant cost of c_1 is incurred for each nonrelevant item that is retrieved, and a cost of c_2 arises for each relevant item missed by the retrieval system. In these circumstances, appropriate transformations of the utility value introduced in Chapter 5 produce a weighting function, known as the utility weight for a given term k, or UTILITY_k, defined as

$$\text{UTILITY}_k = (v_1 + c_2)r_k - (v_2 + c_1)s_k \tag{10}$$

where r_k and s_k, respectively represent the number of relevant and nonrelevant items containing term k [6]. A corresponding term weighting function for term k in document i is then given by

$$\text{WEIGHT}_{ik} = \text{FREQ}_{ik} \cdot \text{UTILITY}_k \tag{11}$$

The utility and term relevance weighting systems of expressions (9) and (11) may be expected to be more powerful than the alternative weighting schemes based on simple term frequency characteristics [expressions (6) and

(7)]. In the utility and term relevance systems, a distinction is made between term occurrences in the relevant and nonrelevant documents, respectively, whereas in the frequency-based systems the term occurrences are used globally over the whole collection irrespective of occurrences in the relevant and nonrelevant documents. On the other hand, it remains to be seen whether for the relevance-based weighting systems the term weights computed by using relevance data for certain queries and documents will prove robust enough to be applied to new documents and queries for which no prior relevance data are available. Procedures for so doing are suggested in the next section.

*B Evaluation of Weighting Systems

Two collections of documents may serve as examples for the evaluation of the weighting systems introduced earlier. These are the Cranfield collection of 424 documents in aerodynamics, and the MEDLARS collection of 450 documents in biomedicine. Each collection is used with 24 search requests. Table 6-1 contains evaluation output for term frequency weightings (6), inverse document

Table 6-1 Term Utility Weight and Relevance Weight Evaluation
(Average Precision Values for Fixed Levels of Recall from 0.1 to 1.0)

Recall	Term frequency	Inverse document frequency		Utility weight $w = 20r - s$ (actual values)		Relevance weights $w = [r/(R-r)] \div [s/(I-s)]$ (actual values)	
	a Cranfield aerodynamics collection (424 documents, 24 queries)						
0.1	0.455	0.566	+24%	0.568	+25%	0.571	+25%
0.2	0.410	0.530	+29%	0.540	+32%	0.558	+36%
0.3	0.391	0.476	+22%	0.503	+29%	0.479	+23%
0.4	0.301	0.421	+40%	0.474	+57%	0.479	+59%
0.5	0.280	0.364	+30%	0.416	+49%	0.434	+55%
0.6	0.233	0.301	+29%	0.328	+41%	0.352	+51%
0.7	0.189	0.254	+34%	0.272	+44%	0.324	+71%
0.8	0.155	0.195	+26%	0.211	+36%	0.220	+42%
0.9	0.121	0.150	+24%	0.162	+34%	0.199	+64%
1.0	0.112	0.132	+18%	0.143	+29%	0.164	+46%
			+27.6%		+37.6%		+47.2%
	b MEDLARS biomedical collection (450 documents, 24 queries)						
0.1	0.543	0.611	+13%	0.676	+24%	0.707	+30%
0.2	0.528	0.601	+14%	0.676	+28%	0.707	+34%
0.3	0.467	0.541	+16%	0.639	+37%	0.705	+51%
0.4	0.421	0.467	+11%	0.609	+45%	0.672	+60%
0.5	0.384	0.438	+14%	0.558	+45%	0.633	+65%
0.6	0.346	0.396	+14%	0.510	+47%	0.616	+78%
0.7	0.316	0.347	+10%	0.459	+45%	0.573	+81%
0.8	0.211	0.245	+16%	0.374	+77%	0.462	+119%
0.9	0.171	0.193	+13%	0.277	+62%	0.354	+107%
1.0	0.120	0.154	+28%	0.204	+70%	0.259	+116%
			+14.9%		+48%		+74.1%

frequency weights (7), utility weights (11), and term relevance weights (9). In each case precision values are shown in Table 6-1 for 10 recall levels varying from 0.1 to 1.0. These are averaged for the 24 search requests. The inverse document frequency weights that are based on frequency characteristics over all documents regardless of relevance produce average precision improvements of about 28 percent for the aerodynamics collections and 15 percent for the biomedical collection over the standard term frequency weights. When the utility weights based on document relevance are used the average advantage in precision increases to 38 and 48 percent, respectively, while an even greater advantage of 46 and 74 percent is obtained for the term relevance weights.

To generate the term relevance and term utility weights TERMREL_k and UTILITY_k, respectively, it is necessary to identify the number of relevant and nonrelevant documents r_k and s_k in which the term occurs. Furthermore, for the utility weights, values must be chosen for the value and cost parameters of equation (10). The output of Table 6-1 is based on the assumption that the cost and value parameters associated with the relevant documents (v_1 and c_2) are given a weight equal to 20 times the value and cost parameters associated with the nonrelevant (v_2 and c_1). The utility function of expression (10) is thus computed as $20r_k - s_k$.

The problem of generating the r_k and s_k values was bypassed in the experiments of Table 6-1 by using the actual values found in the two sample collections for these parameters. That is, the unrealistic assumption was made that the characteristics of all terms in the relevant and nonrelevant documents were known in advance for all queries. This, of course, accounts for the excellent performance of the term utility and term relevance weighting systems in the output of Table 6-1.

In practice, the occurrence characteristics of the terms in the relevant and nonrelevant documents are not available before a search is actually conducted. However, the total number of documents DOCFREQ_k to which a given term is assigned is given, and that in turn can be used to estimate the number of relevant documents (r_k) having term k. Note that the document frequency of a term varies from 0 for a term not assigned to any document in the collection to a maximum of N for a term assigned to all N items in a collection. The parameter r_k, on the other hand, varies from 0 for a term not assigned to any relevant items to a maximum of R, the total number of relevant items which exists with respect to a given query. Alternatively R can be interpreted in some circumstances as the number of documents which a user wishes to retrieve in response to a given query.

Normally, the following relationships exist between the total document frequency DOCFREQ_k of a term, and the frequency r_k in the relevant documents:

1 As DOCFREQ_k increases, so will r_k; thus given two terms TERM_j and TERM_k, $\text{DOCFREQ}_j > \text{DOCFREQ}_k$ generally implies $r_j > r_k$.

2 For normal query terms, the number of relevant documents in which a

term occurs is relatively larger for lower-frequency terms than for higher-frequency terms; mathematically, one can say that when $\text{DOCFREQ}_j > \text{DOCFREQ}_k$, one finds that $r_k/\text{DOCFREQ}_k > r_j/\text{DOCFREQ}_j$. (For example, the one document in which a frequency-one term occurs is more likely to be relevant than the two documents for a term of frequency two.)

Several simple functions can be suggested that conform to these conditions. One possible functional relationship between r_k and DOCFREQ_k for a given term k is shown in the graph of Fig. 6-1. Here for document frequencies between 0 and R, one assumes that a straight-line relation exists between DOCFREQ and r given as $r = (a \cdot \text{DOCFREQ})$ for some constant $a < 1$, and represented by line segment 0A. For frequencies DOCFREQ between R and N, another straight-line relation is assumed expressed as $r = d + (e \cdot \text{DOCFREQ})$ and represented by segment AB. It may be noted that in accordance with assumption 2, the slope of line AB (represented by parameter e) is smaller than the slope of line 0A (parameter a). As a result the proportion $r_k/\text{DOCFREQ}_k$ is relatively larger for terms of smaller frequency DOCFREQ_k than for terms of larger frequency. An alternative functional relationship between r and DOCFREQ which also obeys assumptions 1 and 2 is $r = a + b(\log \text{DOCFREQ})$. It can be shown that if the relationship between DOCFREQ_k and r_k is the one represented in Fig. 6-1, the best term weighting function has the shape represented in Fig. 6-2 [7].

In particular, the optimum weight of a term starts with some constant value a for terms of frequency 1. The weight then increases as the document frequency increases to R, the number of relevant documents which a user wishes to retrieve in response to a query. As the document frequency increases still further, the terms become less important and the term weight decreases. Eventually, for terms of document frequency near the number of documents in the data base (N), the weight decays to 0. The frequency spectrum of Fig. 6-2

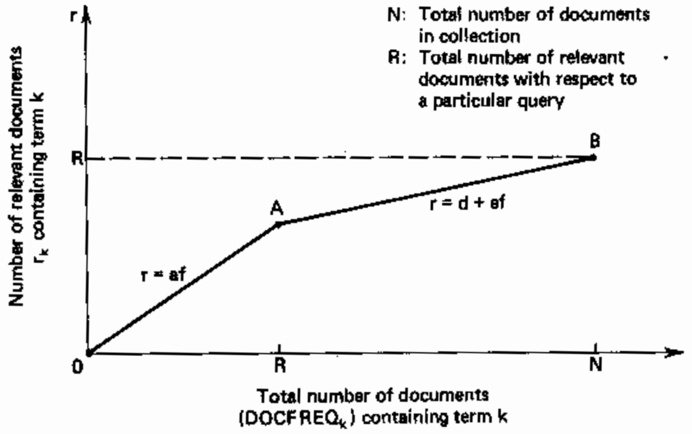

Figure 6-1 Variation of number of relevant documents containing term k with total number of documents containing term k.

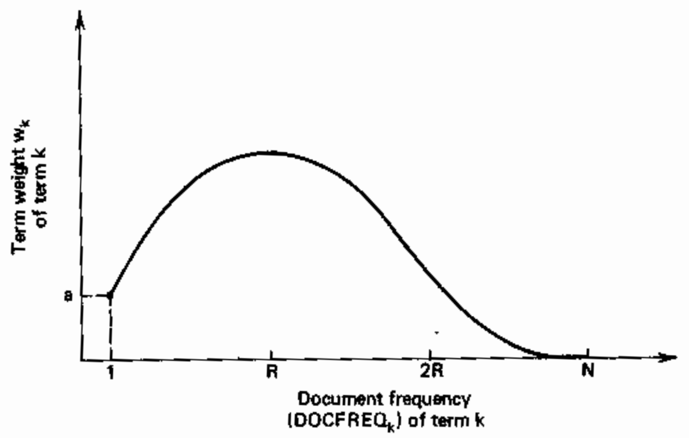

Figure 6-2 Optimum term weighting system assuming relationships of Fig. 6-1.

once again shows that the medium-frequency terms in a collection are the most important for purposes of document indexing.

Consider now the evaluation results obtained for the utility and term relevance weighting schemes where the parameter values for r_k and s_k are no longer assumed to be available, but r_k (and hence $s_k = \text{DOCFREQ}_k - r_k$) is obtained by using one of the functional relationships between DOCFREQ and r (for example the one presented earlier in Fig. 6-1). Evaluation results for the term utility and term relevance weighting systems based on estimated r_k values are included in Table 6-2 for the two document collections previously used in Table 6-1. For the utility weights a logarithmic relationship was assumed between r and DOCFREQ [that is, $r = a + b \log (\text{DOCFREQ})$ for suitably chosen parameter values a and b]. A hybrid function modeled

Table 6-2 Estimated Term Utility and Relevance Weight Evaluation
(Average Precision Values for 24 Queries at Recall Levels from 0.1 to 1.0)

Cranfield aerodynamics (424 documents, 24 queries)				**MEDLARS biomedical (450 documents, 24 queries)**			
Utility weight (estimated values)		**Relevance weight (estimated values)**		**Utility weight (estimated values)**		**Relevance weight (estimated values)**	
0.531	+17%	0.552	+21%	0.592	+9%	0.629	+16%
0.501	+22%	0.520	+27%	0.579	+10%	0.629	+19%
0.450	+15%	0.461	+18%	0.511	+9%	0.601	+29%
0.388	+29%	0.421	+40%	0.440	+5%	0.536	+27%
0.332	+19%	0.369	+32%	0.396	+3%	0.512	+33%
0.288	+24%	0.303	+30%	0.333	−4%	0.456	+32%
0.234	+24%	0.259	+37%	0.309	−2%	0.409	+29%
0.184	+19%	0.192	+24%	0.233	+10%	0.296	+40%
0.138	+14%	0.159	+31%	0.186	+9%	0.218	+27%
0.128	+14%	0.131	+17%	0.139	+16%	0.169	+41%
	+19.7%		+27.7%		+6.5%		+29.3%

on the relationship of Fig. 6-1 is used to relate r and DOCFREQ for the relevance weight calculation: in particular, a straight line similar to line segment 0A of Fig. 6-1 ($r = a \cdot \text{DOCFREQ}$) is used for document frequency values up to DOCFREQ = 8; for larger values of DOCFREQ a logarithmic relationship ($r = d + e \log \text{DOCFREQ}$) is assumed between r and DOCFREQ.

A comparison between the output of Tables 6-1 and 6-2 indicates that the utility and relevance weighting systems are not as powerful when the parameter values must be estimated than when actual values are available. However the relevance weighting system appears to be more effective than the inverse document frequency even when the relevance parameters are estimated. Since the estimated relevance weights are based purely on the occurrence frequencies of the terms in the documents of a collection, the results of Table 6-2 confirm that substantially more information may be contained in the term frequency data than is normally included in conventional retrieval. Additional work is needed to produce good estimates of term relevance and justification for the curves of Figs. 6-1 and 6-2.

**C Term Weighting in Boolean Query Systems

It was mentioned earlier that systems based on Boolean query formulations are capable of separating a document collection into two parts consisting of the retrieved items on one hand and the rejected (nonretrieved) ones on the other. Additional operations are sometimes carried out for the set of retrieved documents only in order to generate additional discrimination or ranking among these documents. No term weights need to be introduced for this purpose and no changes arise in the interpretation of the normal Boolean operations. The question arises whether the necessary discrimination among documents can be obtained directly by reinterpreting the standard Boolean operations to render them applicable to systems using weighted query terms, and possibly weighted documents.

It is not possible in the present context to examine in detail the questions relating to the processing of weighted Boolean queries [8,9]. It may be sufficient instead to suggest some obvious approaches that lend themselves to a practical implementation. Consider two arbitrary index terms A and B, and let **A** and **B** represent the set of documents indexed by terms A and B, respectively. The Boolean operations normally receive the following interpretation:

1 The query "A OR B" is designed to retrieve the document set ($\mathbf{A} \cup \mathbf{B}$) consisting of documents indexed by term A or by term B or by both A and B.

2 The query "A AND B" retrieves document set ($\mathbf{A} \cap \mathbf{B}$) consisting of documents indexed by both terms A and B.

3 The query "A NOT B" retrieves document set ($\mathbf{A} - \mathbf{B}$) consisting of documents indexed by term A that are not also indexed by B.

Let a and b denote term weights varying from a minimum of 0 to a maximum of 1, and consider an extension of those operations that includes the use of weighted query terms. When the term weight is chosen equal to 1, the normal Boolean operation is implied, whereas a term weight of 0 implies that the corresponding operand may be disregarded. Thus one has

$$\begin{aligned}
&A_1 \text{ OR } B_1 \equiv A \text{ OR } B\\
&A_1 \text{ AND } B_1 \equiv A \text{ AND } B\\
&A_1 \text{ NOT } B_1 \equiv A \text{ NOT } B\\
\text{and}\quad &A_1 \text{ OR } B_0 \equiv A\\
&A_1 \text{ AND } B_0 \equiv A\\
&A_1 \text{ NOT } B_0 \equiv A
\end{aligned}$$

When both the document and the query terms are weighted, a weighted Boolean query now receives a simple interpretation. In response to a query such as A_a OR B_b, the set of retrieved documents consists of those having either term A with a weight at least equal to a or term B with a weight at least equal to b. The retrieved items can be ranked according to the sum of the weights $a + b$ in the documents.

When only the documents are weighted, but not the queries, the full document sets **A** and/or **B** are retrieved using the appropriate Boolean combination, and the ranking applies as before. This situation appears simple to implement in operational retrieval because weights can be assigned to the document terms by the expert indexers, or frequency-based weights can be automatically obtained by the system. To assign weights to the query terms, some input is needed from the users, and reliable term weighting information of this kind may be difficult to obtain.

In the unlikely situation where the query terms alone are weighted but the document terms are not, it appears reasonable to suggest that each weighted query term affects a partial set of documents instead of the full set. Consider as an example, query statements of the form $(\ldots ((A_a * B_b) * C_c). \ldots * Z_z)$, where $*$ stands for one of the operators AND, OR, NOT, and where a, b, . . . , z represent weights attached to terms A, B, . . . , Z respectively, such that $0 \le a \le 1, \ldots, 0 \le z \le 1$. The general case involving a multiplicity of binary $*$ operators may be reduced to that of a single binary operator with two operands by assuming that the search process is carried out iteratively, one operator at a time. The problem then consists of interpreting query statements of the form $(A_a * B)$, where $*$ is a binary connective and a and b are the term weights.

The operations of the three Boolean connectives may be described by considering the special case where only one of the two operands carries a weight smaller than 1, that is, where the queries have the form $(A_1 * B_b)$. Extensions to the general case where both query terms carry weights less than unity will then be immediate. Remembering that query term B_0 can be disregarded, whereas B_1 covers the full set **B** of documents indexed by B, it becomes clear that query A OR B_b expands the output document set from **A** to **A** $\cup$ **B** as the weight of b increases from 0 to 1. $\mathbf{A} \cup \mathbf{B}$ comprises the full set of items that are either **A**'s or **B**'s. A query such as (A OR $B_{0.33}$) must then retrieve all the **A**'s plus a third of the **B**'s. Correspondingly, (A AND B_b) shrinks the size of the output from **A** to $\mathbf{A} \cap \mathbf{B}$, that is, to the set of items that are both **A**'s and **B**'s as b increases from 0 to 1. This suggests that (A AND $B_{0.33}$) covers all the **A**'s that are also in **B** plus about two-thirds of the **A**'s that are not in **B**. Finally, (A NOT B_b) shrinks the output from **A** to **A** – **B**, that is, to the items in **A** that are not also in **B**. The query (A NOT $B_{0.33}$) would then cover all **A**'s that are not in **B** plus two-thirds of the items in the intersection between **A** and **B**.

It remains to determine how the partial set of items that are either included in or excluded from the answering document set is to be identified. The following mode of operation suggests itself:

1 OR operation: as b increases from 0 to 1, the items in **B** not already in **A** that are *closest* to the set **A** are successively added to **A** to generate $\mathbf{A} \cup \mathbf{B}$ in answer to query (A OR B).

2 AND operation: as b increases from 0 to 1, the items in **A** − **B** that are *farthest* from $\mathbf{A} \cap \mathbf{B}$ are successively subtracted from **A** until only $\mathbf{A} \cap \mathbf{B}$ remains in answer to query (A AND B) when b is equal to 1.

3 NOT operation: as b increases from 0 to 1, the items in $\mathbf{A} \cap \mathbf{B}$ that are *farthest* from **A** − **B** are successively subtracted from **A** until only **A** − **B** remains in answer to query (A NOT B).

To determine the closeness of a particular document to another document or to a set of documents, the similarity coefficients previously introduced to compare queries and documents [expressions (1) to (5)] can be used to obtain affinity indicators between pairs of documents or between a particular document and a set of documents. In the latter case, a typical document C is chosen to represent the given set, such as, for example, the centroid of the document set, and for each document DOC_i, the size of the coefficient $SIM(C,DOC_i)$ is used to indicate whether DOC_i is to be retrieved or not. The computation of a cluster centroid is described in detail in the next section of this chapter.

Consider, as a typical example, the operations for query ($A_{0.33}$ OR $B_{0.66}$) illustrated in Fig. 6-3 together with other examples. The following steps may be used:

1 Compute the centroids of sets **A** and **B**.

2 Remove from set **A** two-thirds of the documents consisting of those exhibiting the largest distance to the centroid of **B**.

3 Remove from set **B** one-third of the documents consisting of those exhibiting the largest distance to the centroid of **A**.

4 The response set is then the union of the remaining items from **A** and **B**

Correspondingly, the output set for query ($A_{0.33}$ AND $B_{0.66}$) (see Fig. 6-3b) is obtained by removing from set **A** one-third of the items not included in the intersection of **A** and **B** and situated farthest from the centroid of **B**; at the same

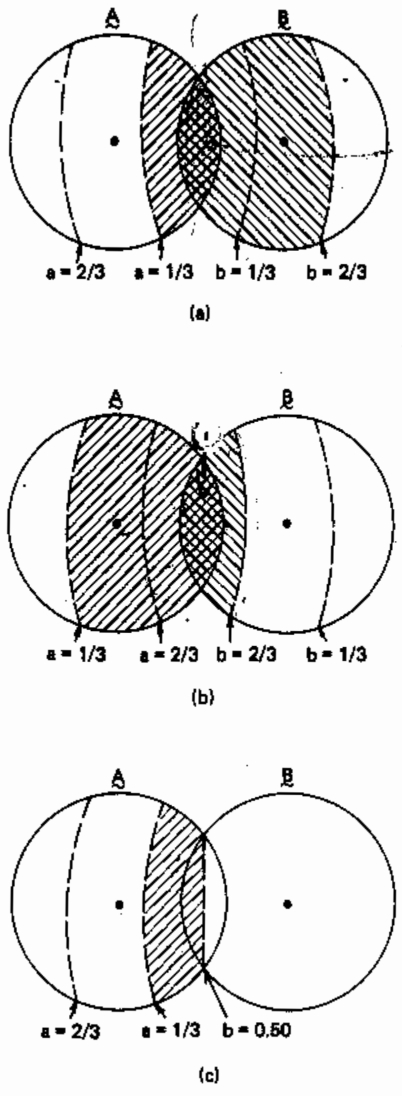

Figure 6-3 Interpretation of weighted Boolean operations. (a) ($A_{0.33}$ OR $B_{0.66}$). (b) ($A_{0.33}$ AND $B_{0.66}$). (c) ($A_{0.33}$ NOT $B_{0.50}$).

time, one removes from **B** two-thirds of the items that are most removed from set **A**. For the query ($A_{0.33}$ NOT $B_{0.50}$) of Fig. 6-3c, two-thirds of the items in **A** are removed from the answer set plus half of the items in the intersection between **A** and **B**.

To summarize, when weighted terms are used for queries that do *not* include Boolean operators, a query-document similarity computation can be used directly to obtain a retrieval value, or ranking, for each document; documents may then be retrieved in decreasing order of their retrieval values. When Boolean operators are included in the queries, a similarity computation is first carried out between certain documents and the centroids, or representatives, of certain document sets. The size of the corresponding similarity coefficients then determines which documents are to be added to (for the OR operation) or subtracted from (for AND and NOT) the basic answering set. The use of term weights in Boolean systems remains to be validated by appropriate retrieval experiments.

4 FILE CLUSTERING

*A Main Considerations

Most information retrieval work is based on the manipulation of large masses of data. The document files to be stored may be extensive, and the vocabularies needed to represent document content may include tens of thousands of terms. In these circumstances, it is useful to superimpose an organization on the stored information in order to simplify file access and manipulation. One way of providing order among a collection of stored records is to introduce a classification, or *clustering*, among the items. Clustering is used to group similar or related items into common classes. In a classified or clustered file, items appearing in a class can be stored in adjacent locations in the file so that a single file access makes available a whole class of items. Such an approach is used in most conventional libraries where the library items are placed on shelves according to their subject content. By browsing among the shelves, the library users can then retrieve a number of different items within a given subject area.

In information retrieval, classification methods are used for two main purposes:

1 To classify the set of *index terms*, or keywords, into term classes according to similarities in the keywords, or according to statistical characteristics of the terms in the documents of a collection

2 To classify the *documents* into subject classes so that related items are accessible to the user population

The *keyword classifications* lead to the construction of thesauruses and synonym dictionaries that can be used for document indexing and query formulation. These may also provide the associative indexing capability previously illustrated in Chapter 3. The *document classifications* on the other hand may serve as devices for the representation of knowledge, and in retrieval they may provide efficient search strategies and effective search results. The efficiency is produced by making it possible for the user to limit the search to specific subject areas. The potential effectiveness of the cluster search process stems from the *cluster hypothesis*, which asserts that closely associated documents tend to be relevant to the same queries [2]. The use of clustered document files may then lead both to high recall and to high precision searches [10].

The following two characteristics are generally considered important for object classifications [11,12]: