Chapter 6

Retrieval Refinements

0 PREVIEW

A number of advanced analysis and search techniques have been mentioned in some of the previous chapters, including in particular the use of term weighting methods for indexing and query formulation, the introduction of clustered file arrangements, the dynamic improvement of query formulations using relevance feedback techniques, and the use of bibliographic citations for content identification and retrieval purposes. In the present chapter, these techniques are examined in more detail, and methods are given for implementing the various techniques in retrieval. Wherever possible evaluation results are included showing the usefulness of the various methodologies in retrieval.

Some of the procedures examined in this chapter should prove immediately useful in conventional and experimental retrieval situations. This is the case notably of some term weighting systems that are easy to generate and exceptionally effective, and of the automatic query adjustment systems based on user-system interaction during the course of the search process. Certain other techniques, such as the dynamic document space modification and the automatic thesaurus construction process using pseudoclassification, may prove important in the long run. All the retrieval refinements described in this chapter are conceptually simple and some of them have already been applied in information retrieval under operational conditions.

1 INTRODUCTION

Conventional retrieval operations use Boolean search requests and inverted file systems. A search then produces two distinct document sets: those retrieved in answer to a given query and those not retrieved. Relationships or similarities between individual documents are not utilized and neither are relationships between keywords or query terms. The experimental retrieval systems described in Chapter 4 are more advanced largely because some structure is imposed on the retrieval environment. Thus, by recognizing relationships among the documents of a collection, items which appear to be related can be grouped into affinity classes, to permit browsing and to simplify searches dealing with particular subject areas. By making distinctions among the terms assigned to the documents, some documents—normally those identified by highly weighted terms—can be retrieved ahead of certain others identified by terms of lesser importance. By using the occurrence characteristics of the terms assigned to a collection, it is possible to identify very broad terms assigned to a large proportion of the documents as well as narrow terms assigned to few documents. The former can be rendered less general by combining them into term phrases, and the latter can be broadened by grouping them into thesaurus classes of wider scope.

The foregoing processes produce a greater measure of discrimination among the documents of a collection and among the terms characterizing document content than is customary in conventional retrieval situations. When the documents are clearly distinguished from each other, maximum recall and precision may be obtained for a search, because each particular relevant item may then be retrievable without also retrieving neighboring items that may not be relevant. Furthermore, documents that are distinguished from each other can be ranked for output purposes in decreasing order of the similarity between query and documents. This brings the most important items to the users' attention early in a search when they are most easily used for the generation of improved search formulations in an interactive search mode.

The discrimination operations all depend on the assignment of importance factors, or weights, to the content identifiers used for a document collection, and on the computation of similarity measures between documents, between terms, or between queries and documents. In this chapter, the properties of vector similarity functions are first outlined and some of the applications of term weighting and vector similarity computations previously mentioned for the experimental systems of Chapter 4 are covered in greater detail. This includes the generation of optimum term weights, the construction of document clusters, and the generation of improved query formulations and better document identifications. Normally, the affinities between document vectors are measured by comparing the terms attached to the respective items. In some circumstances, the terms may be replaced by bibliographic citations. In particular, the structure of a document collection might be determined by the citations and bibliographic references relating the items. This possibility is further examined at the end of this chapter.

*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 $\sqrt{\sum_{k=1}^{t} (TERM_{ik})^2}$, 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:

$$DOC_i = (3,2,1,0,0,0,1,1)$$
$$DOC_j = (1,1,1,0,0,1,0,0)$$

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

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

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

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

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

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

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]:

$$SIM_1(DOC_i, DOC_j) = \frac{2\left[\sum_{k=1}^{t} (TERM_{ik} \cdot TERM_{jk})\right]}{\sum_{k=1}^{t} TERM_{ik} + \sum_{k=1}^{t} TERM_{jk}} \quad (1)$$

$$SIM_2(DOC_i,DOC_j) = \frac{\sum_{k=1}^{t} (TERM_{ik} \cdot TERM_{jk})}{\sum_{k=1}^{t} TERM_{ik} + \sum_{k=1}^{t} TERM_{jk} - \sum_{k=1}^{t} (TERM_{ik} \cdot TERM_{jk})} \quad (2)$$

$$SIM_3(DOC_i,DOC_j) = \frac{\sum_{k=1}^{t} (TERM_{ik} \cdot TERM_{jk})}{\sqrt{\sum_{k=1}^{t} (TERM_{ik})^2 \cdot \sum_{k=1}^{t} (TERM_{jk})^2}} \quad (3)$$

$$SIM_4(DOC_i,DOC_j) = \frac{\sum_{k-1}^{t} (TERM_{ik} \cdot TERM_{jk})}{\min\left(\sum_{k=1}^{t} TERM_{ik}, \sum_{k=1}^{t} TERM_{jk}\right)} \quad (4)$$

$$SIM_5(DOC_i,DOC_j) = \frac{\sum_{k=1}^{t} \min(TERM_{ik},TERM_{jk})}{\sum_{k=1}^{t} TERM_{ik}} \quad (5)$$

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

$$SIM_1(DOC_i,DOC_j) = \frac{2 \cdot (6)}{8 + 4} = 1$$

$$SIM_2(DOC_i,DOC_j) = \frac{6}{8 + 4 - 6} = 1$$

$$SIM_3(DOC_i,DOC_j) = \frac{6}{\sqrt{16 \cdot 4}} = \frac{6}{\sqrt{64}} = \frac{6}{8} = 0.75$$

$$SIM_4(DOC_i,DOC_j) = \frac{6}{4} = 1.5$$

$$SIM_5(DOC_i,DOC_j) = \frac{3}{8} = 0.375$$

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 vec-

tors. 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 $TERM_k$, $DOCFREQ_j > DOCFREQ_k$ 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 $DOCFREQ_j > DOCFREQ_k$, one finds that $r_k/DOCFREQ_k > r_j/DOCFREQ_j$. (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 $r = (a \cdot DOCFREQ)$ 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 $r = d + (e \cdot DOCFREQ)$ 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 $r_k/DOCFREQ_k$ 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 $r = a + b(\log DOCFREQ)$. 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

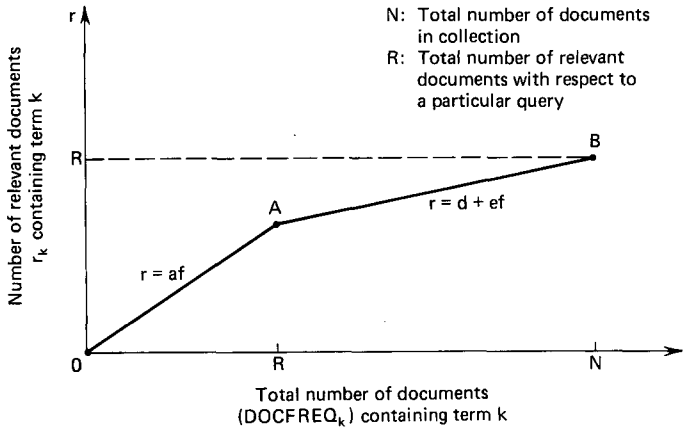

Figure 6-1 Variation of number of relevant documents containing term k with total number of documents containing term k.

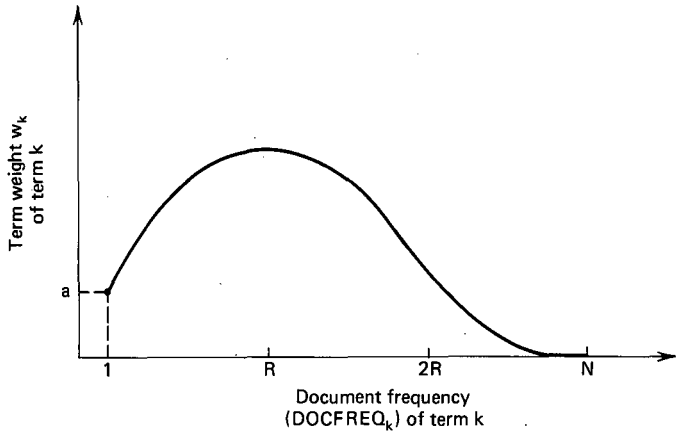

Figure 6-2 Optimum term weighting system assuming relationships of Fig. 6-1.

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 ($r = a \cdot DOCFREQ$) is used for document frequency values up to DOCFREQ = 8; for larger values of DOCFREQ a logarithmic relationship ($r = d + e \log DOCFREQ$) 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

$$A_1 \text{ OR } B_1 \equiv A \text{ OR } B$$
$$A_1 \text{ AND } B_1 \equiv A \text{ AND } B$$
$$A_1 \text{ NOT } B_1 \equiv A \text{ NOT } B$$

and

$$A_1 \text{ OR } B_0 \equiv A$$
$$A_1 \text{ AND } B_0 \equiv A$$
$$A_1 \text{ NOT } B_0 \equiv A$$

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 (. . . $((A_a * B_b) * C_c)$. . . . $* Z_z)$, 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 $0 \leq a \leq 1$, . . . , $0 \leq z \leq 1$. 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. **A** ∪ **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 **A** ∩ **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 **A** ∪ **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 **A** ∩ **B** are successively subtracted from **A** until only **A** ∩ **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 **A** ∩ **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

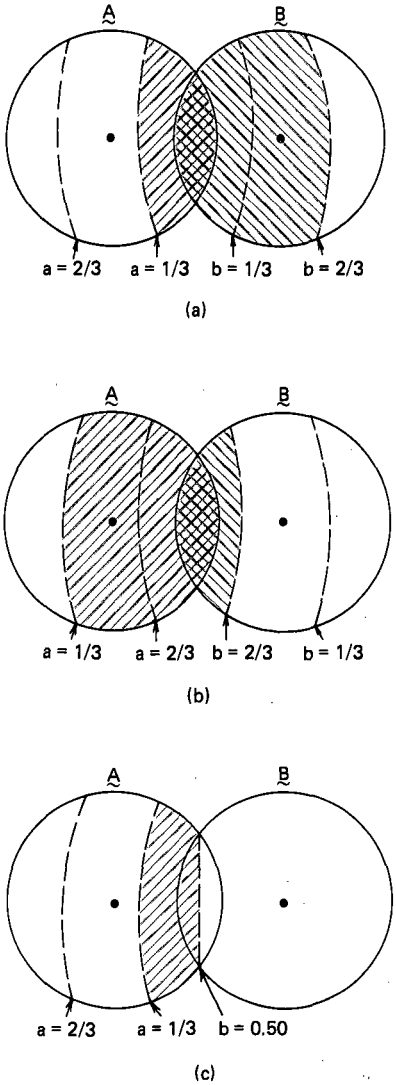

Figure 6-3 Interpretation of weighted Boolean operations. (a) ($A_{0.33}$ OR $B_{0.66}$). (b) ($A_{0.33}$ AND $B_{0.66}$). (c) ($A_{0.33}$ NOT $B_{0.50}$).

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]:

1 The classification should be *stable* in the sense that small alterations of the data, because of either the addition of new items or changes in the old ones, should cause only minor alterations in the classification.

2 The classification should also be *well defined* in that a given body of data should produce a single classification or at least one of a small set of compatible classifications.

As it will be seen, the second property is not always present in various heuristic or single-pass classification systems constructed for actual objects. Rather, examples exist where a variety of different classification systems all perform equally well in practical applications. Stability, on the other hand, appears to be essential if a classification is to operate satisfactorily.

Many different methods can be used to cluster a collection of items. If the classification is to perform a useful role, controls must normally be introduced to limit the minimum and maximum number of classes that are generated, as well as the size of the classes and the overlap between classes. Obviously, the number of classes generated should be greater than 1 but smaller than the total number of objects to be classified. It is helpful if the various classes exhibit a roughly comparable size. Further, storage efficiency considerations make it necessary to limit the overlap between classes, defined as the number or proportion of items jointly assigned to two or more classes.

The number, size, and overlap of the classes may be controlled during cluster generation by parameter settings. An example is given in Fig. 6-4 for six items labeled D_1 to D_6. The input to the clustering process is assumed to be a similarity matrix giving similarity measures between all pairs of items as shown in Fig. 6-4a. The clustering process then consists in fixing a threshold T in the similarity coefficients, and grouping all pairs of items whose similarity exceeds the chosen threshold. The illustration of Fig. 6-4b shows that when $T = 0.95$, no clustering action is possible, and the file consists of 6 unclustered items. As the threshold is lowered to 0.8, three pairs of items can be grouped including (D_1,D_5), (D_1,D_6), and (D_2,D_6); pairs of related items are identified by lines joining the respective items in the graph of Fig. 6-4c. As the threshold value is further reduced to 0.60 and 0.50 additional pairs of items can be joined as shown in Fig. 6-4d and e.

For purposes of the example, one may assume that each cluster of items is defined as a *connected component* in the sense that in a graph representation such as that of Fig. 6-4, a path formed by lines exists from any item to all other items in the cluster. In these circumstances, the illustration of Fig. 6-4 demonstrates that the number of clusters formed decreases and the size of the clusters increases as the similarity threshold is decreased. The number of clusters produced by the four thresholds of Fig. 6-4b to 6-4e is, respectively, 6, 3, 2, and 1.

In information retrieval the cluster processing operation consists in taking an item—for example, a new incoming document or an incoming user query—and comparing it with an existing cluster to determine the affinity of the item with that cluster. This operation is easily carried out by defining for each clus-

	D_1	D_2	D_3	D_4	D_5	D_6
D_1	–	0.3	0.5	0.6	0.8	0.9
D_2	0.3	–	0.4	0.5	0.7	0.8
D_3	0.5	0.4	–	0.3	0.5	0.2
D_4	0.6	0.5	0.3	–	0.4	0.1
D_5	0.8	0.7	0.5	0.4	–	0.3
D_6	0.9	0.8	0.2	0.1	0.3	–

(a)

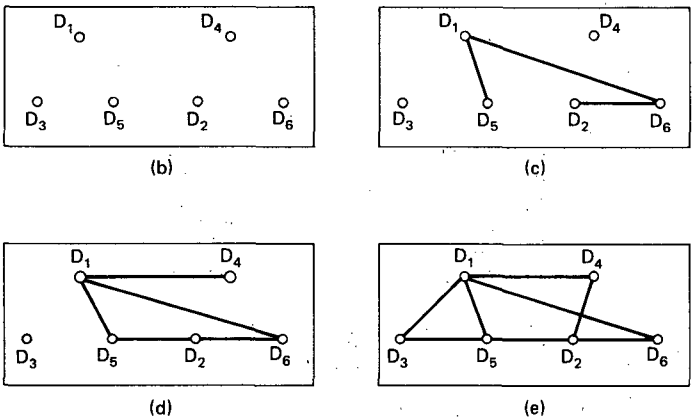

Figure 6-4 Role of threshold in item similarity for cluster formation. (a) Initial item-item similarity matrix. (b) Threshold T = 0.95. (c) Threshold T = 0.80. (d) Threshold T = 0.60. (e) Threshold T = 0.50.

ter a *cluster representative* or *centroid* to represent the class during the cluster manipulations. Comparisons between individual items and the cluster are then performed by computing the item-centroid similarity.

In principle, the centroid of a cluster might be represented by any document located in that cluster. In practice it may be preferable to construct a special centroid vector that is centrally located in the cluster. Typically, the centroid may be defined as the average of the documents in a cluster. That is, the weight of a centroid term is computed as the average of the weights for that term over all items in a cluster, or

$$\text{CTERM}_k = \frac{1}{m} \sum_{i=1}^{m} \text{TERM}_{ik} \tag{12}$$

where CTERM_k is the kth term in the centroid, TERM_{ik} is the kth term in the ith document in the cluster, and the cluster contains m documents in all.

A computation such as the one specified by expression (12) produces a centroid vector that is reasonably representative of the items in a cluster. The

most heavily weighted terms in the individual documents will be heavily weighted in the centroid. The sample centroid derivation shown in Table 6-3 demonstrates that the two terms 3 and 6 best represent the clustered items. These two terms also receive the highest weight in the centroid. The example of Table 6-3 also shows that the standard centroid computation leads to full centroid vectors with few zero terms even when the individual document vectors are sparse (that is, consist mostly of 0 terms). To improve the storage efficiency of the centroid vectors, it is customary to add a thresholding operation which eliminates centroid terms of excessively low weight. The reduced centroid shown in Table 6-3 is obtained by eliminating centroid terms with weights smaller than 1.

The centroid averaging process suggested in expression (12) may overemphasize the importance of some terms that receive excessively high weights. In some systems the final centroid term weights are therefore determined by using *rank values* rather than actual weights. For example, a weight of 1 can be assigned to the term exhibiting the lowest reduced weight; the next lowest term is given a weight of 2, and so on up until all nonzero terms are accounted for. The rank value process produces term weights of 1 and 2 for terms 6 and 3 of Table 6-3, respectively, replacing the original weights of 3 and 4.

Additional cluster centroid definitions may be used to produce binary centroid vectors with centroid weights restricted to 0 and 1 only. Alternatively, the term weights may be normalized to lie between 0 and 1 by dividing each term weight by the length of the centroid vector [1,2,13]

$$\text{CLENGTH} = \sqrt{\sum_{i=1}^{t} (\text{CTERM}_i)^2}$$

Many cluster generation systems, including the method described in Chapter 4 in connection with the SMART system description, produce a hierarchical cluster tree where large clusters are analyzed and broken down into a number of smaller clusters, which are themselves broken down into still smaller clusters, until finally the lowest level clusters are (figuratively) broken down into individual documents. A typical hierarchical cluster arrangement is shown in Fig. 6-5.

Table 6-3 Cluster Centroid Formation for Three Documents

DOCUMENT_1	= (1, 0, 3, 0, 0, 4, 0, 0, 0, 0)
DOCUMENT_2	= (0, 0, 2, 0, 0, 3, 1, 1, 0, 0)
DOCUMENT_3	= (0, 1, 7, 0, 1, 2, 0, 0, 1, 1)
Standard centroid	= (⅓, ⅓, 4, 0, ⅓, 3, ⅓, ⅓, ⅓, ⅓)
Reduced centroid	= (0, 0, 4, 0, 0, 3, 0, 0, 0, 0)
Centroid using rank values	= (0, 0, 2, 0, 0, 1, 0, 0, 0, 0)

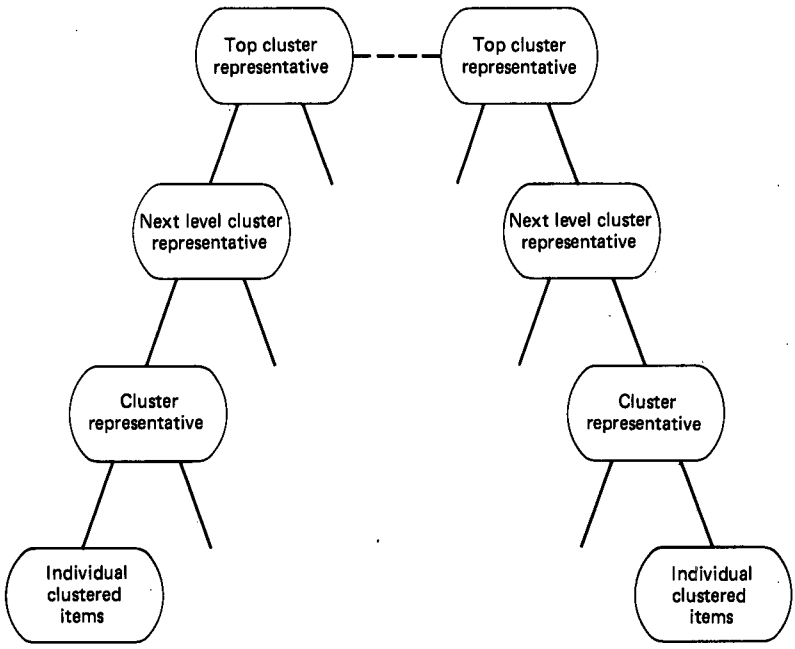

Figure 6-5 Hierarchical cluster arrangement.

A cluster tree such as that of Fig. 6-5 can be searched in many different ways. Intuitively the simplest method consists in proceeding from the top downward in the cluster tree, by comparing the incoming query first with the centroids of the topmost clusters, and continuing downward by next examining the next level of centroids for the clusters that were sufficiently similar to the query. This step is repeated by considering successively smaller clusters until eventually the individual documents in some of the lowest-level clusters are compared with the query. This is a *top-down* cluster search strategy based on the availability of auxiliary files containing all cluster centroids.

An alternative *bottom-up* search strategy may be introduced that uses only the clusters on the next to lowest level of the cluster tree. In that strategy, the incoming queries are compared with the centroids of the lowest-level clusters, and the individual documents included in certain lowest-level clusters are then considered [2,5,14]. The bottom-up search strategy reduces the storage cost for the tree of centroid vectors and may lead to more rapid identification of documents actually relevant to a given query than the top-down search. Furthermore, the likelihood of going down the wrong cluster path and winding up with useless information is reduced when the comparisons with the top centroids are eliminated. Since the number of low-level clusters may, however, be very large, an auxiliary index may be required to gain access to the low-level classes. This index may be conveniently implemented as an inverted file for all low-level centroid terms. A typical search then proceeds by using the individ-

ual query terms to access the inverted file of low-level cluster centroids. This leads to the identification of one or more low-level clusters, and these in turn lead to the retrieval of individual documents located in these classes.

Additional details for cluster generation and searching are including in the remainder of this section.

*B Classification Methods

One method for clustering a collection of objects uses similarity (or distance) measurements between each object and each other object in the collection. The objects are grouped into classes when they exhibit sufficiently large similarities or small distances. This process was illustrated earlier in Fig. 6-4, where an item-item similarity matrix was used to generate several cluster arrangements depending on the threshold used.

The clustering methods based on the availability of item-item similarities lead to a variety of *graph-theoretical* grouping strategies. Normally, rules are imposed on the graph obtained by transforming the item-item similarity matrix into a graph as shown in the example of Fig. 6-4. The following restrictions could be imposed on the graph representing the object similarities:

1 The clusters are defined as the *cliques* of the graph, that is, as the groups (maximal subsets) of items for which each item is connected to each other item in the group.

2 The clusters could also be *strings* of items such that item a is connected to b, and b is further connected to c, and so on until no further connected item can be added to the string.

3 The clusters might also be defined as *stars* where some central item is connected to all other items in the group.

4 Finally, the clusters could be *connected components* where an arbitrary path is found between each item in the component and each other item; the *single-link* method described in Chapter 4 is based on the use of connected components.

Consider, as an example, the connection graph of Fig. 6-4e obtained from the item-item similarity matrix of Fig. 6-4a using a similarity threshold of 0.50. The following classes are obtained from Fig. 6-4e for the graph specifications given earlier:

1 For the clique restriction, one finds one clique with three items and five additional cliques each with two items (D_1,D_3,D_5), (D_1,D_4), (D_1, D_6), (D_2, D_5), (D_2,D_6), (D_2,D_4).

2 Several subdivisions into strings are possible for the graph of Fig. 6-4e. Assuming that the strings to be produced are nonoverlapping, the following two strings will cover the graph: (D_3,D_1,D_4) and (D_5,D_2,D_6); another string subdivision would be (D_3,D_1,D_4,D_2,D_5) plus the single-item string (D_6).

3 Various nonoverlapping stars can be generated from Fig. 6-4e such as (D_1,D_3,D_5,D_4) and (D_2,D_6).

4 Finally for the connected component strategy, a single cluster is obtained consisting of all six documents under consideration.

The example of Fig. 6-4 illustrates that even for the graphical clustering methods that exhibit all the theoretically desirable characteristics such as being stable and well-defined, many different cluster structures can be constructed depending on various parameter settings and clustering requirements. The connection pattern between objects is uniquely determined by the threshold chosen for the item-item similarities. However, the final cluster arrangement also depends on the graph representation used to define the clusters, and on additional rules that may be specified concerning the permitted size and overlap of the clusters (the number of items included in a cluster and the number of items jointly assigned to more than one cluster).

The graph-theoretical clustering algorithms tend to be expensive to implement when the number of objects to be grouped is large. In fact, the construction of the item-item similarity matrix alone requires a comparison of each item with each other item, that is, the computation of about $n^2/2$ similarity coefficients for n items. A number of *heuristic clustering* methods have therefore been developed for which the construction of an item similarity matrix is not required.

Most of the heuristic methods are based on the prior availability and subsequent use and refinement of an initial set of clusters. In some cases, the problem under consideration may simply require that a set of new or unknown objects be added to an already existing cluster structure. Alternatively, when initial clusters are not available, some easily identified property of the objects might be used to define an initial set of trial clusters—for example, all objects that have a given property or term in common could be entered into a common trial class. Another possibility consists in taking a small subset of the objects to be clustered and using these to construct a set of so-called core clusters. The remaining parts of the collection that are not originally handled are then subsequently assigned to the existing core clusters.

Assuming the existence of an initial set of clusters, the heuristic methods next proceed by constructing cluster representatives for the existing clusters. The clustering operation itself is again based on similarity measurements between the objects to be clustered and the already existing cluster centroids. A typical iterative cluster refinement process involves the following steps:

1 Compare each document D with all existing cluster centroids, and obtain similarity coefficients $SIM(D,C_k)$ for all clusters in the cluster structure.

2 Determine the cluster for which $SIM(D,C_k)$ is largest for each given document D and place D into that cluster (alternatively, if overlapping clusters are desired, place D into all clusters for which the corresponding centroid similarity is larger than some arbitrary threshold).

3 Recompute the cluster centroids based on the new document assignments, and return to step 1.

4 Stop the process after a fixed number of iterations through steps 1 to 3 or alternatively stop after all document movements from one class to another have ceased.

Such an iterative cluster reassignment process is normally less expensive to perform than the graph-theoretical process described earlier, because the similarity coefficients are not computed for all pairs of items but only for all items with respect to all cluster centroids. If one assumes that log n clusters exist for the n items, the number of centroid-document comparisons is of order n log n (instead of n^2 as before). However, the heuristic clustering methods are theoretically less satisfying than the graph-based methods because the final cluster structure obviously depends on the initially available cluster arrangement. This undesirable property also characterizes the *one-pass* clustering methods described in Chapter 4. In that case, the documents are processed sequentially one at a time, and each item potentially forms a new class if it does not exhibit sufficient similarities with already existing clusters. The order in which the items enter into the centroid comparison process will then determine the shape of the final clusters.

The two types of classification methodologies can be combined into a single process by using the less expensive n log n process (for example a simple one-pass clustering system) to generate an initial set of trial clusters, and reserving the expensive graph-theoretical method to refine and subdivide each trial cluster. Each trial cluster of size m (m being much smaller than the total collection size n) could then be broken down into several smaller clusters using a graph-theoretical method requiring on the order of m^2 vector comparisons. Such a hybrid clustering process could generate useful low-level clusters that are as effective as clusters produced by graph-type methods applied to the complete collection.

*C Cluster Search Evaluation

Because of the substantial expense involved in clustering large document collections, a thorough evaluation of cluster search methods has never been performed. Only fragmentary results are available obtained with a few typical clustering arrangements and a few sample document collections [14–17]. In principle, the tradeoff involved in using a clustered document collection for information retrieval purposes, as opposed, for example, to using a standard inverted file, is simple to describe: on the one hand, one may expect that improved recall and precision are obtainable from a search when related documents are collected into common classes because the retrieval of a particular document then automatically leads to the retrieval of additional related items; on the other hand, the clustered organization produces a good deal of systems overhead because of the cost of the clustering operation itself and the requirement to store an auxiliary file of cluster centroids. Furthermore, as will be seen, the comparison of incoming queries with the centroid structure may in

fact identify marginal search paths when the centroids appear to be similar to the queries but the corresponding clusters contain few relevant documents.

It was seen in the preceding section that many different methodologies are available to cluster a given document collection. Moreover, given a particular clustering arrangement the comparison of user search queries with the available cluster centroids and the identification of the relevant document clusters can also be carried out in many different ways. Thus given a cluster structure such as the one in Fig. 6-5, one can proceed from the top down, choosing at each level the most likely cluster centroid until eventually the documents in a single cluster on the lowest clustering level are reached. Such a search, known as depth-first, is likely to produce high precision searches that may, however, be deficient in recall. Higher recall may be obtained by following several parallel paths of the cluster structure, leading to the eventual identification of several low-level document clusters. Unfortunately no obvious search strategy exists which will necessarily lead to the retrieval of most useful materials.

One problem arising in cluster searching is the difficulty of obtaining an unequivocal result from a comparison between a user query and a cluster centroid. Normally, an attempt must be made given a particular user query to estimate the probability that a given cluster is a useful cluster. Such an estimate can be based on the frequency of occurrence (or on the weight) of the individual query terms in the corresponding cluster centroid. When the query terms collectively exhibit high occurrence frequencies in a cluster centroid, the presumption may be that the corresponding cluster is useful for retrieval purposes. Unfortunately, the higher level centroids normally represent broad subject classes and the query-centroid comparisons may not identify any particular centroid that is clearly preferable to the other centroids. Such a situation obviously complicates the cluster searching task.

As an example of the efficiencies attainable in cluster searching, consider some sample searches performed for several sets of search requests with a collection of 2,000 documents in computer science. A hierarchical cluster arrangement was generated for the document collection using the one-pass clustering system described in Chapter 4. The hierarchy consisted of four levels, illustrated in the scheme of Fig. 6-5, including 200 clusters on the lowest clustering level with an average of 10 documents each. This arrangement was used with four different search strategies [18]:

1 A standard inverted file search, using an inverted term directory for all terms occurring in the documents of the collection; the documents themselves are entered into the document file in arbitrary order.

2 A standard inverted file search for which the document file is stored in cluster order, that is, one access to the document file retrieves a cluster of related documents.

3 An inverted file constructed for the *centroids* on the lowest cluster level only, providing access to a number of low-level document clusters for each query term.

4 A top-down search of the full cluster hierarchy based on a stored file of all cluster centroids.

The auxiliary inverted term files and the cluster centroid files cannot normally be stored in internal machine memory. In order to carry out the required query-centroid comparisons, it is therefore necessary to transfer various portions of the centroid directory from external bulk storage to internal memory. Such transfers from one type of memory to another are usually carried out for fixed size blocks of data, known as *pages*. For the experiments under discussion, a page is defined as 4,096 characters of storage. A normal page transfer process would transfer into internal storage the particular data page containing the centroid needed at each particular time. To make room for a new page of data, an old page no longer in use must be transferred back to external storage from internal memory. The standard way for doing this is the "least recently used" (LRU) page replacement method which transfers out the page that has remained unused for the longest time.

When only a single page is brought into storage at any one time, a complete disk seek operation is needed for each page (that is, for each centroid) transferred into internal memory. Such a strategy appears ill suited to the processing of hierarchical tree structures, because the need for a particular centroid vector normally implies the simultaneous need for its descendants on a lower level of the tree or for its brothers on the same level of the tree. This suggests that a complete set of centroids ought to be brought into internal memory at the same time, preferably consisting of the originally needed centroid together with its immediate descendants and possibly its brothers. By using a so-called prefetch strategy, the pages containing the descendants or brothers of a given centroid can be prepared for transfer while the initial page is being sought. In these circumstances, a complete disk seek operation is still required for the initial page to be transferred. The pages containing descendant and brother centroids can, however, be transferred at much lower cost amounting to only about one-tenth the time required for a complete disk seek operation.

The file storage requirements for the several search systems are summarized in Table 6-4. The document file itself requires 71 pages of data. The

Table 6-4 Storage Requirements for Cluster Searches
(2,000 Documents in Computer Science; Page Size = 4,096 Bytes)

Standard search using inverted file for document terms	Inverted term index without weights	110	39 pages
	Document file		71 pages
Search using the low-level document clusters only	Inverted index to low-level centroids	84	13 pages
	Document file		71 pages
Full top-down search through cluster hierarchy	Full centroid file (400 terms per centroid)	146	75 pages
	Document file		71 pages

standard inverted term file (using as entries all terms occurring in the documents) uses 39 additional pages. Thirteen pages are needed to store an inverted index for the short centroids on the lowest level of the cluster hierarchy; finally, 75 pages are used to store the full hierarchy of cluster centroids. A cluster storage search using only the low-level centroids appears therefore to produce better storage efficiency than a standard inverted file search or a top-down search through the full store of cluster centroids.

The average search precision and the average number of page faults required to reach a given level of recall using the single-page replacement process are shown in Table 6-5 for three of the four search methods previously described [18]. The averages of Table 6-5 are computed for 23 short Boolean queries containing an average of 4.8 search terms per query. Next to each statistic, Table 6-5 also contains the percentage improvement or deterioration over the standard inverted file search method.

A valid comparison of the search runs can be made only if the total number of retrieved items over the 23 queries is approximately the same for all methods. Since no choice is possible for the number of items retrieved by a

Table 6-5 Short Boolean Queries—Precision and Page Faults
(23 Queries—Separate Relevance Assessments)

Relative recall	Inverted document terms (clustered)	Inverted low-level centroids		Tree search	
	a Average precision comparison				
0.1	0.900	0.406	−55%	0.277	−69%
0.2	0.888	0.406	−54%	0.277	−69%
0.3	0.847	0.403	−52%	0.255	−70%
0.4	0.845	0.393	−53%	0.212	−75%
0.5	0.838	0.388	−54%	0.203	−76%
0.6	0.830	0.385	−54%	0.192	−77%
0.7	0.828	0.356	−57%	0.183	−78%
0.8	0.824	0.346	−58%	0.162	−80%
0.9	0.817	0.329	−60%	0.149	−82%
1.0	0.815	0.329	−60%	0.141	−83%
			−55.7%		−75.9%
	b Average page fault comparison (no prefetch)				
0.1	4.89	3.00	−39%	14.90	+205%
0.2	5.54	3.50	−37%	14.90	+169%
0.3	5.87	4.20	−28%	14.85	+153%
0.4	7.44	4.20	−44%	16.55	+122%
0.5	8.53	4.45	−48%	16.43	+93%
0.6	10.26	4.50	−56%	17.83	+74%
0.7	11.21	4.50	−60%	18.96	+69%
0.8	12.21	4.65	−62%	21.17	+73%
0.9	12.94	4.65	−64%	21.83	+69%
1.0	12.94	4.95	−62%	21.83	+69%
			−50%		+109.6%

conventional Boolean search, that number must necessarily be used as a standard for all methods. For the query collection used in the experiments, a total of 341 documents were retrieved by the conventional inverted file method over the 23 queries. The threshold used for the clustered searches was therefore chosen to retrieve approximately 15 documents per query ($15 \cdot 23 = 345$).

Separate relevance assessments were available of each document with respect to each query (that is, the inverted file searches were not automatically assumed to produce perfect search output). Furthermore, the recall level exhibited in Table 6-5 is computed separately for each search method rather than globally. That is, relative rather than absolute recall levels are shown, defined as the proportion of relevant items retrieved at a given level out of the total relevant that can be retrieved by each particular method.

The results of Table 6-5 indicate that the full cluster tree search is not competitive with the conventional inverted file process either in precision or in the page fault rate when short Boolean queries are processed. In that case only a few lists of document references must be processed in the inverted file system, corresponding to the few terms included in each query. The cluster tree must, however, always be traversed from top to bottom. The low-level cluster inversion is much more attractive in that case, since only about half as many page faults are needed as in the inverted file process. Both cluster search systems do, however, produce substantially lower precision than the inverted file system.

The amount of work required by an inverted file search increases as the query length and to some extent the number of documents to be retrieved increase. The experimental conditions of Table 6-5 thus favor the inverted file technology. When longer, vector-type queries are processed, the evaluation results become very different. Table 6-6 contains both average precision and average page fault rates averaged over 33 long queries (16.8 terms per query) using page replacement methods with and without prefetch [18].

The results in Table 6-6a show that the standard inverted file search again produces the best search precision. However, the cluster searches are now more competitive than before: the tree search shows a deterioration of 15 percent in precision; for the inverted low-level centroids the precision loss is 19 percent, compared with the earlier losses of 76 and 56 percent, respectively. The page fault rates which apply to the cluster searches conducted for the long queries are much improved over the equivalent runs using the shorter queries. Even with the unfavorable page-at-a-time (no prefetch) page replacement method, the full tree search is nearly competitive with the inverted file list processing. Table 6-6c shows that with a prefetch page replacement, the tree search is 30 percent better than the document inversion.

The results of Table 6-6 indicate that for the long, vector-type queries, a cluster search method seems preferable to the conventional inverted list manipulations. Even more clear-cut results in favor of the top-down cluster search are obtained when a larger number of documents are retrieved in each search than the 15 used in the experiments of Tables 6-5 and 6-6 [18].

Table 6-6 Long Weighted Vector-Type Queries—Precision and Page Faults
(33 Queries; 15 Retrieved Items per Query)

Relative recall	Inverted document terms: Clustered storage	Inverted document terms: Random storage		Inverted low-level centroids		Complete tree search	
a Average precision comparison							
0.1		0.893		0.783	−12%	0.837	−6%
0.2		0.871		0.746	−14%	0.775	−11%
0.3		0.844		0.702	−17%	0.748	−11%
0.4		0.835		0.679	−19%	0.734	−12%
0.5		0.827		0.664	−20%	0.721	−13%
0.6		0.817		0.656	−20%	0.682	−17%
0.7		0.812		0.623	−23%	0.670	−17%
0.8		0.803		0.612	−24%	0.639	−20%
0.9		0.784		0.610	−22%	0.612	−22%
1.0		0.757		0.582	−23%	0.607	−20%
					−19.4%		−14.9%
b Average page fault comparison (no prefetch)							
0.1	14.8	15.2	+2%	18.1	+23%	14.9	+1%
0.2	15.7	16.1	+2%	19.4	+23%	17.4	+11%
0.3	17.5	18.2	+4%	22.9	+31%	18.2	+4%
0.4	18.7	19.4	+4%	24.4	+30%	19.2	+3%
0.5	19.9	20.6	+4%	25.9	+30%	21.0	+6%
0.6	21.1	21.8	+4%	28.2	+34%	23.2	+10%
0.7	22.1	23.2	+5%	31.7	+44%	24.5	+11%
0.8	23.2	24.3	+5%	34.7	+50%	26.8	+16%
0.9	24.5	26.0	+6%	36.7	+50%	29.5	+20%
1.0	25.5	27.4	+9%	37.9	+49%	31.8	+25%
			+4.5%		+36.4%		+10.7%
c Page fault comparison (with prefetch)							
0.1	10.12	10.55	+4%	12.97	+28%	6.44	−36%
0.2	11.00	11.47	+4%	14.02	+27%	7.62	−31%
0.3	12.54	13.29	+6%	17.09	+36%	8.16	−35%
0.4	13.58	14.48	+7%	18.23	+34%	8.76	−35%
0.5	14.80	15.66	+6%	19.59	+32%	9.81	−34%
0.6	15.83	16.84	+6%	21.36	+35%	11.06	−30%
0.7	16.90	18.18	+6%	24.01	+42%	11.87	−30%
0.8	17.87	19.19	+7%	26.28	+47%	13.17	−26%
0.9	19.12	20.78	+9%	27.99	+46%	14.79	−23%
1.0	20.08	22.05	+10%	28.80	+43%	16.06	−20%
			+6.7%		+37%		−30%

**D Automatic Pseudoclassification

It was mentioned earlier that automatic classification techniques can be used to construct affinity classes for either documents or terms. In the former case, the principal aim is to impose structure on the search system in the hope of gaining search efficiencies during the comparison between user queries and stored doc-

uments. When *terms* are clustered, the result is a term class arrangement, or thesaurus, that can be used for the construction of expanded search requests and document descriptions by inclusion in the vectors of synonyms and other terms related to those originally available. It is generally accepted that the use of a thesaurus in the indexing and search formulation process can enhance retrieval effectiveness [19,20].

The use of thesauruses in retrieval was briefly illustrated in Chapter 3 as part of the discussion on automatic indexing. Thesauruses are normally constructed manually by subject experts, although automatic classification procedures are usable in principle based, for example, on use of similarity information for term pairs. The similarity between terms might be based on the frequency distribution characteristics of the terms across the documents of a collection. Thus, if two or more terms co-occur in many documents of a given collection, the presumption is that these terms are related in some sense and hence can be included in common term classes. Alternatively, when term distribution information is not available, term classifications can be automatically constructed by adapting an existing document classification and assuming that terms which occur jointly in the document classes could be used to form the desired term classes [21–26].

Even though considerable effort has been devoted to the development of automatic thesaurus construction methods, a really viable thesaurus generation process is still lacking. One problem is the difficulty of identifying representative document collections which could guarantee that the resulting term distribution characteristics would be applicable to other collection environments. Unfortunately, there is no obvious way for obtaining such representative document samples, short of using large comprehensive collections. However, large collections cannot be processed at a reasonable cost. In some studies, term classes have been generated from locally defined, small subcollections of documents, such as the set of documents retrieved in response to a given search request [27]. Whether such procedures will prove generally viable remains to be seen.

Since document retrieval environments normally operate in an on-line mode, permitting users to communicate directly with the retrieval system, the question arises whether information obtained from the user population during the retrieval effort might be used as an aid in constructing term classifications. Such a consideration forms the basis for the so-called pseudoclassification process where the normal use of term frequency information and term probability distributions is replaced by user relevance information of documents with respect to search requests. Since the term classification obtained by a pseudoclassification process depends on particular user queries and specified subject areas, the term classes that are obtained may be applied to other documents in related subjects only if the input data are comprehensive enough to make them applicable to other user groups in different collection environments. The pseudoclassification process is based on the existence of four distinct types of information: (1) a collection of documents in a given subject area, (2) a set of infor-

mation requests addressed to that collection, (3) a retrieval threshold T which leads to the retrieval of a given document in answer to a particular query whenever a similarity measure between the document and query exceeds T, and (4) a set of user relevance assessments specifying the relevance of individual documents with respect to the available queries [28].

Consider, in particular, a given collection of n documents and m queries in a topic area. The previously specified information can be used to construct two binary n by m matrices specifying the retrieval status and the relevance properties, respectively, for each document-query pair (DOC_i,$QUERY_j$). Specifically, let VAL and REL represent the *retrieval* and the *relevance* matrices for a given collection of documents and queries. Thus VAL_{ij} may be defined to be equal to 1 whenever DOC_i is retrieved in response to $QUERY_j$, that is, whenever $SIM(DOC_i,QUERY_j) > T$; otherwise VAL_{ij} is set equal to 0. Correspondingly REL_{ij} is defined as 1 when DOC_i is specified as relevant to the $QUERY_j$ and as 0 otherwise. The relevance and retrieval matrices are represented in Fig. 6-6 for typical document and query collections.

Consider now the results of a retrieval operation. It is obvious from the definition of the VAL and REL matrices that four different situations can arise for a given document-query pair (DOC_i, $QUERY_j$):

1 Both VAL_{ij} and REL_{ij} are defined equal to 1, implying that the relevant document DOC_j is retrieved in response to $QUERY_j$.

2 Both VAL_{ij} and REL_{ij} are defined equal to 0, implying that the nonrelevant document DOC_i is rejected in response to $QUERY_j$.

$$VAL = \begin{array}{c|cccc} & QUERY_1 & QUERY_2 & \dots & QUERY_m \\ \hline DOC_1 & VAL_{11} & VAL_{12} & \dots & VAL_{1m} \\ DOC_2 & VAL_{21} & VAL_{22} & \dots & VAL_{2m} \\ \vdots & & & & \\ DOC_n & VAL_{n1} & VAL_{n2} & \dots & VAL_{nm} \end{array}$$

(a)

$$REL = \begin{array}{c|cccc} & QUERY_1 & QUERY_2 & \dots & QUERY_m \\ \hline DOC_1 & REL_{11} & REL_{12} & \dots & REL_{1m} \\ DOC_2 & REL_{21} & REL_{22} & \dots & REL_{2m} \\ \vdots & & & & \\ DOC_n & REL_{n1} & REL_{n2} & \dots & REL_{nm} \end{array}$$

(b)

Figure 6-6 Retrieval and relevance matrices. (a) Retrieval matrix VAL. (VAL_{ij} implies that DOC_i is retrieved in response to $QUERY_j$.) (b) Relevance matrix REL. (REL_{ij} implies that DOC_i is judged relevant to $QUERY_j$.)

3 and **4** $VAL_{ij} = 1$ and $REL_{ij} =$ or vice versa $VAL_{ij} = 0$ and $REL_{ij} = 1$, implying either that a nonrelevant document is retrieved or that a relevant item is rejected.

From the viewpoint of retrieval effectiveness, cases 1 and 2 represent a favorable outcome, whereas 3 and 4 constitute system failures. A perfect system will produce VAL and REL matrices that are identical, implying that all $(DOC_i, QUERY_j)$ pairs fall into classes 1 or 2, being either correctly retrieved or correctly rejected. Correspondingly, the number of $(DOC_i, QUERY_j)$ pairs falling into classes 3 and 4 may be used as a measure of system inadequacy or failure.

To improve the operations of a retrieval system it is necessary to introduce methodologies capable of shifting some of the $(DOC_i, QUERY_j)$ pairs from classes 3 and 4 to classes 1 or 2, while ensuring at the same time that $(DOC_i, QUERY_j)$ pairs that are already correctly classified are left undisturbed. Two different situations may arise:

3 When a document is incorrectly retrieved ($VAL_{ij} = 1$ and $REL_{ij} = 0$), it becomes necessary to decrease the similarity coefficient SIM with the query so that eventually $SIM(DOC_i, QUERY_j) \leq T$, thus ensuring the rejection of DOC_i in response to $QUERY_j$.

4 On the other hand, for documents that are incorrectly rejected ($VAL_{ij} = 0$ and $REL_{ij} = 1$), the similarity coefficient with the query must be correspondingly increased to achieve $SIM(DOC_i, QUERY_j) > T$, causing the document to be retrieved.

The size of the similarity coefficients for query-document pairs can be altered by introducing refined automatic indexing methods designed to change the original term assignment to queries and documents. Alternatively, sophisticated term weighting strategies could be introduced. A third possibility consists in utilizing term relationship information. This last strategy forms the basis for the pseudoclassification process. Consider in particular a given term classification or thesaurus, in which p different classes are used to group the terms into affinity or similarity classes. The term classification is represented in matrix form in Fig. 6-7. The document and query vectors can now be represented by lengthened constructs as follows

$$DOC_i = (d_{i1}, d_{i2}, \ldots, d_{it}, c_{i1}, c_{i2}, \ldots, c_{ip})$$
$$QUERY_j = (q_{j1}, q_{j2}, \ldots, q_{jt}, c_{j1}, c_{j2}, \ldots, c_{jp})$$

where d_{ik} represents the assignment of term k to document i and q_{jk} represents the assignment of term k to query j. The elements c_{ik} and c_{jk} represent the term class assignments to the respective vectors. In each case c_{ik} (or c_{jk}) is set equal to 1 whenever class k pertains to document i (or to query j). To increase the size of the similarity measure between a given document-query

GROUP =	$CLASS_1$	$CLASS_2$	...	$CLASS_p$
$TERM_1$	1	0	...	1
$TERM_2$	0	1	...	1
.				
.				
.				
$TERM_t$	1	0	...	0

Figure 6-7 Classification matrix GROUP. ($GROUP_{ij} = 1$ implies that $TERM_i$ occurs in $CLASS_j$.)

pair, it suffices to place into a common term class two or more terms that did not originally produce a term match in the respective vectors. This operation adds one or more common class identifiers to both vectors, thereby increasing the similarity between them. Analogously, the similarity coefficients can be decreased by appropriately reducing the number of matching term classes.

Consider, as an example, two typical vectors consisting of six terms each (where a 0 entry is used to denote a term that is absent):

$$DOC_i = (0, 0, \text{lift, propeller, roll, wing})$$
$$QUERY_j = (\text{aileron, drag, 0, propeller, 0, wing})$$

In their original form, two terms (propeller and wing) are present in both vectors. If the document is now assumed to be incorrectly rejected ($REL_{ij} = 1$ and $VAL_{ij} = 0$), the size of the matching coefficient can be increased by placing into a term class two or more initially nonmatching terms. Specifically for the two sample vectors, aileron and lift could be placed into a common class k (or alternatively, drag and lift, or roll and aileron, or roll and drag); in each case, the identifier for class k would be added to *both* vectors thereby increasing the similarity between them. For example, placing terms aileron and wing into class k adds the indicator for class k to the DOC_i vector because of the presence of lift in DOC_i, and the class k indicator to $QUERY_j$ because of the presence of aileron in the original query vector. This operation on the term classification produces the following lengthened vectors exhibiting one additional matching identifier:

$$DOC_i = (0, 0, \text{lift, propeller, roll, wing, ``class k''})$$
$$QUERY_j = (\text{aileron, drag, 0, propeller, 0, wing, ``class k''})$$

When a document is incorrectly retrieved ($VAL_{ij} = 1$ and $REL_{ij} = 0$), the reverse operation is performed and certain terms are removed from the thesaurus classes so as to decrease the number of matching classes assigned to both vectors.

The assignment of terms to classes (or the eventual removal of terms from classes) presents no difficulty in principle when the number of document-term

pairs is small and when a large number of thesaurus classes can be introduced. Unfortunately, in practice it is necessary to operate with a large number of (DOC,QUERY) pairs and a restricted number of term classes (small p), and the task consists in constructing the best possible term classification which places a maximum number of (DOC,QUERY) pairs into the correctly retrieved or correctly rejected groups.

Since a given (DOC,QUERY) pair does not occur in isolation in the retrieval system, an operation on the term classification undertaken to help retrieve (or reject) a given document may have unfortunate effects on other documents in the collection. Thus, a given operation such as placement of $TERM_i$ and $TERM_j$ into class k may properly raise the similarity measure for a particular (DOC,QUERY) pair and shift that pair from class 4 to class 1; at the same time that same operation may carry a large number of (DOC,QUERY) pairs out of the 1 and 2 classes and place them into the undesirable 3 or 4 classes. Hence it becomes necessary to check the applicability of a proposed thesaurus operation by determining its effect on all (DOC,QUERY) pairs in the system, and to carry out a given operation only when its overall effect is judged to be beneficial.

The standard pseudoclassification process which is designed to produce an *optimal* term classification in which the maximum possible number of (DOC,QUERY) pairs *satisfy assessment* (that is, belong to classes 1 and 2) is then burdened with a number of substantial disadvantages:

1 To prove the optimality of a given term classification for a particular query and document collection, it is necessary to evaluate each perturbation of the classification by checking its effect on all other (DOC,QUERY) pairs in the system; thus each classification step necessarily involves all (DOC,QUERY) pairs.

2 The sequence in which the individual (DOC,QUERY) pairs are considered and the structure of the term classification available at the beginning of the process may complicate the classification process.

3 A given thesaurus operation may have to be considered many times in a given pass through the (DOC,QUERY) pairs, because although undesirable at a given moment in time, that same operation may be useful at a later time after additional changes have been introduced in the classification.

A variety of heuristic procedures have thus been proposed for the construction of acceptable term classifications. These methods differ from the previously described situation in that convergence of the procedures can normally be proved, but not optimality of the result.

In the standard pseudoclassification method the changes made to the term classification may lead to oscillations because a change once introduced may later have to be reversed. Alternatively, sequences of classification changes may be developed which are repeated indefinitely, so that the process runs forever. To avoid this possibility, on-the-spot global decisions can be introduced to ascertain whether a given operation on the classification should be carried

out at any given time. The idea is to process the (DOC,QUERY) pairs in sequence and to make the applicability of a given thesaurus operation dependent only on the (DOC,QUERY) pairs previously processed in the current pass. Furthermore, the whole process must be guaranteed to terminate. The following two strategies have been shown to be effective in practice [29,30]:

1 A proposed change in the term classification is accepted if no *essential* deterioration is produced for any other previously processed document-query pair. By essential deterioration is meant that the document-query similarity for a relevant document currently retrievable is not decreased so far as to render the item nonretrievable; similarly, the correlation measure for a nonrelevant item currently rejected is not increased so much that the item is eventually retrieved.

2 A proposed change in the term classification is accepted if the number of document-query pairs that exhibit improvements as a result of the change exceeds the number of pairs exhibiting deteriorations. Improving means increasing the similarity for relevant documents, and vice versa for nonrelevant items. Deterioration is the reverse of improvement. This strategy uses a global condition in which correctly retrieved or rejected pairs are allowed to cross the retrieval threshold provided only that overall the gains exceed the losses.

Several small-scale evaluations of these strategies were carried out with experimental document collections. In each case, the first strategy appears most effective in increasing the number of (DOC,QUERY) pairs that satisfy the relevance assessment (correctly retrieved or correctly rejected pairs), while the second strategy should be used in order to obtain the best recall and precision results.

The normal pseudoclassification process builds a complete term classification matrix which specifies for each term one or more classes to which the term may be assigned during the course of the operations. However, the term classes themselves are never used for retrieval purposes. Instead relations between *term pairs* are used by adding special identifiers to the document and query vectors whenever a particular term pair is included in a common class. This suggests that the construction of a full-term classification could be replaced by a simple determination of term pair relationships followed by the incorporation of term pair relationship indicators (instead of term class indicators) in the query-document similarity computations. Furthermore, the determination of the relationships between term pairs might depend on the *global occurrence characteristics* of each particular term pair in the documents and queries of a collection. Such considerations have led to the construction of term-term similarity matrices used for pseudoclassification purposes in which the term relationships are derived from the term occurrence frequencies in the documents of a collection [31–32].

Consider, in particular, the collection of term pairs ($TERM_k$,$TERM_h$) obtainable from a collection of queries and documents with the property that one of the terms in each pair occurs only in a given DOC_i while the other occurs

only in $QUERY_j$ for a given $(DOC_i, QUERY_j)$ pair. Alternatively, one of the two terms might be assigned to both DOC_i and $QUERY_j$, while the other is assigned to either query or document alone. Such a term pair is called an *acceptable* pair. Clearly a full term match does not exist for an acceptable term pair when a standard vector matching process is used, because the assignment of the two terms differs for the components of a given $(DOC_i, QUERY_j)$ pair.

If one assumes that the acceptable term pairs occur principally in document-query pairs that already satisfy assessment, that is $(DOC_i, QUERY_j)$ pairs that are either correctly retrieved or correctly rejected, then no particular action is warranted. On the other hand, if the term pairs occur mainly in $(DOC_i, QUERY_j)$ pairs for which DOC_i is relevant to $QUERY_j$ ($REL_{ij} = 1$) but the document is not retrieved ($VAL_{ij} = 0$), then one might conjecture that a positive semantic relationship exists between the respective terms and that this term relationship has not in fact been taken into account in the query-document match. Analogously, if the term pairs occur in query-document situations where the documents are retrieved even though they are not identified as relevant ($REL_{ij} = 0$ but $VAL_{ij} = 1$), then a negative semantic relationship may be assumed to exist between the corresponding terms.

This suggests the following term matching strategy. For each acceptable term pair $(TERM_k, TERM_h)$ a term relationship factor is computed as a *direct* function of the number of document query pairs containing the term pair for which $REL_{ij} = 1$ and $VAL_{ij} = 0$, and as an *inverse* function of the number of document-query pairs containing the terms such that $REL_{ij} = 0$ and $VAL_{ij} = 1$. The term relationship factor may then be incorporated into the query-document matching process in an attempt to increase the similarity coefficients for documents that are incorrectly retrieved or incorrectly rejected.

The term relationship factors can be computed in a single pass through the $(DOC_i, QUERY_j)$ pairs to construct a *term-term relationship* (TTR) matrix of dimension t by t containing for each acceptable term pair $(TERM_k, TERM_h)$ the corresponding number of $(DOC_i, QUERY_j)$ pairs for which $REL_{ij} = 1$ and $VAL_{ij} = 0$ as well as the number of $(DOC_i, QUERY_j)$ pairs for which $REL_{ij} = 0$ and $VAL_{ij} = 1$. In particular, for a given $(TERM_k, TERM_h)$ pair $(k > h)$, the below diagonal (k,h)th element of the term-term relationship matrix may be used to store the positive count, Pos(k,h), of term-pair occurrences in $(DOC_i\ QUERY_j)$ pairs for which the term pair is acceptable while the document is incorrectly rejected. Analogously, the above diagonal (h,k)th matrix element stores the negative count, Neg(k,h), of term pair occurrences in $(DOC_i, QUERY_j)$ pairs where the document is incorrectly retrieved. An appropriate term relationship function T(k,h) for terms $TERM_k$ and $TERM_h$ might be defined as

$$T(k,h) = \frac{Pos(k,h) - a\ Neg(k,h)}{Pos(k,h) + a\ Neg(k,h)}$$

for some constant a.

	Aileron	Drag	Lift	Propeller	Roll	Wing
$QUERY_k$	0	1	0	1	0	1
DOC_i	1	1	1	0	0	1
DOC_j	1	0	0	1	1	1

(a)

$(DOC_i, QUERY_k)$: $REL_{ik} = 1$, $VAL_{ik} = 0$
$(DOC_j, QUERY_k)$: $REL_{jk} = 0$, $VAL_{jk} = 1$

(b)

$(DOC_i, QUERY_k)$ { (aileron, drag), (aileron, propeller), (aileron, wing), (drag, lift) (drag, propeller), (lift, propeller), (lift, wing), (propeller, wing) }

$(DOC_j, QUERY_k)$ { (aileron, drag), (aileron, propeller), (aileron, wing), (drag, propeller) (drag, roll), (drag, wing), (propeller, roll) (roll, wing) }

(c)

TTR =

	Aileron	Drag	Lift	Propeller	Roll	Wing
Aileron		1		1		1
Drag	1			1	1	1
Lift		1				
Propeller	1	1	1		1	
Roll						1
Wing	1		1	1		

Counts for incorrectly retrieved items

Counts for incorrectly rejected items

(d)

Term relationship formula	$T(k, h) = \frac{Pos(k, h) - Neg(k, h)}{Pos(k, h) + Neg(k, h)}$	
Null relationship	*Positive relationship*	*Negative relationship*
T(aileron, drag) = 0	T(drag, lift) = 1	T(drag, roll) = -1
T(aileron, propeller) = 0	T(lift, propeller) = 1	T(drag, wing) = -1
T(aileron, wing) = 0	T(lift, wing) = 1	T(propeller, roll) = -1
T(drag, propeller) = 0	T(propeller, wing) = 1	T(roll, wing) = -1

(e)

Figure 6-8 Term relationship computation. (a) Original term vectors. (b) Relevance characteristics. (c) Acceptable term pairs. (d) Term-term relationship matrix. (e) Term relationship factors.

The single-pass construction of the term-term relationship matrix consists in processing each (DOC_i,$QUERY_j$) pair in turn, identifying the acceptable term pairs, and increasing the count stored in the (k,h)th or (h,k)th matrix elements depending on whether the term pair ($TERM_k$,$TERM_h$) is associated with an incorrectly rejected or an incorrectly retrieved document-query pair [31–32]. An example is given in Fig. 6-8 for two document vectors DOC_i, DOC_j and query $QUERY_k$, where DOC_i is assumed incorrectly rejected and DOC_j incorrectly retrieved. The acceptable term pairs are listed in Fig. 6-8c. The corresponding term-term relationship matrix is given in Fig. 6-8d and the term relationship computations are included in Fig. 6-8e.

It remains to define a vector similarity function incorporating the normal similarities between the term vectors DOC_i and $QUERY_j$, as well as the added term relationships derived from the term-term relationship matrix. Let $SIM(DOC_i,QUERY_j)$ represent the matching function used to compare the DOC_i and $QUERY_j$ vectors, and let $SIM_1(TERM_k,TERM_h)$ and $SIM_2(TERM_k,TERM_h)$ represent similarity functions based on the occurrence counts of positively [$T(k,h) > 0$] and negatively [$T(k,h) < 0$] related term pairs, respectively. Then an appropriate composite similarity function could be defined as

$$\begin{aligned} NEWSIM(DOC_i,QUERY_j) = {} & a_0 SIM(DOC_i,QUERY_j) \\ & + \sum_{k,h} a_1 SIM_1(TERM_k,TERM_h) \\ & + \sum_{k,h} a_2 SIM_2(TERM_k,TERM_h) \end{aligned} \tag{13}$$

for properly chosen constants a_0, a_1, and a_2, similarity functions SIM, SIM_1, and SIM_2, and acceptable term pairs ($TERM_k$,$TERM_h$) contained in the appropriate query-document vectors. The summation signs preceding the second and third terms of expression (13) indicate that the term pair similarity factors need to be summed for all acceptable ($TERM_k$,$TERM_h$) pairs contained in a given pair of query-document vectors.

Experimental evaluation results available for small document collections of several hundred documents and several dozen queries show improvements for the composite similarity function NEWSIM over the standard cosine similarity measure ranging from 10 percent in the precision values to over 30 percent in precision for fixed recall levels [32].

5 DYNAMIC QUERY ADJUSTMENT

A General Considerations

Most operational retrieval systems offer an interactive search environment in which users, or search intermediaries, communicate directly with the search system and responses are furnished more or less instantaneously following submission of the search requests. Two kinds of interactive manipulations may be

carried out during the course of an information search: the first relates to the submission of queries (search terms) and to the viewing of displayed responses; the other is the construction of the query formulations, sometimes aided by displays of thesaurus contents.

In some of the more sophisticated on-line systems it is possible to list for a given term all alphabetically related terms, or terms included in the same thesaurus category as the original term. In constructing an effective query statement, information about the number of items retrieved in response to earlier query formulations (known as the number of "hits") could then be useful; alternatively, the display of portions of retrieved documents or of documents related to those retrieved earlier can also be used for query reformulation purposes. Typically, the number of documents retrieved furnishes some idea about the specificity of the search terms used in the current query formulation: the terms can then be either broadened if the aim is to increase the number of retrieved items, or they can be narrowed in the opposite case. The display of the related search vocabulary or of previously retrieved document texts immediately suggests additional or alternative query formulation possibilities.

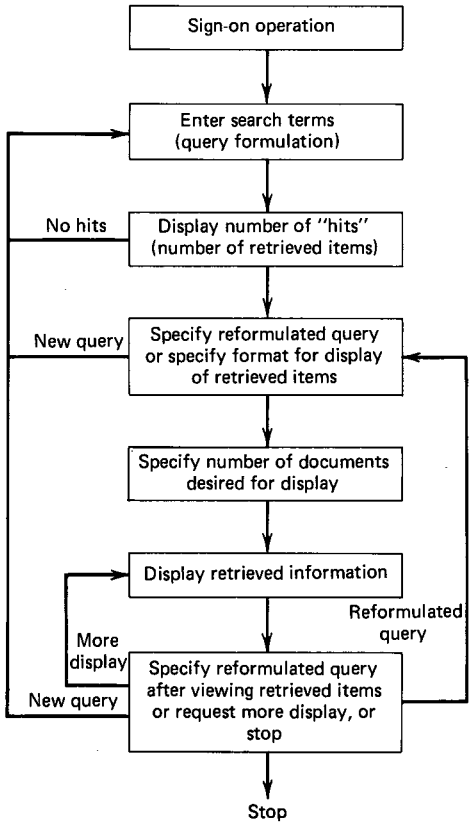

Figure 6-9 Typical interactive retrieval sequence.

Currently available evidence indicates that the query reformulation process is easier to manage by the user when the feedback information utilized for the formulation of query statements consists of previously retrieved document excerpts or surrogates rather than of vocabulary displays [33–36]. This reflects the fact that the display of retrieved documents furnishes indications of how the system has interpreted the user queries, but vocabulary displays do not. However, vocabulary displays can be produced inexpensively before performing any searches at all, but document displays are dependent on a prior search.

The more sophisticated, existing interactive search systems simplify the alteration of query statements by maintaining in storage various stages of earlier query formulations, thereby enabling the user to perform only selective alterations at each step. Appropriate tutorial features may also be used during the search process, including the display of multiple-choice response frames that give the user a choice of processing steps to perform next, as well as backtrack methods capable of returning to earlier displays. A typical interactive search process is illustrated in the chart of Fig. 6-9 [37].

*B Feedback Theory

The query reformulation methods mentioned earlier are designed to generate query statements capable of retrieving the relevant items and of rejecting the nonrelevant ones. When the query modification is carried out manually, the process is difficult to control, first because the characteristics of the relevant and nonrelevant items are known imperfectly, and also because document characteristics are not easily transformed into correct query formulations.

The *relevance feedback* process described in Chapter 4 relieves the user, or search intermediary, by automatically generating new query formulations based on relevance assessments obtained from the users during earlier search operations. More specifically, a newly constructed query will exhibit a high similarity with the document set previously identified as relevant and a low similarity with the nonrelevant document set. Assuming that the set of relevant documents D_R with respect to a query is known, and hence also the set of nonrelevant documents D_{N-R}, the best query Q is one that maximizes a function F defined as the difference between the average query-document similarity for all relevant items and the average query-document similarity for the nonrelevant ones. That is, a query vector Q is wanted which maximizes F, where

$$F = \overline{\underset{D_i \epsilon D_R}{SIM}}\,(Q,D_i) - \overline{\underset{D_i \epsilon D_{N-R}}{SIM}}\,(Q,D_i) \tag{14}$$

$\overline{SIM}$ represents the average similarity coefficient between the query and the set of all documents included in the relevant and nonrelevant document subsets, respectively. When the similarity between vectors is measured by the cosine coefficient, the optimal query has term weights proportional to the difference between the average weights of the terms in the relevant and the average

weights of terms in the nonrelevant items [expression (5) in Chapter 4] [38,39, 40].

In practice the relevant and nonrelevant document sets D_R and D_{N-R} are of course not known in advance—if they were, no search would have to be conducted to identify them. Instead a query statement is used to retrieve a number of documents. The user then identifies some subset $D_{R'}$ of relevant items and some subset $D_{N'}$ of nonrelevant items. A new query can then be constructed proportional to the difference between the average of the R′ relevant items previously identified and the N′ nonrelevant items [expression (6) of Chapter 4]. This query will not, however, perform well unless the subsets $D_{R'}$ and $D_{N'}$ are representative of the actual relevant and nonrelevant sets D_R and D_{N-R} for the given query.

Consider as an example the situation of Fig. 6-10 where the original query is assumed to have retrieved three relevant and four nonrelevant items (located inside the dashed line of Fig. 6-10). The feedback query actually constructed by the normal relevance feedback system (open triangle in Fig. 6-10) will lie in the center of the set of relevant items where the average distance to the relevant is smallest. When that query is actually used for retrieval, a great many nonrelevant items located outside the original retrieval perimeter will then be retrieved. The ideal feedback query (filled triangle in Fig. 6-10) would be situated much closer to a set of relevant items not previously retrieved and farther away from the relevant items retrieved earlier. The retrieval failure illustrated in Fig. 6-10 can be avoided in part by deemphasizing the role of the previously retrieved items in constructing a feedback query and assigning greater importance to the original query statement. In other words, the feedback query is expected to be reasonably close to the original query statement, and in addition, it should exhibit a greater resemblance to the items previously identified as relevant than to the nonrelevant ones. A typical formula used to construct an

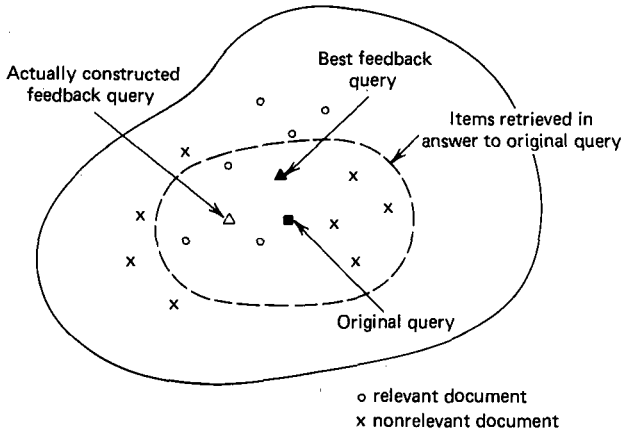

Figure 6-10 Unfavorable relevance feedback situation.

improved query statement Q′ from an original query Q, a set of R′ documents previously identified as relevant, and a set of N′ documents identified as nonrelevant is given by

$$Q' = \alpha(Q) + \beta\left(\frac{1}{R'}\sum_{D_i \epsilon D_{R'}} D_i\right) - \gamma\left(\frac{1}{N'}\sum_{D_i \epsilon D_{N'}} D_i\right) \tag{15}$$

where α, β, and γ are constants, and the terms in parentheses represent the earlier query statement, and the average relevant and average nonrelevant previously identified documents.

The relevance feedback operation is most effective when the relevant documents are tightly clustered and hence exhibit large pairwise similarities, and the nonrelevant documents are also tightly clustered. At the same time the distance between the relevant and nonrelevant items should be large; that is, the similarities between them should be small (see Fig. 4-14a for a schema representing such an ideal document space).

*C Feedback Variations

The relevance feedback process can be implemented in a variety of different processing modes. The formal feedback equation [expression (15)] makes provisions for the incorporation of both relevant and nonrelevant documents in the feedback process. However, the role of the two document types is basically unequal: when the relevant items are used to update the query, certain terms are added to the query and the weight of other terms is increased; the query is then forced to move in the direction where other relevant items may be expected to be located. When nonrelevant items are used for feedback purposes, certain terms are deleted from the query, and other term weights are decreased. The negative feedback operation diminishes the affinity of the query with some items but does not provide alternative directions to control the query movement. Thus a *positive* feedback strategy, where relevant documents are used to update the query serves to construct new queries that will exhibit greater resemblance with the items originally defined as relevant. A *negative* strategy using the nonrelevant items for query updating decreases the query similarity with these nonrelevant items but may not provide a positive direction toward which to move.

The relevance feedback process can then be executed in the following modes [41,42]:

1 A positive mode where relevant documents are available for feedback purposes

2 A negative mode where nonrelevant documents are used to update the query

3 A mixed mode incorporating both positive and negative modes

In each case, the number of documents used to update the query can be varied.

Among the strategies that have been formally evaluated are the standard mixed strategy [equation (15)], a strategy using *all* relevant items that may be available but only *one* nonrelevant document (normally the one retrieved earliest in a search providing ranked output), and a system which emphasizes the original query statement and deemphasizes the role of the feedback documents [the α parameter in expression (15) receives a large weight].

A positive relevance feedback operation is illustrated in the example of Table 6-7. The initial query vector is shown in Table 6-7a. The term weights are listed under the corresponding terms in each case. The updated query (Table 6-7c) contains terms from document 102 originally retrieved and identified as relevant to the query. Two document vectors (numbers 80 and 81) are shown at the bottom of Table 6-7 whose retrieval rank in order of query-document similarity improves from 14 to 7 and from 137 to 6, respectively, as a result of the feedback operation. A comparison of the vectors in Table 6-7b, d, and e reveals that the feedback operation for document 102 is directly responsible for the retrieval improvement of documents 80 and 81.

The positive feedback strategy is obviously not applicable when no relevant documents are retrieved by the original query vector. Furthermore the

Table 6-7 Positive Relevance Feedback Illustration

	Vector type	Illustration
a	Initial query vector	airplane available blast dynamic 12 12 12 12 gust information regime response 12 12 12 12 subsonic 12
b	Relevant document 102 retrieved with rank 2 (partial vector)	gust lift oscillating penetration 48 48 12 12 response subsonic sudden 24 12 12
c	Query modified by document 102	airplane available blast dynamic 12 12 12 12 gust information lift oscillating 60 12 48 12 penetration regime response subsonic 12 12 36 24 sudden 12
d	Relevant document 80 (improves from rank 14 to rank 7; partial vector)	gust lift penetration sudden 24 72 12 12
e	Relevant document 81 (improves from rank 137 to rank 6; partial vector)	lift oscillating sudden 84 12 12

Adapted from reference 13.

Query Q146: What information is available for dynamic response of airplanes to gusts or blasts in subsonic regime?

strategy must fail when dissimilarities are evident in the set of retrieved documents. In that case, the new query is in principle required to approach several different directions at once, and the effect of the feedback operation becomes unpredictable. A negative feedback operation then becomes mandatory.

A comparison of the positive and mixed retrieval strategies indicates that, on the average, the performance of queries that do retrieve some relevant documents in the initial search is not hindered by incorporating some nonrelevant documents in the feedback operation. However, the amount of perturbation in the query vectors is normally larger for the mixed feedback system than for the purely positive one. The mixed strategy must therefore be used with care because of the greater potential for improper query disturbance.

The purely negative feedback strategy is even more difficult to control. Consider the case of Table 6-8, where the most important query term ("data set") is subtracted out because that term happens to have been present in some of the nonrelevant items previously retrieved. Various solutions can be devised which will avoid the predicament illustrated in Table 6-8. One possibility consists in leaving original query terms intact in a negative feedback situation; the updated query will then consist of the original query terms plus additional negatively weighted terms taken from the nonrelevant documents previously retrieved. Alternatively, a thesaurus could be used to add related terms to the original query vector prior to the negative feedback operation. An illustration of such a modified negative feedback system is shown in Table 6-9 for the query previously used as an example in Table 6-8. A comparison of the final query statements shows that the problem exhibited in Table 6-8 is avoided by the process of Table 6-9 [13,42,43].

Several variants of the basic relevance feedback system have been described in the literature. These include a *query splitting* system designed to generate several subqueries from a given original query whenever the set of relevant items retrieved by a given query is not sufficiently homogeneous. Each of the subqueries is then expected to retrieve a different set of relevant items [44]. Another feedback variant, known as "cluster feedback," constructs separate feedback queries from the relevant and/or nonrelevant documents located in different document clusters in a clustered collection [13]. This last process thus performs the feedback operations separately in each document cluster. The feedback refinements remain to be thoroughly evaluated.

Table 6-8 Example of Inadequate Negative Feedback

	Type of vector	Illustration				
a	Initial query vector	available 12	current 12	data set 12		specification 12
b	Sum of retrieved nonrelevant documents	access 48	data set 60	file 24	list 24	structure 84
c	Standard negative feedback result	available 12	current 12	specification 12		

Adapted from reference 13.
Query: Please give specification for all currently available data sets.

Table 6-9 Negative Feedback with Related Concepts for Query of Table 6-8

	Type of vector	Illustration
a	Initial query vector	available current data set specification 12 12 12 12
b	Concepts related to "data set" with correlation strength	access bandwidth file interface line list 77 28 50 58 52 47 retrieval sort structure transmission 49 40 49 30
c	Related concept vector (top 5 concepts with weight of 24)	access file interface line structure 24 24 24 24 24
d	Query vector with related concepts	access available current data set file 24 12 12 12 24 interface line specification structure 24 24 12 24
e	Sum of retrieved nonrelevant documents	access data set file list structure 48 60 24 24 84
f	Updated query vector (computed by d − e with initial query terms intact)	access available current data set interface −24 12 12 12 24 line list specification structure 24 −24 12 −60

The basic relevance feedback process may be expected to produce between 10 and 20 percent improvement in precision for fixed recall levels. When a second feedback iteration is added, that is, when a modified query is itself modified a second time by a feedback process, additional improvements of a few percentage points may be gained.

An evaluation of the relevance feedback process raises complex problems that are not normally met in the evaluation of single search systems. In particular, it is unclear how documents should be treated that were originally retrieved by a given query and are retrieved once more by the modified feedback query. Some of these documents may well be relevant to the queries. However, a user may not be interested in seeing for a second time items that were already retrieved in an earlier search iteration, and the repeated retrieval of such relevant items should not be included in the recall-precision computations.

Several feedback evaluation systems have been proposed in the literature. The most effective of these known as the "test and control" evaluation, is represented in simplified form in Fig. 6-11 [45]. The idea is to break up a collection into two parts with roughly equal relevance properties. The first subcollection, known as the "test collection," is used in a relevance feedback situation to generate a set of modified feedback queries from the originally available queries. The second, so-called control collection, is then processed in two different searches first against the original queries and later against the modified queries. Recall-precision measurements are used in each case to compare the respective performance. Under experimental conditions, improvements of up to 10 percent in precision for fixed recall levels have been noted for the modified feedback queries compared with the original ones.

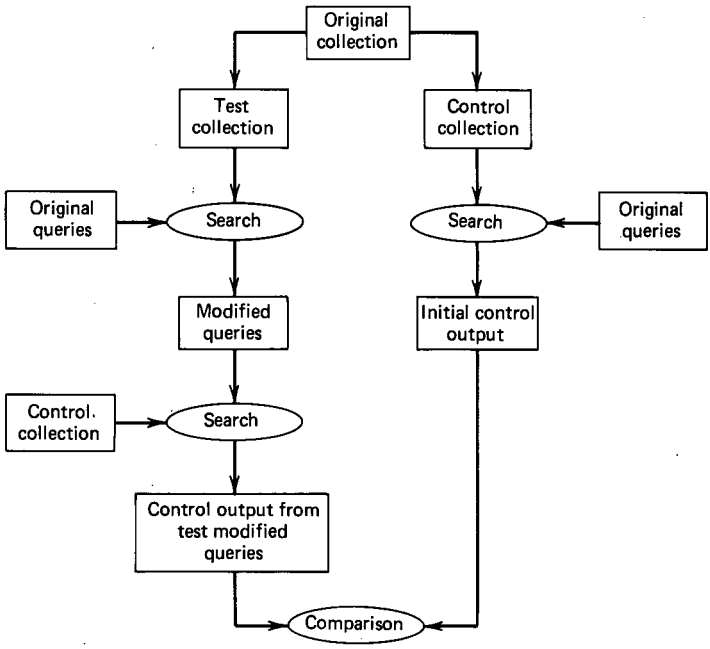

Figure 6-11 Test and control feedback evaluation.

D Dynamic Document Space

It was noted earlier that in some circumstances it becomes useful to modify not only the query statements during the course of the retrieval process, but also the document vectors themselves. For example, when the user population is fairly homogeneous and many queries of the same type are expected to be submitted to the retrieval system, a dynamic document modification process may lead to easier retrieval of relevant documents and a more useful document space for search purposes.

Consider, for example, the document modification system represented in simplified form in Fig. 6-12a. The process consists in modifying the documents retrieved with respect to a given query and identified as relevant by the user so as to render them somewhat more similar to the query that was used to retrieve them. This can be done by adding certain query terms to the various document vectors, or by increasing the weight of query terms that already occur in the document vectors. By making the document vectors more similar to a common query, they are also made more similar to each other. Hence when a new query is submitted that resembles the original one, the modified documents become more easily retrievable.

The document space modification process of Fig. 6-12b is an extension of the earlier one, because documents identified as nonrelevant by the user population are also altered in Fig. 6-12b so as to make these items less similar to the original query, while at the same time, the relevant documents are moved closer to the query as before. In each case, the idea is to perform only minor

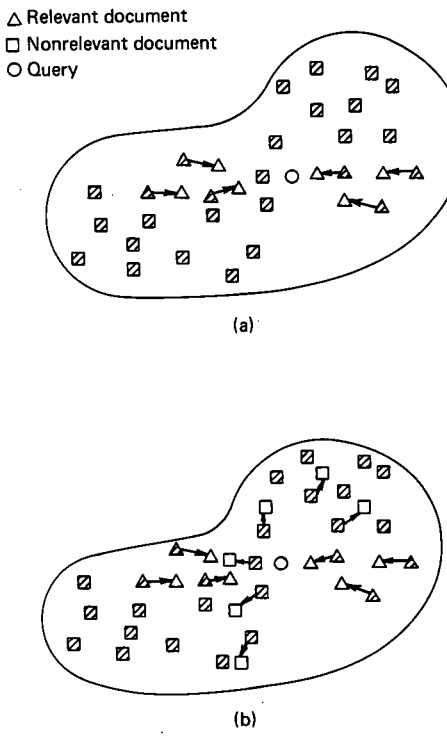

Figure 6-12 Document space modification. (a) Relevant document modification. (b) Relevant and nonrelevant document modification.

adjustments in the document vectors in response to any particular user-system interaction. Significant document movements will then occur only after a consensus has been reached among several different users concerning the usefulness of particular items in response to a number of different queries. Following many small individual moves of the document vectors, an equilibrium space may eventually be produced where no major additional document alterations occur. Such a conjecture remains to be proved [46].

When the user population of the retrieval system is not sufficiently homogeneous, few queries may be submitted that are similar to each other. In that case, it may not be possible to construct an improved document space. However, a document space modification method may nevertheless be useful in order to handle document *growth and retirement*. In particular, when a collection grows indefinitely through the addition of new items, limitations in the available storage space may make it necessary to retire from the active storage area those items that are least likely to be useful for retrieval purposes. The problem then consists in identifying items as candidates for possible retirement. A number of strategies suggest themselves for this purpose: items may be retired if they have never been retrieved in response to earlier queries or, having been retrieved, if they are always identified as nonrelevant to the respective user queries or if they are always retrieved late in a search.

Some of the foregoing retirement strategies involve a document space

modification step, followed by the retirement of items that will have moved to the periphery of the document space. Preliminary evaluations indicate that automatic document retirement methods can usefully be incorporated into a document retrieval system [47].

The concept of document collection modification remains to be tried out in real user environments. In principle, it appears natural that the relationship between items should be altered as science advances, new applications emerge for old ideas, and the use of the collection changes. On the other hand, when changes are introduced into the stored data, searches that are repeated periodically will not necessarily retrieve the same information as before. This loss of stability may be disturbing to some users. Furthermore, the effort to alter the document representations of many items of a transitory nature may hardly be worth the effort. Because of the connection between collection modification on the one hand and collection management (growth and retirement) on the other, the document space modification process may be expected to attract increased attention in the future.

6 CITATION PROCESSING

A Basic Citation Properties

The use of bibliographic references and citations in document processing has not yet been considered. Citations and references between documents are usable for many types of literature studies. References and citations attached to individual literature items may provide alternative expressions of document content. Citation and reference counts can also be used to assess the importance of individual documents or of complete document collections, by assuming that citation frequencies reflect the influence of bibliographic items in a field of study. By extension this argument may also be applied to authors of documents or to selected groups of authors; in particular, author influence might be measured by using the citation frequencies of documents written by a given author. Finally, citations and references can be followed from item to item, and the resulting "network" of papers can serve as a basis for a variety of historic studies and for an examination of the development of individual disciplines [48–53].

A distinction must be made between bibliographic references and citations. A *reference* for a given document is a bibliographic item mentioned in the bibliography of that document. Thus the phrase "A refers to B" implies that document B is included as an entry in the bibliography of document A. *Citations* between documents specify additional relationships between bibliographic items. A citation is an inverse reference: when a document A refers to document B, the latter is cited by document A. For a particular document, it is then possible to identify a set of references obtainable by consulting the bibliography of that document, as well as a set of citations made by other documents to the document in question. To identify the citations it is necessary to use a

Figure 6-13 Citation terminology: Stiles *refers to* Luhn, or Luhn is *cited by* Stiles. (Luhn is contained in Stiles's reference list.)

citation index or, alternatively, to process the reference lists using appropriate sort and merge operations. The dual relationship between references and citations is represented graphically in Fig. 6-13—a reference by an outgoing arrow and a citation by an incoming arrow. When document B refers to document A, B may be expected to postdate A; hence vertical displacement from top to bottom in a citation graph represents increasing time.

The occurrence of bibliographic citations is a comparatively rare phenomenon. For example, about 25 percent of all published papers are never cited at all. Of the archive of citable papers in a particular field of endeavor, about one half are not cited in any given year. Of those that are cited in a particular year, 72 percent are cited once in that year, and 18 percent are cited twice. Thus only about 5 percent of the archive of citable papers is cited at least three times in a given year [54]. From these statistics, it is not difficult to conclude that when a given document attracts many dozens of citations—and, of course, such papers are not hard to find—the very rarity of the phenomenon carries significance. A long list of studies have also shown that citation counts for papers, journals, authors, etc., correlate highly with peer judgments concerning the importance and influence of documents and document authors.

*B Main Citation Usage

In order to make use of bibliographic reference and citation data, it is first necessary to collect citations for a given corpus of documents. This can be done by taking a number of source documents and following either references or citations to new documents. These new documents can then be used as sources for new searches that discover still other documents. Two possible search strategies are represented graphically in Fig. 6-14.

The search of Fig. 6-14a is a pure citation search which uses as a source a well-known (historical) document in a given area (Luhn). A citation index is then used to identify a number of more recent documents all of which cite Luhn. One of these citing items (Cleverdon) is then used to produce still newer documents which cite Cleverdon (documents Lancaster, Keen, Salton of Fig. 6-14a). The search could continue to cover many generations of documents, thereby using connections between documents over long periods of time.

The search of Fig. 6-14b, on the other hand, covers documents that remain more or less stationary in time. The source document (Luhn) is used as before to uncover the citing items Stiles, Cleverdon, and Sparck Jones. One of these (Sparck Jones) is then used in a backward search by taking the references for Sparck Jones (documents Luhn, van Rijsbergen, Harter) and obtaining further

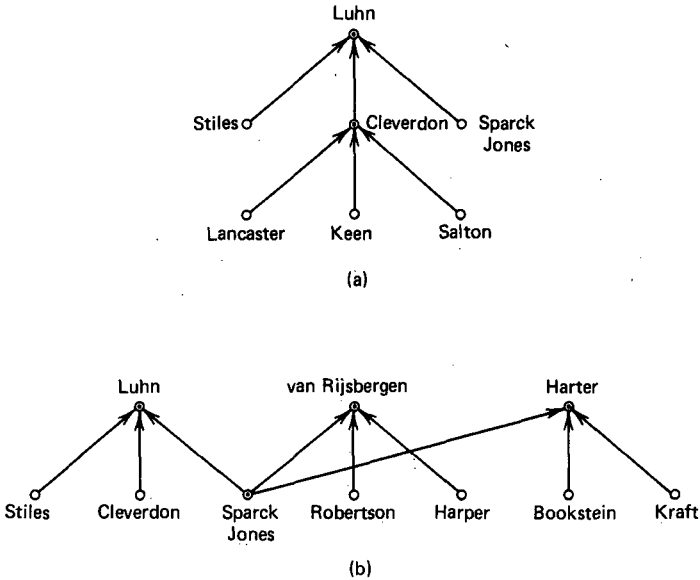

Figure 6-14 (a) Citation search forward in time. (b) Citation search forward and backward.

citations for some of these referenced items. Such a lateral search stays within a relatively narrow time frame, as opposed to the vertical search of Fig. 6-14a.

Citation and reference tracing methods of the kind illustrated in Fig. 6-14 may be used to generate complete citation networks for individual subject areas. Such citation networks are then used as a basis for historical researches of various kinds. A typical citation network is illustrated in Fig. 6-15, where once again downward displacement of the page implies increasing time.

Each reference or citation between two items represents a relationship indicator for the items. However, a direct reference (item A refers to item B) does not imply identity in the subject areas covered by the items. A stronger indication of congruence in subject matter is furnished by the so-called bibliographic coupling and cocitation counts. Two items are coupled bibliographically when they share certain references, that is, when their bibliographies contain various items in common. The coupling strength between two items may in fact be defined as the number of references in common for both items [55,56]. One may expect that when the coupling strength between documents is sufficiently large, the subject matter of the items exhibits strong similarities.

Two documents may also be related by common citation patterns from other documents. In particular, a *cocitation link* is said to relate two items when these items are jointly cited by a third document. The number of cocitations for two documents, that is, the number of documents that jointly cite the two items, is believed to be more significant than the bibliographic coupling strength in pointing to subject matter similarities. Cocitations have thus been widely used to study the development of individual fields of science as well as

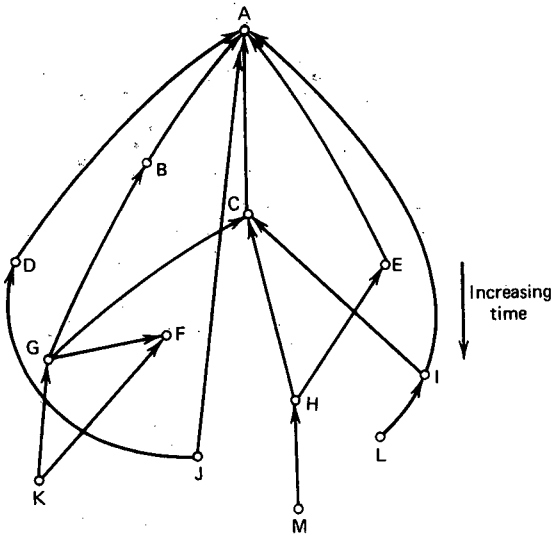

Figure 6-15 Typical citation network.

the structure of science as a whole [57–59]. Figure 6-16 presents a sample contour map in which individual documents are grouped, based in part on the strength of their cocitation links. In particular, the diameter of the circles around each document is related to the total number of citations attracted by the documents and hence to the importance of the documents. The distance between two documents, on the other hand, is inversely related to the cocitation strength between them, that is, the closer the items in the map, the higher

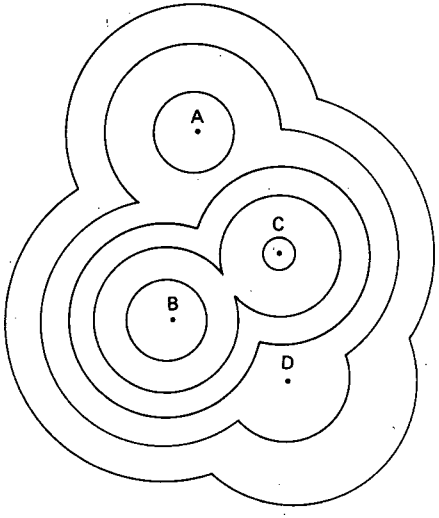

Figure 6-16 Typical contour map based on cocitations.

the number of cocitations between them. Maps such as that of Fig. 6-16 have been constructed at different periods in time for certain subject areas thereby describing the development of a given field of study [50,59].

If citations and references attached to documents carry subject significance, it appears reasonable to use them for content representation in addition to, or instead of, the normal index terms. In particular, if the citation and reference patterns are known for a collection of documents, a particular document could be represented by a bibliographic attribute vector of the form

$$DOC_i = (r_{i1}, r_{i2}, \ldots, r_{in}, c_{i1}, c_{i2}, \ldots, c_{im}) \tag{16}$$

where r_{ij} is an identifier representing the jth reference attached to DOC_i and c_{ik} the kth citation to the document. The normal vector similarity computations between pairs of documents or between queries and documents, can then be performed using the standard term vectors or alternatively, the bibliographic attribute vectors of expression (16). The two vector forms can also be combined by adding the bibliographic information to the normal index terms.

Studies in which bibliographic indicators have replaced the standard terms and keywords for content identification have shown that bibliographic indicators represent more specific content identifications than normal keywords [55, 60,61]. Furthermore, precision improvements have been noted in searches carried out with bibliographic vectors replacing the normal term vectors [62]. In some retrieval environments the bibliographic identifiers may be immediately available as a by-product of the computer composition of the documents, whereas the terms and keywords might have to be generated by a more or less complicated indexing process. One may expect an increased utilization of bibliographic data for retrieval purposes in the immediate future [63].

7 SUMMARY

The standard information retrieval operations may be refined by using sophisticated automatic indexing methods that are capable of assigning weighted, instead of binary, terms to the documents of a collection. The term weights may be based on the term occurrence characteristics in the documents of a collection—for example, terms occurring mostly in documents identified as relevant to the user queries may receive higher weights than terms occurring in the nonrelevant documents.

Improved search efficiency may be obtained by clustering the document collection and placing into common classes all documents that appear to be sufficiently similar to each other. Various automatic classification methods are available for this purpose, and efficient top-down or bottom-up cluster search methods have been devised.

The manipulation of weighted query and document terms also leads to methods for generating improved query statements based on information obtained from the user population in the course of the retrieval operations. The

relevance feedback operations have been shown experimentally to lead to substantial improvements in retrieval effectiveness. Relevance information obtained from the users about the importance of certain documents in the collection can also lead to improvements in the document descriptions themselves, and to modern collection management strategies capable of accommodating growing or shrinking document collections. Furthermore, relevance data are needed for the construction of improved term weighting measures: indeed the relevance feedback process represents an obvious possibility for the estimation of the term relevance factor. Finally, relevance information may be used to generate trial or starting clusters in various heuristic clustering systems.

Improvements in search and retrieval performance may be obtainable by incorporating bibliographic citations instead of only content terms in the document descriptions. There are indications that at a time when document texts are increasingly available in machine-readable form, the latter possibility may be especially attractive.

REFERENCES

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

[2] C.J. van Rijsbergen, Information Retrieval, 2nd Edition, Butterworths, London, 1979.

[3] H.E. Stiles, The Association Factor in Information Retrieval, Journal of the ACM, Vol. 8, No. 2, April 1961, pp. 271–279.

[4] M.E. Maron and J.L. Kuhns, On Relevance, Probabilistic Indexing and Information Retrieval, Journal of the ACM, Vol. 7, No. 3, July 1960, pp. 216–244.

[5] W.B. Croft, Organizing and Searching Large Files of Document Descriptions, Doctoral Thesis, University of Cambridge (England), 1978.

[6] H. Wu and G. Salton, A Term Weighting Model Based on Utility Theory, in Information Retrieval Research, R.N. Oddy, S.E. Robertson, C.J. van Rijsbergen and P.W. Williams, editors, Butterworths, London, 1981, pp. 9–22.

[7] C.T. Yu, K. Lam, and G. Salton, Term Weighting in Information Retrieval Using the Term Precision Model, Journal of the ACM, Vol. 29, No. 1, January 1982, pp. 152–170.

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

[9] A. Bookstein, On the Perils of Merging Boolean and Weighted Retrieval Systems, Journal of the ASIS, Vol. 29, No. 3, May 1978, pp. 156–158.
pp. 156–158.

[10] W. Goffman, An Indirect Method of Information Retrieval, Information Storage and Retrieval, Vol. 4, No. 4, December 1968, pp. 361–373.

[11] N. Jardine and R. Sibson, A Model for Taxonomy, Mathematical Biosciences, Vol. 2, No. 2–3, May 1968, pp. 465–482.

[12] K. Sparck Jones, Some Thoughts on Classification for Retrieval, Journal of Documentation, Vol. 26, No. 2, June 1970, pp. 89–101.

[13] G. Salton, Dynamic Information and Library Processing, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[14] W.B. Croft, A Model of Cluster Searching Based on Classification, Information Systems, Vol. 5, No. 3, 1980, pp. 189–195.
[15] G. Salton, Cluster Search Strategies and the Optimization of Retrieval Effectiveness, in The Smart Retrieval System, G. Salton, editor, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971, Chapter 10.
[16] R.T. Grauer and M. Messier, An Evaluation of Rocchio's Clustering Algorithm, in The Smart Retrieval System, G. Salton, editor, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971, Chapter 11.
[17] R.T. Dattola, Experiments with a Fast Algorithm for Automatic Classification, in The Smart Retrieval System, G. Salton, editor, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971, Chapter 12.
[18] G. Salton and H. Wu, A Comparison of Inverted File Searching with Cluster Tree Search Operations, Technical Report, Computer Science Department, Cornell University, Ithaca, New York, 1981.
[19] C.W. Cleverdon and E.M. Keen, Factors Determining the Performance of Indexing Systems, Aslib Cranfield Research Project, Volumes 1 and 2, Cranfield (England), 1968.
[20] G. Salton and M.E. Lesk, Computer Evaluation of Indexing and Text Processing, Journal of the ACM, Vol. 15, No. 1, January 1968, pp. 8–36.
[21] V.E. Giuliano and P.E. Jones, Linear Associative Information Retrieval, in Vistas in Information Handling, P. Howerton, editor, Spartan Books, Rochelle Park, New Jersey, 1963.
[22] B. Litofsky, Utility of Automatic Classification Systems for Information Storage and Retrieval, Ph.D. dissertation, University of Pennsylvania, Philadelphia, Pennsylvania, 1969.
[23] A.R. Meetham, Graph Separability and Word Grouping, Proceedings of 1966 ACM National Conference, Association for Computing Machinery, New York, 1966, pp. 513–514.
[24] K. Sparck Jones, Automatic Keyword Classification for Information Retrieval, Butterworths, London, 1971.
[25] G. Salton, Experiments in Automatic Thesaurus Construction for Information Retrieval, Information Processing-71, North Holland Publishing Company, Amsterdam, 1972, pp. 115–123.
[26] P.K.T. Vaswani and J.B. Cameron, The NPL Experiments in Statistical Word Associations and Their Use in Document Indexing and Retrieval, Computer Science Report No. 42, National Physical Laboratory, Teddington, England, April 1970.
[27] R. Attar and A.S. Fraenkel, Local Feedback in Full-Text Retrieval Systems, Journal of the ACM, Vol. 24, No. 3, July 1977, pp. 397–417.
[28] D.M. Jackson, The Construction of Retrieval Environments and Pseudoclassification Based on External Relevance, Information Storage and Retrieval, Vol. 6, No. 2, June 1970, pp. 187–219.
[29] C.T. Yu, A Formal Construction of Term Classes, Journal of the ACM, Vol. 22, No. 1, January 1975, pp. 17–37.
[30] C.T. Yu, A Methodology for the Construction of Term Classes, Information Storage and Retrieval, Vol. 10, Nos. 7/8, July/August 1974, pp. 243–251.
[31] C.T. Yu and V.V. Raghavan, Single-Pass Method for Determining the Semantic Relationship between Terms, Journal of the ASIS, Vol. 28, No. 5, November 1977, pp. 345–354.
[32] V.V. Raghavan and C.T. Yu, Experiments on the Determination of the Relation-

ships between Terms, Proceedings of First Annual SIGIR Conference, Association for Computing Machinery, New York, May 1978, p. 150.

[33] M.E. Lesk and G. Salton, Interactive Search and Retrieval Methods Using Automatic Information Displays, AFIPS Conference Proceedings, Vol. 34, Montvale, New Jersey, 1969, pp. 435–446.

[34] R.N. Oddy, Information Retrieval through Man-Machine Dialogue, Journal of Documentation, Vol. 33, No. 1, March 1977, pp. 1–14.

[35] C. Vernimb, Automatic Query Adjustment in Document Retrieval, Information Processing and Management, Vol. 13, No. 6, 1977, pp. 339–353.

[36] J.H. Williams, Jr., Functions of a Man-Machine Interactive Information Retrieval System, Journal of the ASIS, Vol. 22, No. 5, September–October 1971, pp. 311–317.

[37] P.C. Mitchell, J.T. Rickman, and W.E. Walden, SOLAR—A Storage and On-Line Automatic Retrieval System, Journal of the ASIS, Vol. 25, No. 5, September–October 1973, pp. 347–358.

[38] J.J. Rocchio, Jr., Relevance Feedback in Information Retrieval, in The Smart Retrieval System—Experiments in Automatic Document Processing, G. Salton, editor, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971, Chapter 14.

[39] G. Salton, Relevance Feedback and the Optimization of Retrieval Effectiveness, in The Smart Retrieval System—Experiments in Automatic Document Processing, G. Salton, editor, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971, Chapter 15.

[40] J.J. Rocchio, Jr., Document Retrieval Systems—Optimization and Evaluation, Report ISR-10 to the National Science Foundation, Harvard Computation Laboratory, Cambridge, Massachusetts, March 1966.

[41] E. Ide, New Experiments in Relevance Feedback, in The Smart Retrieval System, G. Salton, editor, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971, Chapter 16.

[42] E. Ide and G. Salton, Interactive Search Strategies and Dynamic File Organization, in The Smart Retrieval System, G. Salton, editor, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1971, Chapter 18.

[43] J. Kelly, Negative Response Relevance Feedback, in The Smart Retrieval System, G. Salton, editor, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1971, Chapter 20.

[44] A. Borodin, L. Kerr, and F. Lewis, Query Splitting in Relevance Feedback Systems, in The Smart Retrieval System, G. Salton, editor, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1971, Chapter 19.

[45] Y.K. Chang, C. Cirillo, and J. Razon, Evaluation of Feedback Retrieval Using Modified Freezing, Residual Collections, and Test and Control Groups, in The Smart Retrieval System, G. Salton, editor, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971, Chapter 17.

[46] T.L. Brauen, Document Vector Modification, in The Smart Retrieval System—Experiments in Automatic Document Processing, G. Salton, editor, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971, Chapter 24.

[47] K. Sardana, Automatic Document Retirement Algorithms, Scientific Report No. 22, Section 12, Computer Science Department, Cornell University, Ithaca, New York, 1974.

[48] E. Garfield, Citation Indexing—Its Theory and Applications in Science, Technology and Humanities, John Wiley and Sons, New York, 1979.

[49] D.J. deSolla Price, Networks of Scientific Papers, Science, Vol. 149, No. 3683, 1965, pp. 510–515.
[50] E. Garfield, Citation Measures as an Objective Estimate of Creativity, Current Contents, No. 34, August 16, 1970, pp. 4–5.
[51] J.R. Cole and S. Cole, The Ortega Hypothesis, Science, Vol. 178, No. 4059, October 27, 1972, pp. 368–375.
[52] E. Garfield, Citation Analysis as a Tool in Journal Evaluation, Science, Vol. 178, No. 4060, November 3, 1972, pp. 471–479.
[53] J.H. Westbrook, Identifying Significant Research, Science, Vol. 132, No. 3435, October 18, 1960, pp. 1229–1234.
[54] D.J. deSolla Price, The Citation Cycle, in Key Papers in Information Science, B. C. Griffith, editor, Knowledge Industry Publications Inc., White Plains, New York, 1980, pp. 195–210.
[55] M.M. Kessler, Comparison of Results of Bibliographic Coupling and Analytic Subject Indexing, American Documentation, Vol. 16, No. 3, July 1965, pp. 223–233.
[56] M.M. Kessler, Bibliographic Coupling between Scientific Papers, American Documentation, Vol. 14, No. 1, January 1963, pp. 10–25.
[57] E. Garfield, ISI Is Studying the Structure of Science through Cocitation Analysis, Current Contents, No. 7, February 13, 1974, pp. 5–10.
[58] H. Small and B.C. Griffith, The Structure of Scientific Literature, I. Identifying and Graphing Specialties, Science Studies, Vol. 4, 1974, pp. 17–40.
[59] H. Small, Co-citation in the Scientific Literature: A New Measure of the Relationship between Two Documents, Journal of the ASIS, Vol. 24, No. 4, July–August 1973, pp. 265–269.
[60] E. Garfield, Science Citation Index—A New Dimension in Indexing, Science, Vol. 144, No. 3619, May 8, 1964, pp. 649–654.
[61] E. Garfield, The Citation Index as a Subject Index, Current Contents, No. 18, May 1, 1974, pp. 5–7.
[62] G. Salton, Automatic Indexing Using Bibliographic Citations, Journal of Documentation, Vol. 27, No. 2, June 1971, pp. 98–110.
[63] E. Garfield, ISI's On-Line System Makes Searching So Easy Even a Scientist Can Do It: Introducing METADEX Automatic Indexing and ISI/BIOMED Search, Current Contents, No. 4, January 26, 1981, pp. 5–8.

BIBLIOGRAPHIC REMARKS

Additional coverage of query formulation and query modification in information retrieval is provided by

H.S. Heaps, Information Retrieval—Computational and Theoretical Aspects, Academic Press, New York, 1978.
G. Salton, Dynamic Information and Library Processing, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

Clustering algorithms are examined in many texts. The second reference below (Hartigan) provides actual computer programs for several clustering methods and the third (Sparck Jones) specializes in term classification

P.H.A. Sneath and R.R. Sokal, Numerical Taxonomy: The Principles and Practice of Numerical Classification, W. H. Freeman, San Francisco, California, 1973.

J.A. Hartigan, Clustering Algorithms, John Wiley and Sons, New York, 1975.

K. Sparck Jones, Automatic Keyword Classification for Information Retrieval, Butterworths, London, 1971.

A thorough coverage of citation processing is provided by

E. Garfield, Citation Indexing—Its Theory and Applications in Science, Technology and Humanities, John Wiley and Sons, New York, 1979.

EXERCISES

6-1 Consider the following similarity matrix between objects

	1	2	3	4	5	6	7	8	9
1	0	0	1	0	1	1	1	0	0
2	0	0	0	0	0	0	0	0	1
3	1	0	0	0	1	0	1	0	0
4	0	0	0	0	0	1	0	1	0
5	1	0	1	0	0	0	0	0	0
6	1	0	0	1	0	0	0	1	0
7	1	0	1	0	0	0	0	0	0
8	0	0	0	1	0	1	0	0	0
9	0	1	0	0	0	0	0	0	0

a Find all the cliques.

b Find all the connected components.

c State the algorithm used to determine the clusters using a single-link method.

d Give a list of desirable clustering properties for a document collection leading to an effective and/or efficient retrieval system. Under what circumstances would it be preferable to use cliques rather than connected components to search a collection of objects, and vice versa? What are the tradeoffs between the two methods?

e Do the cliques listed under part a exhibit the desirable properties listed under part d? Same question for the connected components listed under part b. If not, how could the cliques or the connected components be modified to produce an improved classification?

6-2 What is the main idea behind the pseudoclassification process? What are the differences between a standard term classification and the notion of pseudoclassification? Would you expect the results of a pseudoclassification to remain unchanged when the user population of a retrieval system changes?

6-3 How can a citation index be used in retrieval? Suppose the cocitation strength were available for all document pairs in a collection of documents. How could that fact be utilized to design an effective retrieval service? Are there obvious similarities between the use of citations and the use of pseudoclassification in retrieval?

6-4 Consider the document output obtained for a given query by an initial search and a feedback search based on user relevance assessments for the top three retrieved items.

	Initial search			Feedback search	
	Rank	Document number	Relevant	Document number	Relevant
Initially retrieved	1	680		425	x
	2	425	x	430	
	3	430		129	x
	4	129	x	680	
	5	320		529	
	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.

a Compute the recall-precision pairs following the retrieval of each document for both searches.

b Are the differences in recall-precision measures reflective of the improvements actually obtained by the feedback process?

c Design a feedback evaluation system which emphasizes the retrieval of relevant items not previously seen by the user.

d Use the sample search to compare the new feedback evaluation derived under part c with the original evaluation under part a.

6-5 Consider the following document collection indexed by three terms A, B, and C:

	D_1	D_2	D_3	D_4	D_5
A	1	0	1	0	1
B	1	1	0	0	1
C	0	1	0	1	0

a Exhibit the documents retrieved by the following queries

(A OR B) AND C
(A AND B) OR C
A NOT B

b How many documents would you expect to retrieve using the following weighted Boolean queries for the sample collection?

$A_{0.33}$ OR $B_{0.66}$
$A_{0.33}$ AND $B_{0.66}$
$A_{0.33}$ NOT $B_{0.50}$

c Do any common documents appear in the responses to the three weighted Boolean queries of part b? Carefully explain your reasoning.