Chapter 4

The SMART and SIRE Experimental Retrieval Systems

0 PREVIEW

This chapter deals with analysis, file organization, search, and retrieval methodologies that may be used with the advanced information retrieval systems of the future. The SMART system is perhaps the best known of the experimental systems that are not based on a standard inverted file technology. The SMART system design is described in detail in this chapter, including the automatic indexing methods, the clustered file organization, which collects related records into common classes; and the interactive search process which is used to construct improved query formulations based on relevance information supplied by the users to the system.

Various other experimental retrieval systems, including SIRE, also utilize novel features that are not common in conventional retrieval. Some of these extensions are examined at the end of this chapter. The use of local clustering methods and the incorporation of weighted terms in Boolean retrieval are considered in this connection together with additional user feedback and query reformulation methods.

1 INTRODUCTION

Conventional retrieval systems are based for the most part on a common set of principles and methodologies. The documents are normally indexed manually

by subject experts or professional indexers using a prespecified, controlled vocabulary; alternatively, some systems use the words included in document texts or text excerpts as index terms. Users or search intermediaries formulate search statements using terms from the accepted vocabulary together with appropriate Boolean operators between terms. The main file search device is an auxiliary, so-called inverted directory which contains for each accepted content identifier and for some of the objective terms a list of the document references, or markers, to which that term has been assigned. In a free text search system, the inverted directory contains the text words from the documents and the references to all documents containing each given word. The documents to be retrieved in response to a given search request are then identified by obtaining from the inverted directory the document reference lists corresponding to each query term, and performing appropriate list comparison and merging operations in accordance with the logical search term associations contained in the query statements. An *exact match* retrieval strategy is used which consists of retrieving all items whose content description contains the term combination specified in the search requests. Furthermore, all retrieved items are considered by the system to be equally relevant to the user's needs, and normally no special methods are provided for ranking the output items in presumed order of goodness for the user.

The existing methodology has certain obvious advantages. In some cases, the indexers and search intermediaries may become expert in assigning useful content indicators to the stored documents and incoming user queries. The retrieval system will then exhibit a high level of effectiveness. The inverted file design also produces rapid response times, because the inverted file manipulations will identify all matching documents before the main document file is ever used. The document file must then be accessed only for those documents which will be shown to the user in response to the query.

The benefits of the conventional retrieval designs are available only under special circumstances, however. In particular, when the indexing and search operations are not carried out consistently by the several intermediaries, the effectiveness of the retrieval operations will suffer. If, for example, one indexer uses "search" when another would use "retrieval," some documents may not be retrieved when wanted. The use of controlled vocabulary lists can minimize this problem, but cannot eliminate it. In practice, the required degree of indexing consistency cannot be guaranteed when similar information items are treated by different indexers and searchers. In such circumstances, the various documents pertaining to a common subject area may be identified very differently, and many of these items may not be retrievable when wanted.

The inverted directory of index terms is most useful in situations where the search vocabulary remains static over long periods of time and is limited to a relatively small number of terms. In such cases, the directory size will remain manageable and relatively little updating may be needed. In practice, the indexing vocabulary is often large, consisting of several tens of thousands of terms for each subject area, and the vocabulary is not stable. The inverted directories

to be stored may then become very large, and these directories must be kept up to date when new documents are added to the file or new vocabulary terms are introduced.

The customary library file organization which places in related or adjacent file positions all items that exhibit similar subject content is not followed by the existing inverted file systems. Instead, the main document file is often kept in arbitrary order, and it becomes difficult, given the location of a particular useful document, to determine the location of other related documents. This complicates the file search process, which ideally consists in locating certain useful items, and then proceeding from there to identify additional items resembling the ones found earlier. For the same reason, the inverted file strategy is not well suited to the processing of approximate queries designed to approach a given subject area in small steps, and to the kind of browsing which proves so useful in conventional libraries. Instead, it becomes necessary to formulate complete and specific queries, in the hope of retrieving the entire set of relevant items from the start.

The standard inverted file technology is difficult to change because of the large investments already made in the existing commercial systems. Nevertheless, substantial improvements are possible in the operations of the conventional systems. Thus, in some partly experimental systems, term weighting facilities have been superimposed on the standard Boolean query formulations and inverted file search procedures, leading to the ranking of retrieved documents in accordance with the weights of the matching query and document terms. This makes it possible to present the retrieved documents to the user in decreasing order of the sum of certain term weights. Presumably the user will find it easier to prepare improved query formulations when the items judged by the system to be most relevant are retrieved ahead of other more marginal items.

Systems have also been developed that are based on totally new conceptions of the retrieval task. One of the best known of these, the SMART system, is described in the remainder of this chapter together with certain other nonstandard retrieval system designs.

2 THE SMART SYSTEM ENVIRONMENT

*A Vector Representation and Similarity Computation

The SMART system distinguishes itself from more conventional retrieval systems in the following important respects: (1) it uses fully automatic indexing methods to assign content identifiers to documents and search requests; (2) it collects related documents into common subject classes, making it possible to start with specific items in a particular subject area and to find related items in neighboring subject fields; (3) it identifies the documents to be retrieved by performing similarity computations between stored items and incoming queries, and by ranking the retrieved items in decreasing order of their similarity with the query; and finally, (4) it includes automatic procedures for producing im-

	$TERM_1$	$TERM_2$	...	$TERM_t$
DOC_1	$TERM_{11}$	$TERM_{12}$	...	$TERM_{1t}$
DOC_2	$TERM_{21}$	$TERM_{22}$	...	$TERM_{2t}$
⋮	⋮			
DOC_n	$TERM_{n1}$	$TERM_{n2}$	...	$TERM_{nt}$

Figure 4-1 Term assignment array (n documents, t terms).

proved search statements based on information obtained as a result of earlier retrieval operations [1,2,3,4].

In the SMART system each record, or document, is represented by a *vector* of terms. That is, a particular document, DOC_i, is identified by a collection of terms $TERM_{i1}, TERM_{i2}, \ldots, TERM_{it}$, where $TERM_{ij}$ is assumed to represent the weight, or importance, of term j assigned to document i. By "term" is meant some form of content identifier, such as a word extracted from a document text, a word phrase, or an entry from a term thesaurus. A given document collection may then be represented as an array, or matrix, of terms where each row of the matrix represents a document and each column represents the assignment of a specific term to the documents of the collection. A sample term assignment array is shown in Fig. 4-1. In the SMART system, positive term weights are chosen for terms actually assigned to the documents (that is, $TERM_{ij}$ is a positive number when term j actually occurs in document i); and $TERM_{ij}$ is set equal to zero when term j is not present as an identifier of document i.

A particular query, say $QUERY_j$, can be similarly identified as a vector $QTERM_{j1}, QTERM_{j2}, \ldots, QTERM_{jt}$, where $QTERM_{jk}$ represents the weight, or importance, of term k assigned to query j. Instead of insisting on a complete match between all nonzero query and document terms before a document is retrieved by the system, the retrieval of a stored item can be made to depend on the magnitude of a similarity computation measuring the similarity between a particular document vector and a particular query vector as a function of the magnitudes of the matching terms in the respective vectors. A similarity measure often used with the SMART system is the cosine measure, defined as

$$\text{COSINE}(DOC_i, QUERY_j) = \frac{\sum_{k=1}^{t} (TERM_{ik} \cdot QTERM_{jk})}{\sqrt{\sum_{k=1}^{t} (TERM_{ik})^2 \cdot \sum_{k=1}^{t} (QTERM_{jk})^2}} \quad (1)$$

The cosine correlation measures the cosine of the angle between documents, or between queries and documents, when these are viewed as vectors in the multidimensional term space of dimension t. In three dimensions, when only three terms identify the documents, the situation may be represented by the configuration of Fig. 4-2. Each axis corresponds to a different term, and the position of each document vector in the space is determined by the magnitude

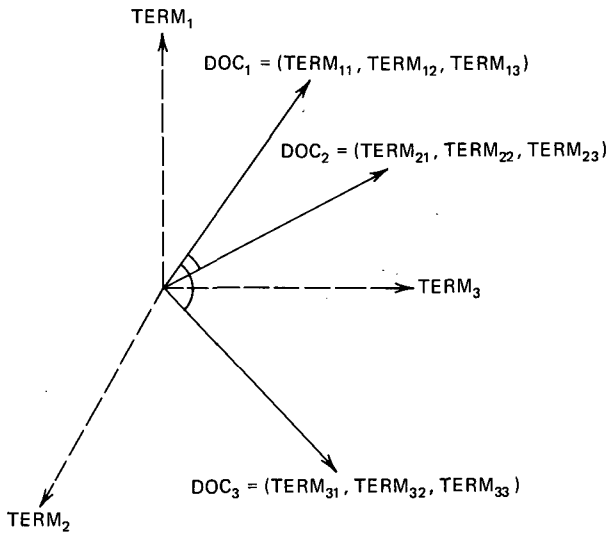


Figure 4-2 Vector representation of document space.

(weight) of the terms in that vector. The similarity between any two vectors is then represented as a function inversely related to the angle between them. That is, when two document vectors are exactly the same, the corresponding vectors are superimposed and the angle between them is zero.

The numerator of the cosine coefficient gives the sum of the matching terms between DOC_i and $QUERY_j$ when the indexing is binary, that is, when $TERM_{jk}$ is assumed equal to 1 whenever term k actually occurs in document i. When the indexing is not binary, the numerator represents the sum of the products of the term weights for the matching query and document terms. The denominator acts as a normalizing factor (by dividing the expression by the product of the lengths of the query and document vectors). This implies that each item is represented by a vector of equal length. If the angle between vectors is small and the normalized vectors are used, the cosine of the angle between the vectors may also be approximated by the distance between the tips of the corresponding vectors.

When a numeric measure of similarity is used for documents and queries, it is no longer necessary to retrieve all documents that exactly contain all the query terms. Instead the retrieval of a document can be made to depend on a particular threshold in the similarity measure or on a specific number of items to be retrieved. Assuming, for example, that a threshold of 0.50 is used, all items are retrieved for which the value of expression (1) is greater than or equal to 0.50; alternatively the top n items may be retrieved, where n is the number of items originally wanted. The retrieved documents may be conveniently presented to the user in decreasing order of their similarity values with the respective search requests. This is of special importance in an interactive retrieval situation where users are directly tied into the retrieval system, because new improved query formulations can then be constructed by utilizing information

extracted from previously retrieved documents. Since the first few documents retrieved in response to a search request are those exhibiting the greatest query-document similarity, they are also the ones most likely to be relevant to the users' information needs. Hence they may be most important for query reformulation purposes.

*B Vector Manipulation

The query reformulation process incorporated into the SMART retrieval system is known as "relevance feedback" because relevance assessments supplied by the users for previously retrieved documents are returned to the system and used to construct new query vectors. The reformulated queries can then be compared with the stored documents in a new search operation. The aim is to construct new queries exhibiting a greater degree of similarity with the documents previously identified as relevant by the user than the original queries; at the same time, the new queries are expected to be less similar to the documents identified as nonrelevant by the user than the originals. The assumption is that the reformulated queries will retrieve more items resembling the relevant ones previously retrieved, and fewer items resembling the nonrelevant ones.

The query reformulation process is then based on the following complementary operations:

1 Terms that occur in documents previously identified as relevant by the user population are added to the original query vectors, or alternatively the weight of such terms is increased by an appropriate factor in constructing the new query statements.

2 At the same time, terms occurring in documents previously identified as nonrelevant by the users are deleted from the original query statements, or the weight of such terms is appropriately reduced.

Obviously, the query reformulation process can be carried out automatically by the retrieval system, given only an indication of relevance or nonrelevance of certain previously retrieved items obtained from the user population.

The effect of the relevance feedback operation is represented in the illustration of Fig. 4-3. In Fig. 4-3 and the related diagrams that follow, each document is identified by a point representing the tip of the corresponding vector, and the distance between two points is assumed to be inversely related to the respective cosine similarities between the vectors. Thus when two points appear close together in the space of Fig. 4-3, a substantial similarity exists between the vectors; the reverse is true for two points that appear far apart in the space. In Fig. 4-3, an original query vector (represented by an open triangle) appears with three retrieved documents of which one was identified as nonrelevant and two as relevant. Following the previously mentioned changes in the original query, a new query (the closed triangle) is constructed which appears shifted in the space: the distance to the relevant items is now smaller than be-

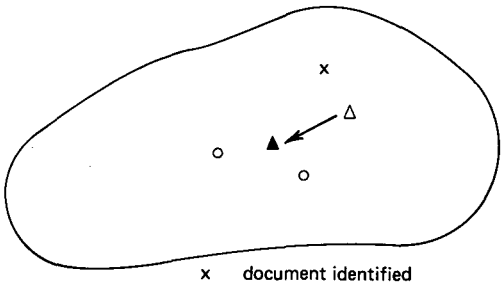


Figure 4-3 Relevance feedback illustration.

fore, and the distance to the nonrelevant one has increased. The relevance feedback process can be repeated several times in the hope of eventually obtaining adequate search output.

The cosine coefficient was introduced earlier to obtain a measure of similarity between a query vector and the various documents in a collection. The same measure also lends itself to the determination of similarities between pairs of documents. Thus, given the term vectors for two documents, DOC_i and DOC_j, the similarity between them may be defined as

$$\text{COSINE}(DOC_i, DOC_j) = \frac{\sum_{k=1}^{t} (TERM_{ik} \cdot TERM_{jk})}{\sqrt{\sum_{k=1}^{t} (TERM_{ik})^2 \cdot \sum_{k=1}^{t} (TERM_{jk})^2}} \tag{2}$$

By determining the similarity between various pairs of documents, it now becomes possible to construct a *clustered document file*, consisting of classes or clusters of documents such that the documents within a given class exhibit substantial similarities with each other. A clustered file resembles in concept the normal classified document arrangement used in conventional library situations, where items dealing with related subject areas are placed together in a common subject class. The clusters are, however, derived automatically and the construction method may be adapted to the particular collection environment under consideration. Thus, it is possible automatically to construct clustered files incorporating a large number of small clusters or a small number of large clusters; the classes may also overlap in the sense that certain documents may appear in more than one class. This last feature represents a substantial advantage over conventional library classification systems.

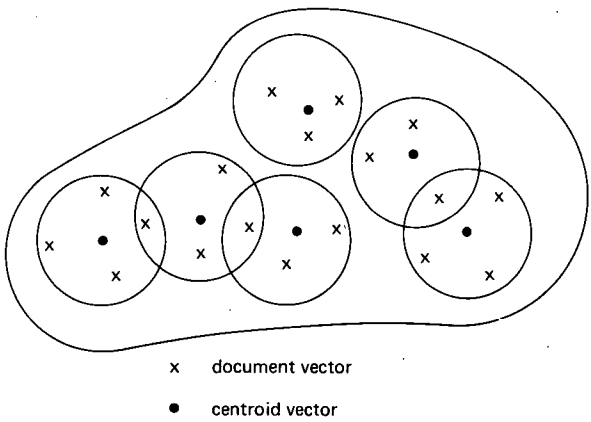


Figure 4-4 Sample clustered document collection.

A clustered document file is represented schematically in Fig. 4-4, where each x represents a document vector, and the distance between two particular x's is again assumed inversely proportional to their similarity. The large circular groupings in the figure identify the document clusters. It may be seen that some overlap exists between the clusters: certain documents located near the periphery of some classes actually appear in several classes. In order to render possible the manipulation of a clustered collection, it is convenient to use a special class vector, or *centroid*, to represent a given cluster. The centroids are similar to the centers of gravity of a set of points, and are represented by heavy dots in the illustration of Fig. 4-4. Given a set of m documents constituting a certain document class p, a given centroid vector $\text{CENTROID}_p = \text{CTERM}_{p1}, \text{CTERM}_{p2}, \ldots, \text{CTERM}_{pt}$ may be computed as the mathematical average of the document vectors included in the pth class. Thus, following the model used in the previous chapter for the construction of term classes, the weight of term k in the centroid for class p can be computed as the average of the weight of term k in all m document vectors incorporated into class p. That is,

$$\text{CTERM}_{pk} = \frac{1}{m} \sum_{i=1}^{m} \text{TERM}_{ik} \tag{3}$$

where the summation covers the m documents of class p.

Given a clustered collection of the type shown in Fig. 4-4, a document search is now carried out in two main steps. Each query is first compared with the various centroid vectors by computing the corresponding centroid-query similarity coefficients. For classes whose centroids exhibit a sufficiently high similarity with the query, the individual documents are next compared with the query, using the formula of expression (1), and documents showing sufficiently high similarities with the query are retrieved for the user's attention. Assuming that n documents exist in a collection which is divided into x clusters each con-

taining approximately n/x documents, the number of vector comparisons needed to compare a query with the best cluster is x + n/x (instead of the n comparisons needed in an unclustered file). The number of needed vector comparisons is minimized when the number of clusters x equals $\sqrt{n}$.

For large collections involving many heterogeneous documents, a great many clusters may need to be defined. In that case, the number of required query-centroid comparisons may become excessively large. The search efficiency may then be increased by taking the set of centroids and applying the clustering methodology previously used for the document vectors to compare pairs of centroid vectors. Centroids that are sufficiently similar are then grouped into superclasses identified by supercentroids, as shown in Fig. 4-5, where two superclasses are represented. The file search now requires three steps: first a comparison of the query with the supercentroid vectors; then for some supercentroids, a comparison with the individual centroids included in the corresponding superclusters; finally, for certain centroids, a comparison with the individual document vectors located in the respective document clusters.

The clustered file organization is adaptable to a growing collection environment, because new incoming documents can be treated just like incoming queries. The new items are compared with the existing supercentroids and centroids, and eventually they are included in those clusters for which the document-cluster similarity is sufficiently large. As in the standard inverted file organization, two distinct files are needed for the file search process:

1 The main document file, arranged in order by clusters such that all items included in a common cluster are retrievable in a single access to the main file

2 The auxiliary file of cluster centroids, and supercentroids arranged in a hierarchical tree format where each supercentroid contains pointers specifying the locations of the individual centroids for that superclass, and each centroid in turn points to the locations of the individual document vectors for that class

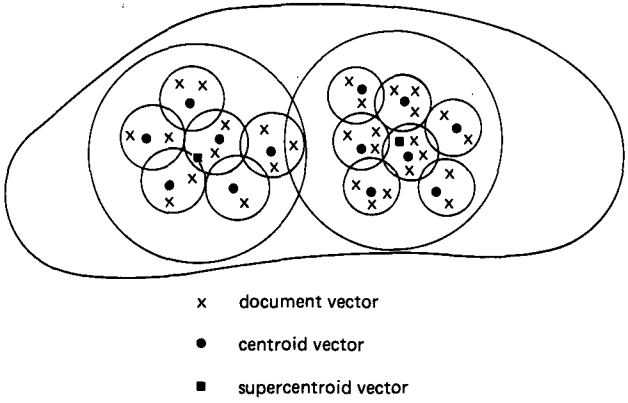


Figure 4-5 Introduction of superclusters.

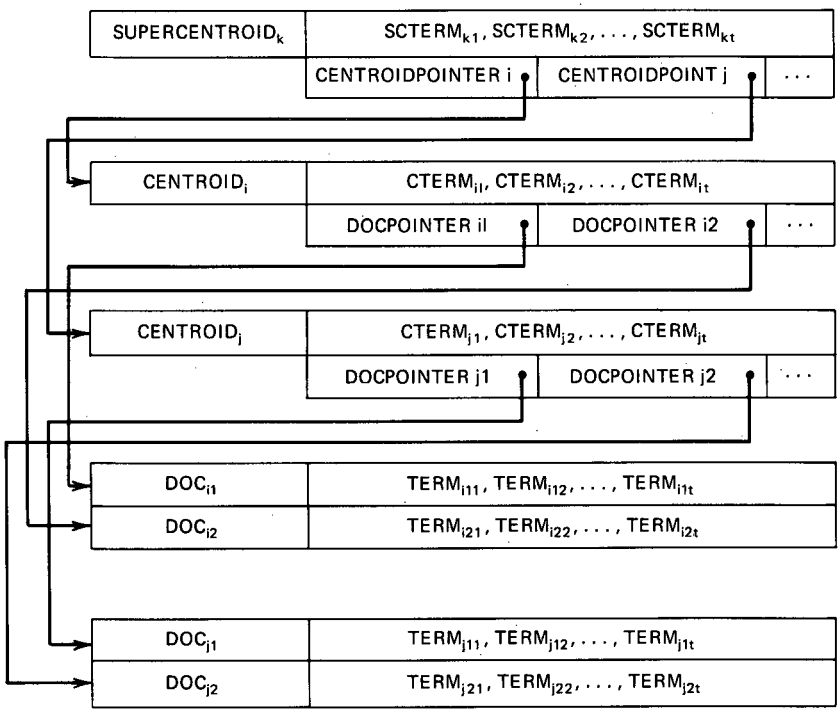
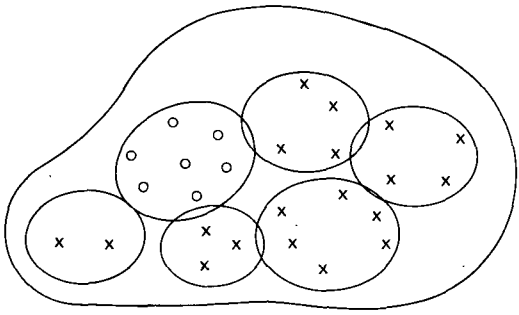


Figure 4-6 Hierarchical vector arrangement for clustered file.

The file arrangement is illustrated in simplified form in Fig. 4-6.

C Vector Generation

Consider now the document indexing process, that is, the method used to construct the document vectors. The basic function of an indexing system is the segregation of the subset of documents relevant to some query from the remainder of the collection. Preferably, all the relevant items might then occur in one or more document clusters, whereas the nonrelevant items would be placed in separate clusters. A situation of this type is shown in Fig. 4-7. Such an ideal document space might be constructed by assigning to the relevant document set the terms utilized by the user population to formulate the corresponding search requests. Unfortunately, it is difficult to know in advance what terms will be considered useful for query formulation purposes, and even in interactive systems where information can be generated about the usefulness of certain terms with respect to certain topic areas, it is still necessary to make the somewhat hazardous assumption that all users interested in specified subject areas would choose the same query terms to express their information needs. Similarly, one would have to assume that all such users would accept the same set of documents as relevant to their queries.



o relevant document with respect to a query

x nonrelevant document with respect to a query

Figure 4-7 Ideal document space.

In practice, it is not possible to find a single clustering which is ideal for all subject areas and all users, even given full relevance information in advance. Furthermore, new documents that are added to the collection must necessarily be processed without the use of document relevance information. The best policy may then lead to the use of index terms capable of distinguishing each particular document from the remainder of the collection. This can be achieved by using as a controlling criterion the document frequency of each term in the collection, that is, the number of documents to which a term is assigned. The term discrimination theory examined in the previous chapter indicates that the terms to be preferred are medium-frequency terms that are assigned to a few documents but not to the rest of the collection.

In the SMART system, the text of the document abstracts (or the text of the query statements obtained from the user population) is analyzed automatically. Terms whose document frequency is neither too large nor too small are incorporated directly into the document or query vectors for indexing purposes. Terms whose document frequency exceeds a given threshold are considered too broad and unspecific; they are rendered more specific by being combined with other terms into *term phrases* before assignment to the document and query vectors. On the other hand, terms with a very low document frequency that are assigned to one or two documents only are considered too specific; they can be broadened by grouping them into term classes of the kind found in a *thesaurus* of terms. The thesaurus class identifiers are then incorporated into the term and document vectors instead of the individual rare terms. These operations are described in more detail in the next section, where the SMART procedures are treated more thoroughly.

A simplified flowchart of the SMART processing chain is shown in Fig. 4-8. The processes represented on the left side of the chart consisting of the

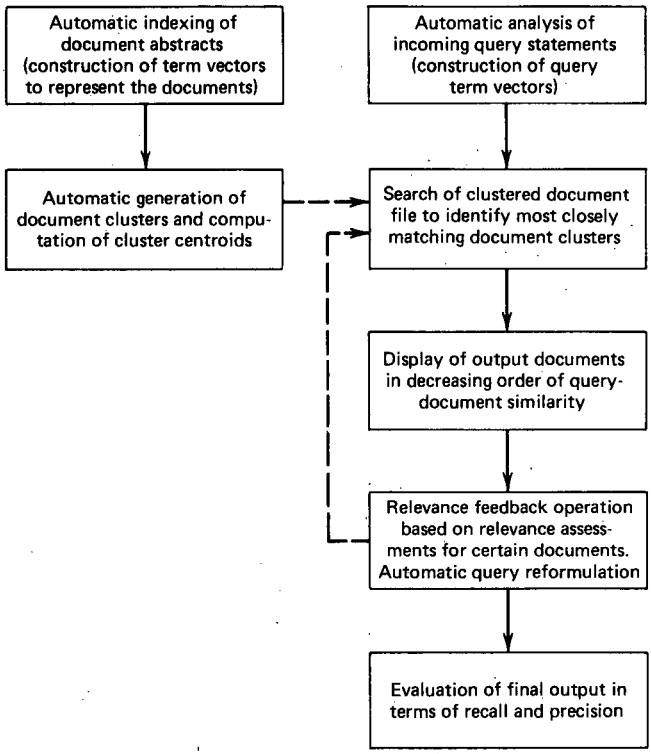


Figure 4-8 Simplified SMART system flowchart.

initial construction of term vectors for the documents and the generation of clustered files are carried out only once, or in any case at infrequent time intervals. The query-processing operations, on the other hand, including the generation of query vectors, the clustered file search, and the relevance feedback operations are performed for each query. Only the query-handling operations need to be done in real time while the user is present.

The SMART system is designed to make available a large variety of experimental methods, including many different automatic indexing procedures, query-document similarity functions, cluster file generation methods, and query-reformulation processes. The system can then serve as a test vehicle to validate the various experimental procedures. For this reason evaluation methods have been incorporated into the system which produce recall and precision information following the retrieval operations.

The following assumptions are implicit in the model used by the SMART environment for the representation of document and query operations:

1 Each document is represented by a particular vector, that is, a particular position in a t-dimensional vector space, where t is the number of permissible terms available for indexing purposes.

2 Each term included in a given document or query vector is assumed to be unrelated (orthogonal) to the other terms, and all terms are considered equally important (except for distinctions inherent in the assignment of weights to the individual terms).

The assumptions are in fact only first-order approximations to the true situation because documents exhibit not only subject content but also extent, or scope. A survey or tutorial text normally has greater scope than a research article. This suggests that the subject matter of a document be represented by a point in a certain location of the subject space as suggested by Figs. 4-3 to 4-5, and that an area of space surrounding the subject points be used to denote scope and extent. The document scope might be determined by using the frequencies with which individual documents are cited in the literature. Thus documents attracting many citations could be assumed to have wide scope.

The second area of simplification in the SMART model is the orthogonality assumption for the various index terms. In the SMART system, each term represents a separate coordinate in the vector space, and no term relationships are assumed to exist. In actual fact, words do not, however, occur independently of each other in the document texts, and neither do the index terms assigned to a collection of items. In recent years, a great deal of work has been devoted to the study of retrieval models which take into account certain term dependencies. These models tend to be complex and difficult to use. The available experimental evidence suggests that the greatest deviations from independence arise for the very rare terms that occur in a few documents only. The discrimination value theory indicates that these terms are not very important for retrieval purposes. For the vast majority of the medium- and high-frequency terms, the independence assumption is not in fact unreasonable.

Perhaps in a few years, some experimental retrieval systems will make provision for the representation of document scope and impact and for the use of term dependencies. At the present time efforts in this direction are in a very preliminary state.

3 SMART SYSTEM PROCEDURES

*A Automatic Indexing

When the SMART system was originally designed in the middle 1960s the experts generally felt that sophisticated language analysis procedures would be required in a machine indexing system to replace the intellect normally applied by the human indexer. Accordingly, the original SMART indexing system was based on the following language analysis tools:

1 Synonym dictionaries, or *thesauruses*, would be used to group the individual terms into classes of synonymous or related terms. When a thesaurus is available, each original term can be replaced by a complete class of related terms, thereby broadening the content description.

2 *Hierarchical term arrangements* could be constructed to relate the con-

tent terms in a given subject area. With such preconstructed term hierarchies, the standard content descriptions can be "expanded" by adding to a given content description hierarchically superior (more general) terms as well as hierarchically inferior (more specific) terms.

3 *Syntactic analysis* systems would serve for the specification of the syntactic roles of the terms, and for the formation of complex content descriptions consisting of term phrases and larger syntactic units. A syntactic analysis system can be used to supply specific content identifications, and it prevents confusion between compound terms such as "blind Venetian" and "Venetian blind."

4 *Semantic analysis* systems might supplement the syntactic units by using semantic roles attached to the entities making up a given content description. Semantic analysis systems utilize various kinds of knowledge extraneous to the documents, often specified by preconstructed "semantic graphs" and other related constructs.

The early test results obtained with the SMART system showed that some complicated linguistic methodologies that were believed essential to attain reasonable retrieval effectiveness were in fact not useful in raising performance [5,6]. In particular, the use of syntactic analysis procedures to construct syntactic content phrases and the utilization of concept hierarchies could not be proved effective under any circumstances. It may be that these failures were attributable not to the actual processes themselves but rather to the particular implementations used in the test situations. The fact is that linguistic theories were not sufficiently well understood 15 years ago to permit the construction of effective semantic maps and hierarchical term arrangements, or to design accurate and complete syntactic analysis systems. Some progress has been made in this area in the last few years. However, versatile linguistic methodologies that would be applicable to a wide variety of subject areas are still not easily incorporated into an unrestricted language processing environment at the present time. For this reason the standard SMART indexing procedures are based on simpler language process considerations that are by now well understood. These simple automatic methods can be shown to be superior to conventional indexing methodologies in laboratory test situations [6,7]:

1 The individual words that make up a document excerpt (abstract) or a query text are first recognized.

2 A stop list, comprising a few hundred high-frequency function words, such as "and," "of," "or," and "but," is used to eliminate such words from consideration in the subsequent processing.

3 The scope of the remaining word occurrences is broadened by reducing each word to word stem form; this can be done by using relatively simple suffix removal methods together with special rules to take care of exceptions, as previously explained [8,9].

4 Following suffix removal, multiple occurrences of a given word stem are combined into a single term for incorporation into the document or query vectors.

At this point, the standard query-document vector matching methods introduced in the previous section could be used to identify all documents whose word stem vectors are sufficiently similar to the query vectors. The SMART system retains a much larger number of content indicators than is customary in conventional systems, and the larger number and greater diversity of the terms compensate to some extent for the lack of precision in the term selection. Nevertheless, the foregoing process would provide inferior search results in many cases, because some word stems are obviously more important for the representation of document content than others. Two main operations are needed to transform the word stem vectors into useful term vectors: first a term weight can be assigned to each term reflecting the usefulness of the term in the collection environment under consideration; and second, terms whose usefulness is inadequate as reflected by a low term weight can be transformed into better terms [10–13].

As explained earlier, it is convenient to separate the term weighting task into two parts: first, one must take into account the term characteristics within a given document or document excerpt; second, it is important to consider also the function of the term in the remainder of the collection. The considerations detailed in Chapter 3 favor terms that exhibit high importance in the documents to which they are assigned, as measured, for example, by their occurrence frequencies in the individual documents. At the same time, the best terms must be able to distinguish the documents to which they are assigned from the remainder of the collection; hence their importance factor in the document collection as a whole ought to be low. The importance of a given term k in an individual document i is conveniently measured by the frequency of occurrence in the document $FREQ_{ik}$. The usefulness of the term in the collection as a whole may be reflected by the term discrimination value $DISCVALUE_k$, or alternatively by an inverse function of the document frequency $DOCFREQ_k$ (that is, the number of documents to which the term is assigned). Two possible term weighting functions reflecting the usefulness of term k in document i are

$$\text{WEIGHT}_{ik} = \frac{\text{FREQ}_{ik}}{\text{DOCFREQ}_k} \qquad (4)$$

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

Terms with a low weight according to expression (4) could in principle be deleted from the indexing vocabulary. In practice, the deletion of broad, high-frequency terms is likely to cause losses in recall, and the elimination of specific, low-frequency terms may impair the precision. It is then preferable to alter such terms completely. The most obvious methods available for this purpose consist in creating specific *term phrases* incorporating the high-frequency terms that are originally considered too broad, and forming *thesaurus classes* of the low-frequency terms that may be too specific to be used by themselves.

Consider first the phrase-formation process. Ideally, a phrase is a language construct with specific syntactic and semantic properties. Because a full syn-

tactic and/or semantic analysis of document and query texts is not currently possible for normal information retrieval purposes, a simple phrase-formation process must be used. The following phrase-formation criteria are of greatest importance:

1 The phrase components should occur in a common context within the document or query to which the phrase is assigned as a content identifier.
2 The phrase components should represent broad concepts, and their frequency of assignment to the documents of a collection should be sufficiently high.

In the SMART system a phrase is defined as a pair of two distinct word stems not contained on the stop list, such that the components occur in the same sentence within a document or query text, and at least one component has a document frequency in the collection exceeding a given threshold. A more stringent phrase-construction process is obtained by also taking into account the distance in the text between potential phrase components, that is, the number of intervening words between them. The following phrase-construction process can be used in practice:

1 Start with the query and document texts; use a stop list to eliminate common function words; generate word stems by using a suffix deletion process to reduce the original words.
2 Take pairs of the remaining word stems, and let each pair define a phrase provided that the distance in the text between components does not exceed n words (at most n − 1 intervening words), and that at least one of the components of each phrase is a high-frequency term; this frequency cutoff is manipulated to reduce the number of generated phrases to manageable size.
3 Phrases for which both components are identical are eliminated, as are duplicate phrases where all components match an already existing phrase.
4 Phrase weights are assigned as a function of the weights of the individual phrase components; if the term weights are restricted to values between 0 and 1, the phrase weight can be defined as the product of the individual component weights.

The phrase-generation process is illustrated for a sample sentence in Table 4-1. Some of the original words, such as "in," "need," "of," and "require," are deleted following a comparison with the entries contained in a stop list. The remaining words are transformed into word stems: this generates INFORM from "information" and RETRIEV from "retrieval." A maximum distance of four is assumed between phrase components in the text, leading to the generation of the seven phrases listed in Table 4-1c.

The phrase-formation process increases the specificity of the content identifiers attached to queries and documents. The converse operation consists in using thesaurus classes for content identification. A thesaurus contains groupings of similar or related terms into term classes. The use of thesauruses is

"People in need of information require effective retrieval services"

(a)

"PEOPLE INFORM EFFECT RETRIEV SERVICE"

(b)

PEOPLE	INFORM	EFFECT	RETRIEV
INFORM	EFFECT	EFFECT	SERVICE
INFORM	RETRIEV	RETRIEV	SERVICE
INFORM	SERVICE		

(c)

Table 4-1 Phrase-generation process. (a) Original sentence. (b) Word stems generated from original sentence. (c) Word pairs (phrases) generated assuming maximum component distance of four.

often advocated for purposes of synonym recognition: when the query specifies "manufacture" and the document contains "production," a term match is obtainable by including both terms in a common thesaurus class. Because the frequency of assignment of a thesaurus class is approximately equal to the sum of the frequencies of the individual terms in the class, thesauruses are most useful for the classification of the low-frequency terms that need to be broadened. Since the automatic term classification (thesaurus) construction methods are all based in one way or another on the computation of similarities between terms, thesauruses are not easily generated automatically. Normally, the term similarities would be obtained by computing similarity coefficients between the term vectors (columns) of the term assignment array (see Fig. 4-1). In effect the similarity between two terms will then depend on co-occurrences of the terms in the documents of a collection. Unfortunately, the terms of most interest for thesaurus construction purposes are those whose overall occurrence frequency in a collection is low. These terms do not co-occur very often in the same documents, and their computed similarity measure must be expected to be very low and may not then furnish an accurate indication of term relationship.

In practice, it becomes necessary either to use a manually constructed thesaurus that obeys the thesaurus construction principles described in the previous chapter, or else to use an automatic thesaurus construction method in which the grouping criteria are relatively weak [14,15]. The well-known *single-link* classification method appears to be most useful in this connection [16]. The single-link process is based on a computation of the term-term similarities for all term pairs. The process then proceeds iteratively as follows:

1 For each term pair ($TERM_i$, $TERM_j$) whose similarity exceeds a given threshold, an attempt is made to add a third term, $TERM_k$, to the group by computing the similarity between $TERM_k$ and each of the original terms; the new term is added whenever its similarity with *at least one* of the original terms exceeds a stated threshold.

2 The process is then repeated for term triples, quadruples, etc., by adding a new term whenever its similarity with one of the original terms exceeds the stated threshold.

An example of this process is shown in Fig. 4-9. The significant term-pair similarities are shown in Fig. 4-9a in tabular form. Figure 4-9b presents the same information in graph form, where the nodes of the graph designate terms, and the branches between nodes represent the corresponding term similarities. The final term clusters are given in Fig. 4-9c. The initial cluster is obtained by first forming the class $\{T_1, T_2\}$. To this are added terms T_3 and T_7 because of the similar term pairs (T_2, T_3) and (T_2, T_7). This produces a new class $\{T_1, T_2, T_3, T_7\}$. This group can be increased to its full size by adding T_8 because of the connection between T_3 and T_8. A second class is formed of terms T_4, T_5, and T_6 in view

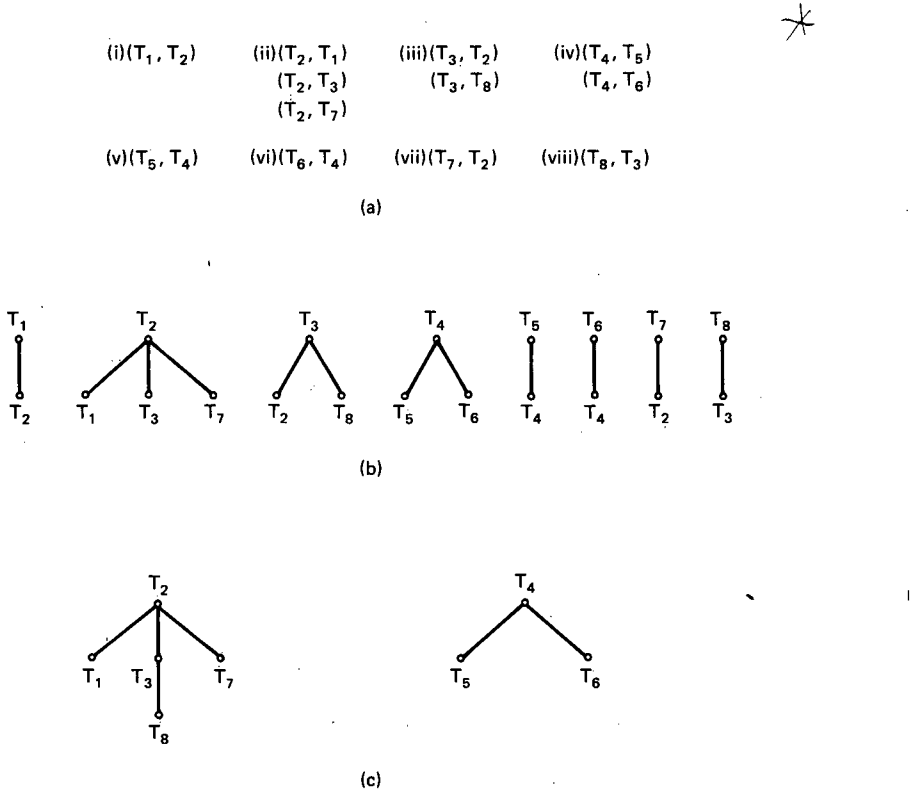


Figure 4-9 Single-link cluster example. (a) Initially available term pairs. (b) Corresponding graphical representation. (c) Corresponding single-link clustering.

of the existence of the similar pairs (T_4,T_5) and (T_4,T_6). For the example under consideration the thesaurus classes produced are as follows: $\{T_1,T_2,T_3,T_7,T_8\}$ and $\{T_4,T_5,T_6\}$.

The single-link cluster process requires of the order of n^2 term comparisons to classify n terms, since each term is associated with at most $(n-1)$ other terms. This method has been used to cluster large collections of items [17], and it may be applicable to the general term classification problem because the number of terms does not normally exceed several tens of thousands.

For document clustering, where the file size may comprise hundreds of thousands or even millions of items, less expensive methods may turn out to be more appropriate. A summary of the SMART automatic indexing system is presented in the flowchart of Fig. 4-10.

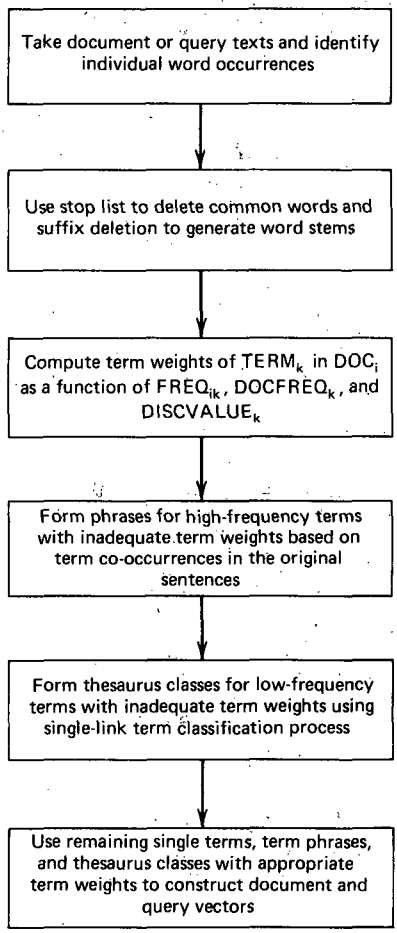


Figure 4-10 Typical SMART automatic indexing process.

*B Automatic Document Classification

It was seen earlier that the preferred file organization in the SMART environment is a clustered collection where the incoming queries are compared first with the centroids of certain document classes and then with the individual documents for those classes whose centroid similarity with the queries is sufficiently large. The clustered file organization lends itself to rapid collection searches and to fast and possibly effective retrieval strategies because all the related documents in a given class are in principle retrievable in a single file access. Furthermore, a clustered file is updated easily, because new documents can be treated like incoming queries in that comparisons with the existing centroids lead to the incorporation of new items into the most closely fitting classes.

The available clustering methods fall into two main types according to whether an initial set of classes already exists or on the contrary new classes must be constructed for a given collection of items. When document sets are to be clustered, a prior classification is not generally available; it is then necessary to construct a new classification for the given set of items. A second clustering criterion distinguishes the *hierarchical grouping* methods which utilize a full document-document similarity matrix specifying the similarity for all document pairs, from the *iterative partitioning* procedures where a rough classification is first generated which is then refined in several steps.

The hierarchical grouping methods which include the previously mentioned clique and single-link procedures are theoretically more satisfying than the alternative partitioning methods because in these methods each item is effectively considered as a possible "seed point" around which a new class is to be built, whereas the classes formed with the iterative partitioning process depend on the strategy used for the initial cluster generation. A form of iterative partitioning is nevertheless used with the SMART system because the number of required document comparisons to cluster n items is of the order of n log n, whereas n^2 comparisons may be needed to obtain the document pair similarities needed by the hierarchical grouping strategies. When the collections are large, a relatively inexpensive clustering method seems essential.

The SMART "single-pass" clustering proceeds in a bottom-up fashion by considering the records one at a time in arbitrary order, while attempting to group them into clusters [18,19]. The first item is initially identified with cluster one. The next item is compared with cluster one and merged with it if found to be sufficiently similar. If the new item is not similar to any already existing cluster, a new cluster is generated. Subsequent items are compared with all existing cluster centroids and entered into classes whenever the centroid similarity is sufficiently large. When a new item is entered into a cluster, the corresponding cluster centroid must be redefined by incorporating terms from the new term vector into the original cluster centroid.

In principle, the single-pass clustering process should serve to assign each item to at least one cluster, and the classification should be complete after one

pass through the file. In practice, the resulting classes may not be usable for search purposes without additional refinements. Several problems may arise:

1 The number of clusters produced by the initial pass may become excessively large, implying that a query submitted to the system may have to be compared with a very large number of centroids before access to the individual records is actually obtained.

2 The size of certain clusters may become too large, particularly if a great many records in a collection cover fairly homogeneous subject areas.

3 Alternatively, the cluster size may be very small, and could indeed be limited to a single record in cases where so-called loose records exist that do not match any other records in the collection.

4 The overlap among clusters, that is, the number of items jointly contained in more than one cluster may be too large or too small.

To respond to these eventualities, controls must be introduced to regulate cluster size, cluster overlap, and number of clusters generated, and to handle any "loose" items remaining at the end of the first pass. The loose items actually present no severe problem, since they can naturally be merged with the closest existing clusters. To control the size and the number of clusters, a *cluster splitting* operation is carried out whenever a given cluster size exceeds some preestablished threshold. This is done by generating a term-term similarity matrix for all terms located in the excessively large clusters, and using a new *local* clustering operation to produce two or more new centroids replacing each centroid originally attached to a cluster that had grown too large.

The cluster splitting operation is illustrated in the example of Fig. 4-11 where the assumption is that no class may contain more than four elements. The initial state consists of four clusters, each containing between two and four records. These four centroids are themselves grouped into a supercluster with supercentroid S as shown in Fig. 4-11a. If a new record is added to cluster A, an illegal situation arises, since the cluster size is assumed limited to four elements. The A centroid must then be split thereby creating two new centroids A′ and A″ as shown in Fig. 4-11c. At this point the supercluster S is no longer viable since it now contains five elements. This is remedied by splitting S into S′ and S″, thereby creating a new hypercluster H included in Fig. 4-11d. The cluster splitting process thus propagates upward in the "cluster tree," starting with the lowest level clusters and moving upward as shown in the example.

A simplified flowchart of the single-pass cluster generation and search process is shown in Fig. 4-12. Generation and search differ in substance only for the lowest-level centroids: during cluster generation a new record must be added to the lowest level of the cluster tree and the cluster splitting routine may need to be invoked; during normal searching, on the other hand, the low-level clusters simply lead to the individual records on the lowest level. The program of Fig. 4-12 maintains a list of centroids to be split. When that list is not empty, the splitting routine is invoked following the placement of each incoming item.

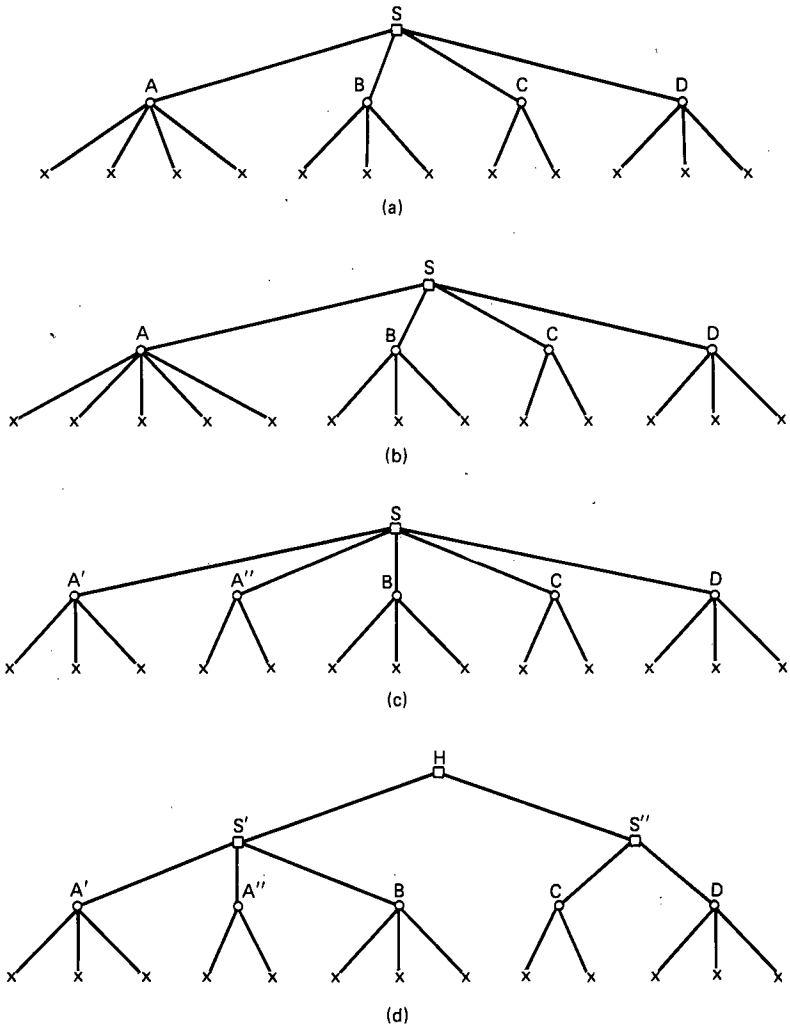


Figure 4-11 Example of cluster splitting process. (a) Initial state of cluster structure. (b) Addition of one more item to cluster A. (c) Splitting cluster A into two pieces A′ and A″. (d) Splitting supercluster S into two pieces S′ and S″.

The cluster generation method may be adapted to special retrieval requirements: when a premium is placed on search precision and the retrieval of nonrelevant items must be avoided, the cluster structure should consist of a large number of small, disjoint clusters on the lowest level of the search tree. When the recall proves more important a smaller number of larger, partly overlapping clusters should prove more effective. Suitable adjustments in the thresholds that control the clustering process can be used to satisfy varying retrieval requirements.

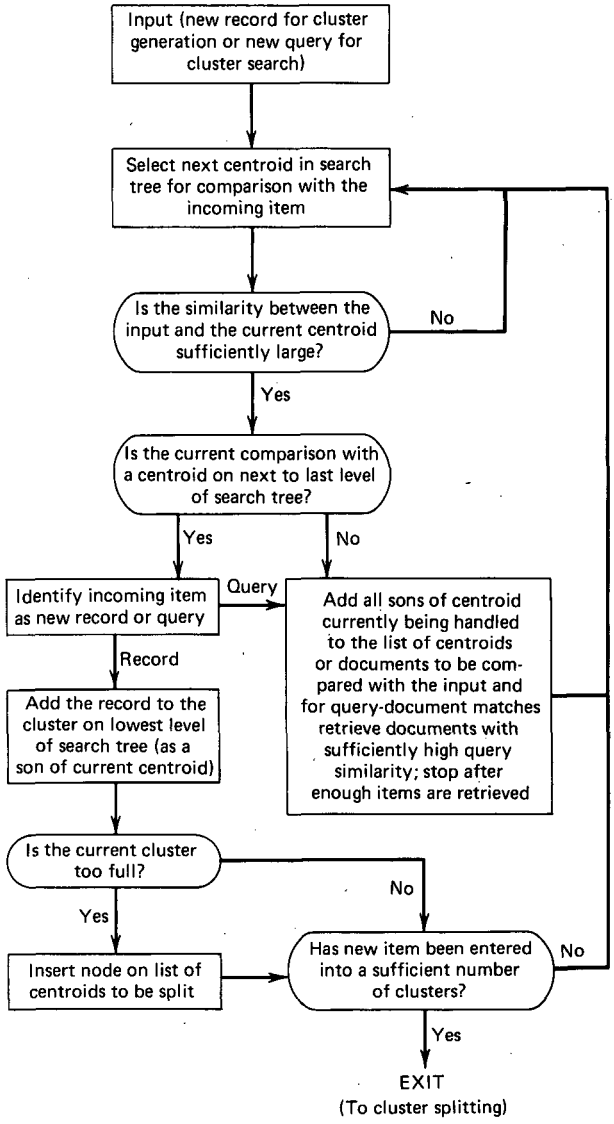


Figure 4-12 Cluster generation and search. (*Adapted from reference 18.*)

*C Relevance Feedback Operations

Most modern on-line search and retrieval environments make provisions for the utilization of flexible search procedures that can generate satisfactory query statements and useful retrieval output in several small steps. An iterative search system is typically implemented by initially submitting a tentative

query, and then using system facilities to improve these query statements and the resulting set of retrieved items.

The methods used to construct improved query formulations fall into two broad classes: those which are applied before a file search is actually carried out, and those which depend on the prior retrieval of some retrieved items. In the former case, it is possible to use displayed vocabulary excerpts, term frequency information, and citations to so-called source documents previously identified as relevant by the user to construct improved queries. More specifically, a stored thesaurus of terms can be used to obtain synonym lists for the terms originally included in a search formulation. In addition, frequency data about the number of documents to which a term is assigned can also be stored in the thesaurus to provide an indication of the potential effect of the term for search purposes. When relevant source documents are known in advance, a query might simply be replaced by the terms assigned to such documents in the hope of retrieving additional documents similar to those originally identified as relevant.

An alternative query alteration process is based on the execution of an initial search operation and an initial retrieval of certain stored documents. The display of information relating to these items such as the titles or abstracts of the previously retrieved documents is then used to modify the query statements, normally by adding terms from documents that appear relevant to the user and by deleting terms included in the items that appear useless. This produces new queries whose resemblance to the relevant documents is greater than before while their resemblance to the nonrelevant items is smaller, as suggested in the example of Fig. 4-3. In the SMART system, this process has been called "relevance feedback." It has been shown experimentally that the relevance feedback process can account for improvements in retrieval effectiveness of up to 50 percent in precision for high-recall (broad) searches, and of approximately 20 percent in precision for low-recall searches [20].

Consider first the unrealistic situation where the complete set of relevant documents D_R is known in advance, and hence also the complete set of nonrelevant documents $D_{N-R} = N - D_R$. (In such a case, there is no point in submitting a search request because the known relevant documents can then simply be retrieved from the file without performing a file search.) It can be shown that in such a situation the optimal query which is best able to distinguish the relevant from the nonrelevant documents is a vector obtained by taking the difference between the set of relevant and the set of nonrelevant documents, respectively. More formally, if $TERM_{ik}$ represents the value, or weight, of term k in document i as before, then the average value of term k in the set of R relevant documents is $\left(\frac{1}{R}\right)\left(\sum_{i \epsilon D_R} TERM_{ik}\right)$. Similarly, the average value of term k in the set of $N - R$ nonrelevant documents is $\left(\frac{1}{N-R}\right)\left(\sum_{i \epsilon D_{N-R}} TERM_{ik}\right)$. The value of term k in the optimal query Q_{opt} is then defined as

$$(Q_{opt})_k = C\left(\frac{1}{R}\sum_{i\epsilon D_R} TERM_{ik} - \frac{1}{N-R}\sum_{i\epsilon D_{N-R}} TERM_{ik}\right) \quad (5)$$

where C is a constant [21]

In practice the sets of relevant documents D_R and of nonrelevant items D_{N-R} are not known in advance. However, the relevance judgments obtained from the user population for previously retrieved relevant documents furnish approximations to the sets D_R and D_{N-R}. The assumption is then made that a near-optimal query can be obtained by taking the difference between the subsets of relevant and of nonrelevant items known at any particular time. By repeating the relevance feedback operation several times, and retrieving at each point a new set of documents with an improved query formulation, one may eventually obtain a reasonable approximation to the actual set of relevant and nonrelevant items.

Several formulations are now possible for a new improved query vector Q′, starting with an initial query formulation Q and a set $D_{R'}$ of R′ documents identified as relevant as well as a set $D_{N'}$ of N′ nonrelevant documents, where $D_{N'}$ and $D_{R'}$ are initial representations of the actual set of nonrelevant and relevant documents, respectively. First it is possible by analogy to the optimal formula of expression (5) to define the new query as simply the difference between the known sets of relevant and nonrelevant items, respectively:

$$Q' = C\left(\frac{1}{R'}\sum_{i\epsilon D_{R'}} DOC_i - \frac{1}{N'}\sum_{i\epsilon D_{N'}} DOC_i\right) \quad (6)$$

DOC_i again represents the vector for the ith document. In formula (6) the information contained in the original query Q is not used. In practice the original query formulation may contain important information; hence an improved feedback strategy may be produced by using the original query and adding terms from the relevant documents, and/or deleting terms from the nonrelevant ones:

$$Q' = \alpha Q + \beta\left(\frac{1}{R'}\sum_{i\epsilon D_{R'}} DOC_i\right) - \gamma\left(\frac{1}{N'}\sum_{i\epsilon D_{N'}} DOC_i\right) \quad (7)$$

where α, β, and γ are suitable constants. Expression (7) specifies a new query as the vector sum of the old query plus the weighted difference between the average of the known relevant and the average of the known nonrelevant items.

The operations specified by expression (7) are illustrated in the example of Fig. 4-13, where a query Q comprising five terms is shown together with a relevant document D_1 and a nonrelevant document D_2. The constants α, β, γ are assumed to take on values of 1, 1/2, 1/4, respectively. The new query derived in Fig. 4-13c exhibits increased weights for terms 1 and 3, a decreased weight for term 5, and a new term (term 2) not present in the original query. Correspond-

	Term 1	Term 2	Term 3	Term 4	Term 5
Q = (	5 ,	0 ,	3 ,	0 ,	1)
D_1 = (	2 ,	1 ,	2 ,	0 ,	0)
D_2 = (	1 ,	0 ,	0 ,	0 ,	2)

(a)

$$Q' = Q + \frac{1}{2}\left(\sum_{D_{R'}} D_i\right) - \frac{1}{4}\left(\sum_{D_{N'}} D_i\right)$$

$$Q' = (5,0,3,0,1) + \frac{1}{2}(2,1,2,0,0) - \frac{1}{4}(1,0,0,0,2)$$

$$= 5\frac{3}{4}, \frac{1}{2}, 4, 0, \frac{1}{2}$$

(b)

Assume $S(Q,D_i) = \sum_{j=1}^{t} (Q_j, D_{ij})$

$S(Q,D_1) = (5\cdot2) + (0\cdot1) + (3\cdot2) + (0\cdot0) + (1\cdot0) = 16$

$S(Q',D_1) = (5\frac{3}{4}\cdot2) + (\frac{1}{2}\cdot1) + (4\cdot2) + (0\cdot0) + (\frac{1}{2}\cdot0) = 20$

$S(Q,D_2) = (5\cdot1) + (0\cdot0) + (3\cdot0) + (0\cdot0) + (1\cdot2) = 7$

$S(Q',D_2) = (5\frac{3}{4}\cdot1) + (\frac{1}{2}\cdot0) + (4\cdot0) + (0\cdot0) + (\frac{1}{2}\cdot2) = 6\frac{3}{4}$

(c)

Figure 4-13 Relevance feedback operation. (a) Originally available query Q, relevant document D_1, nonrelevant document D_2. (b) Query alteration. (c) Query-document similarities.

ingly, the query-document similarity obtained with the new query Q′ is larger than before for document D_1 and smaller than before for D_2.

In practice one finds that the information contained in the relevant documents is more valuable for query reformulation purposes than the terms which originate in the nonrelevant items. The reason is that the set of relevant documents with respect to a given query may be expected to be located in a reasonably homogeneous area of the document space, as in the example of Fig. 4-7. The addition to the query of terms from these relevant items will then produce a definite movement of the query in the direction of these relevant items (see Fig. 4-3). The set of nonrelevant items, on the other hand, is normally much more heterogeneous. The average nonrelevant item may therefore be located almost anywhere in the document space, and subtraction of the corresponding terms

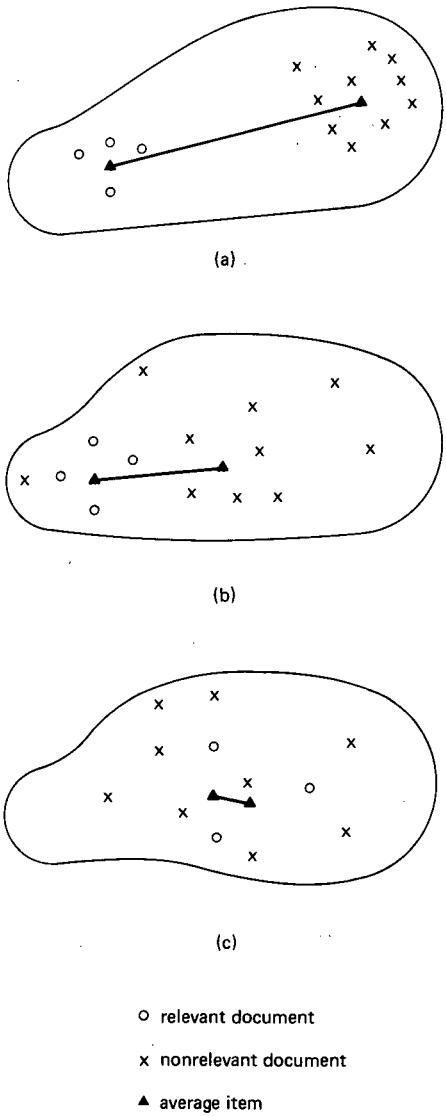


Figure 4-14 Relevance feedback environment. (a) Ideal relevance feedback situation. (b) Tight relevant document set, but loose nonrelevant set. (c) Intermingled relevant and nonrelevant documents.

removes the query from that area of the space without specifying a definite alternative direction. The feedback equation [expression (7)] should then be used with a smaller weight for γ than for β. Alternatively γ might be specified as 0, thereby creating a *positive feedback* strategy. A third alternative consists in using only a small set of nonrelevant documents by including in the feedback equation only one or two documents—for example, the nonrelevant documents retrieved earliest in a given search (those exhibiting the lowest ranks when the documents are arranged in decreasing order of the query-document similarity).

Since the difference between the average relevant and the average nonrelevant documents is equivalent to the distance between the corresponding vectors in the vector space, it is not surprising that the formal as well as experimental test results show that the retrieval operation is most effective when the relevant documents as well as the nonrelevant documents are tightly clustered, and the difference between the two groups is as large as possible. Such a situation is represented in Fig. 4-14a.

The relevance feedback operation is less favorable in the more realistic case where the set of nonrelevant documents covers a wider area of the space. The corresponding distance between the average relevant and nonrelevant items is much smaller in Fig. 4-14b than in Fig. 4-14a. Finally, the situation is distinctly unfavorable when relevant and nonrelevant are intermixed as shown in Fig. 4-14c. That situation represents a failure of the basic assumption of the SMART document analysis, namely that document content can be represented for retrieval purposes by the term occurrence vectors. Additional information must then be added to the document vectors to distinguish those which are useful from the others. Some approaches to this problem are mentioned in the next section.

The experimental evidence available for relevance feedback indicates that one or two feedback operations are quite effective in raising retrieval performance. Following the second query reformulation a state of diminishing returns sets in and not much further improvement can be expected.

*D Dynamic Document Space

The query alteration process described in the previous subsection was based on information obtained from the user population in the course of the normal retrieval process. In a system where customer intelligence is available, an attempt can also be made to improve the document vector representation (instead of only the query vectors) by incorporating into the document representations new information obtained from the users during the search operations. One possibility consists in adding to the originally available document terms new information derived from relevance assessments furnished by the users in the course of a retrieval operation.

Specifically, when a number of documents retrieved in response to a given query are labeled by the user as relevant, it is possible to render these documents more easily retrievable in the future by making each item somewhat more similar to the query used to retrieve them. Analogously, retrieved documents labeled as nonrelevant are rendered less easily retrievable by being shifted away from the query. Following a large number of such interactions, documents which are wanted by the users will have been moved slowly into the active portion of the document space—that part in which a large number of user queries are concentrated, while items which are normally rejected will be located on the periphery of the space. Eventually, such items could be discarded.

By analogy to the relevance feedback operations, the dynamic document

vector alteration can be carried out by constructing a new document representation DOC_i' from the old document representations DOC_i and the terms contained in query Q. The query terms are added to the original document vector using the positive weighting factors α, β, γ, δ, and ϵ:

1 For documents designated as relevant, which must be rendered more similar to query Q:
   - **a** A query term *not* present in the document is added to the document with a weighting factor of α.
   - **b** A query term also present in the document receives increased importance by incrementing its weight by a factor β.
   - **c** A document term not present in the query is decreased in weight by a factor $-\gamma$.

2 For documents designated as nonrelevant which must be rendered less similar to the query:
   - **a** Document terms also present in the query are rendered less important by decreasing their weight by a factor of $-\delta$.
   - **b** Document terms absent from the query are increased in weight by a factor ϵ [22,23].

Several laboratory tests were carried out to test the foregoing dynamic document space alteration methods. In each case, a collection of user queries was first used to generate a modified document space. A new query set, distinct from the original one, could then be processed against both the originally available document space and the modified space. Improvement in recall and precision of from 5 to 10 percent could be detected for the modified space compared with the original space. One would expect that the document vector modification process carried out for a given time period with a particular user population would eventually produce an equilibrium position where documents important to the users could become easily retrievable whereas extraneous documents would be easily rejected. Such a conjecture remains to be verified in practice.

4 AUTOMATIC ENHANCEMENTS OF CONVENTIONAL RETRIEVAL

*A Document Ranking and Term Weighting

Various retrieval systems and procedures have been implemented over the last few years that are based in one way or another on the SMART model [24,25]. Possibly the most useful of these from a practical viewpoint are systems that preserve as much as possible the conventional processing methodology while adding enhancements to simplify the retrieval operations and to improve system effectiveness. The main characteristics of currently existing retrieval technologies are the inverted file organizations and the use of Boolean query formulations. This suggests that improved retrieval services could be obtained by combining the basic inverted file systems with SMART-like "back-end" proce-

dures, designed to overcome some of the disadvantages of conventional systems.

Among the methods usable for improving the output produced by conventional file systems are the incorporation of weighted instead of binary terms and the presentation of the retrieved output in decreasing order of the query-document similarity. The generation of *ranked* document output increases user satisfaction and retrieval precision. Furthermore, the ranked retrieval can substantially enhance the chances of success of the relevance feedback process by bringing the relevant items to the user's attention early in the search; this in turn leads to the construction of useful feedback queries.

Ranked retrieval and relevance feedback can in principle be implemented in binary indexing systems where the index terms are either present or absent from the document and query vectors. However, the documents are easier to rank when *weighted terms* are assigned to the queries, and possibly also to the documents, because a composite query-document similarity coefficient can then be computed for each query-document pair based on the weights of matching query-document terms.

In a system such as SMART where each document is represented by a vector of terms and each vector is stored as a complete entity, the use of term weights presents no conceptual problem, because a term weight can simply be listed with each term identification in the corresponding term vector. Thus a particular document i dealing with "fruit" might be listed as

$$DOC_i = (APPLE,4;\ PEAR,3;\ \ldots\ ;\ PLUM,1)$$

to indicate that the document deals rather more with pears than with plums, and even more with apples. In inverted file systems, the term weighting information must be included in the various inverted lists. Thus the term list for a given term k must now be expanded to include not only the document references for documents to which term k is assigned but also the particular weights for that term in the various documents. A sample inverted file of that kind is shown in Table 4-2.

Given some particular query Q_j, a file organization such as that of Table 4-2 now makes it possible to compute a similarity measure between the query and the document terms. For each term k included in query j, the corresponding term list is extracted from the inverted file, and the weights of terms that occur in both query j and document i are appropriately combined to produce a similarity measure between the given query j and each document that shares

Table 4-2 Typical Inverted Term Lists with Weights

$TERM_1$	DOC_1,WT_{11}; DOC_j,WT_{j1}; . . . ; DOC_m,WT_{m1}
$TERM_2$	DOC_1,WT_{12}; DOC_k,WT_{k2}; . . . ; DOC_n,WT_{n2}
⋮	
$TERM_s$	DOC_j,WT_{js}; DOC_m,WT_{ms}; . . . ; DOC_p,WT_{ps}

terms with the query. Consider a query consisting of three terms r, s, and t, and let the similarity measure between the query j and document i be defined as the sum of the products of the matching terms [that is, the numerator of the cosine formula of expression (1)]. To compute the similarity measure between the given query and document, it suffices to enter first the inverted list for term r and pick out $TERM_{ir}$ representing the weight of term r in document i. This leads to the computation of the product $QTERM_{jr} \cdot TERM_{ir}$. A subsequent access to the inverted lists for terms s and t makes it possible to add to the earlier product the terms $QTERM_{js} \cdot TERM_{is}$ and $QTERM_{jt} \cdot TERM_{it}$.

To compute the cosine measure of expression (1) an additional normalization factor consisting of the sum of the squares of all term weights included in each document and each query is needed. This makes it necessary to provide for each document i the factor $\sum_{k=1}^{t} (TERM_{ik})^2$. These factors can be computed in advance for each document and stored in a special "document length" file which is accessed for all documents that have a nonzero similarity with the query. The expanded inverted document reference lists together with the document length file permit the computation of the full cosine similarity measure for all documents sharing one or more terms with the query, followed by the ranking of documents in decreasing order of the query-document similarity.

The SIRE (Syracuse information retrieval experiment) system includes the normal inverted file processing facilities used for the Boolean query operations, as well as the expanded term weighting operations that provide ranked document output [26–28]. The SIRE file organization is shown in Fig. 4-15, and the corresponding processing chain is presented in simplified form in Fig. 4-16. In essence, the SIRE processing chain consists of two main sections. Initially, a Boolean query formulation is processed in the conventional manner. This step identifies all documents whose term assignment precisely matches the query formulation. This subset of documents is then further processed by computing a cosine similarity measure between each document and a "flattened" query consisting of all the original query terms connected by Boolean OR operators. Thus an original query formulated as (A AND B) OR (C AND D) becomes A OR B OR C OR D, or simply the query vector (A,B,C,D).

In the SIRE system the users are not expected to assign weights to the terms. Instead the weight of term k in document i is defined as $FREQ_{ik}$, that is, as the frequency of occurrence of the term in the document. Correspondingly, the document length needed in the denominator of the cosine computation is obtained as the sum of the squares of the individual term frequencies in the document. It is easy to see how the file organization of Fig. 4-15 leads to the proper computation of the cosine measure. A directory to the inverted lists is used to gain access to the document reference numbers and the weights (term frequencies) for all query terms. This produces the numerator of the cosine for each relevant document (each document retrieved by the Boolean query) and the corresponding query. The document length is next obtained for each document from the document length list (Fig. 4-15c). Following the computation of the

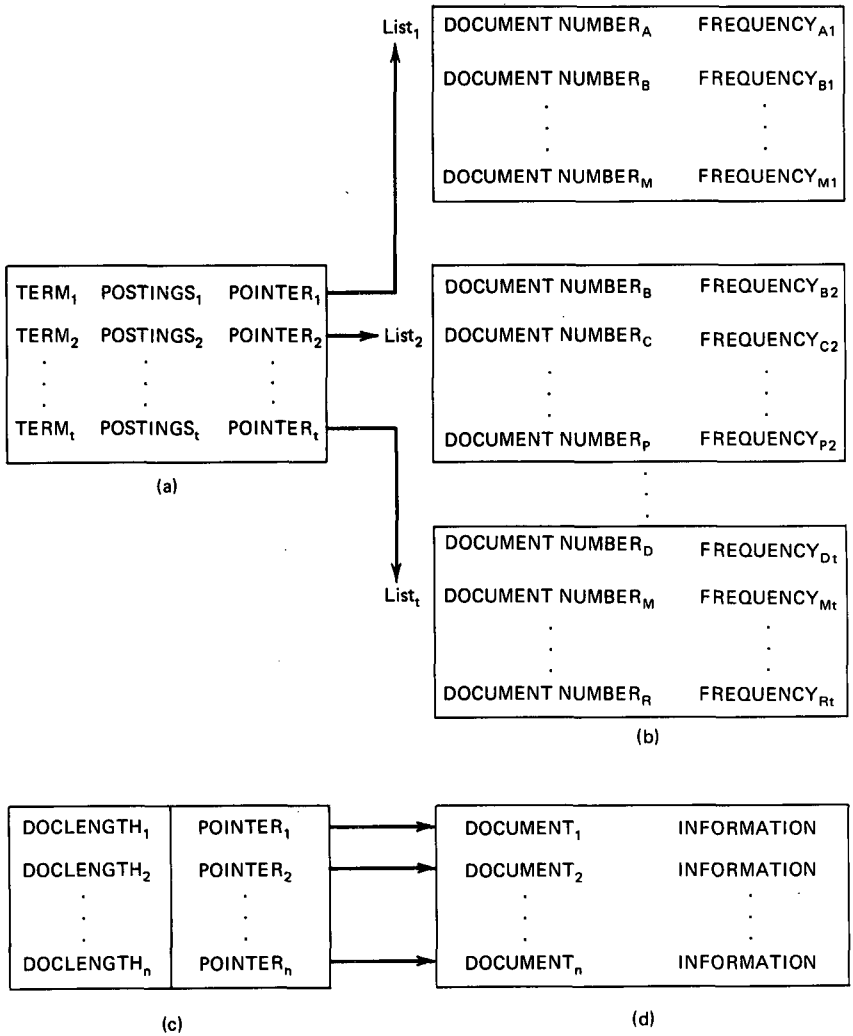


Figure 4-15 SIRE file organization. (a) Directory to inverted lists. (b) Inverted document reference lists. (c) Document length list. (d) Main document file.

query-document similarities, the output information may be extracted from the main document file in decreasing order of the cosine measure by using the file pointers provided for that purpose.

In the SIRE system, the cosine computations and the ranking algorithms are completely divorced from the Boolean retrieval operations. Alternatively, it is possible to give up the Boolean operations completely and process query formulations expressed simply as sets of weighted query terms. In the BROWSER system, a file organization substantially equivalent to that shown in Fig. 4-15 is used together with term weights based on an inverse document

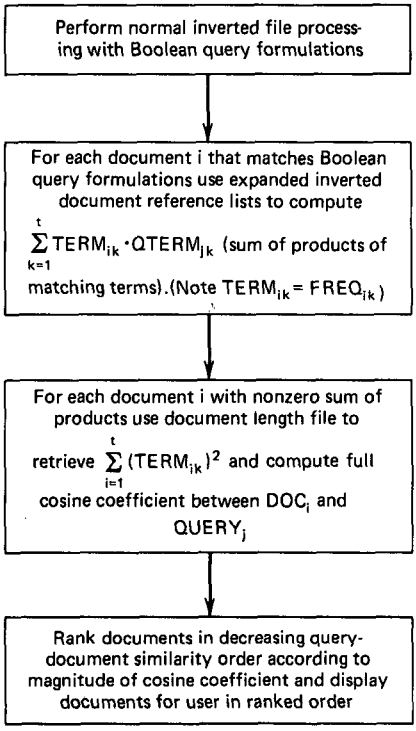


Figure 4-16 SIRE processing chain.

frequency factor [29]. CITE is another system that uses the conventional inverted file technology with term weights, document ranking, and relevance feedback based on the methods described earlier [30].

Another possibility for combining Boolean query processing with term weighting systems is to allow weighted terms directly as part of the normal Boolean formulations. Unfortunately, this approach raises a number of difficult problems, because the normal two-valued logic used to process Boolean queries assumes that a term is either present in or absent from a document or query identification. When the Boolean system is extended to include term weight or importance factors, it becomes necessary to interpret the meaning of compound expressions consisting of weighted terms and Boolean operators such as A_a AND B_b, where a,b represent the weights for terms A and B, respectively. In so doing one would like to define matching functions that resemble the normal Boolean operations and produce equal output for distinct query formulations that are logically equivalent [31,32].

It is possible to design retrieval models that furnish unambiguous retrieval output for weighted Boolean queries. The theory of fuzzy sets provides a model in which the weight of term k in document i represents the degree to which the document is a member of the set of documents indexed by the term k [33]. A weight of 1 then indicates that DOC_i is a full member of the document set in-

dexed by $TERM_k$, a weight of 0 says that the document does not belong to the set, and an intermediate weight designates a partial degree of membership. The effect of compound Boolean expressions can also be defined in the fuzzy set model. Whether weighted Boolean queries will ever become popular in retrieval remains to be seen.

*B Retrieval through Man-Machine Dialogue and Local Clustering

The previously described relevance feedback process is based on a somewhat rudimentary type of user-system interaction, because the system generates query formulations based on relevance assessments returned by the user for previously retrieved documents. There exist interactive systems where the user is expected to take a much more active role in the retrieval operations. Thus users may be asked to carry out the query reformulations by individually choosing good terms for incorporation into the queries; the corresponding terms may then be chosen from among a set of potentially useful terms displayed by the system. Users may also be asked to assign positive and/or negative weights to the terms considered for incorporation into the query. Finally the users may be asked to assign weights (positive or negative) to displayed documents that may or may not have been retrieved in earlier searches [34].

Relevance judgments can be used as a basis for query reformulation also in conventional retrieval environments where Boolean query statements are used to retrieve documents manually indexed by keywords. The user feedback process devised for the European Community retrieval service consists of the following main steps [35]:

1 Relevance assessments are obtained for some of the documents retrieved in response to an initial search request.

2 The set of terms used to index some of the items known as relevant is examined [for example, (A AND B AND C AND D) and also (E AND F AND G)].

3 Some terms from the query statements chosen under step 2 are removed so as to broaden the resulting search statements [for example, statements (A AND B AND D) and also (E AND F) are constructed].

4 These shortened queries are used as new search statements to retrieve additional documents; the relevance of some of these newly retrieved documents is then assessed.

5 For each of the new query statements a "query quality factor" is computed as the ratio between the new relevant items retrieved divided by the new nonrelevant items retrieved.

6 Those partial queries with sufficiently high query quality factors are chosen and a final feedback query is constructed by inserting OR connectives between the corresponding partial query formulations [for example, the new statement used could be (A AND B AND D) OR (E AND F)].

7 The newly constructed query is used for search purposes, and the process is repeated if desired.

Additional feedback techniques incorporating slight variations of such a process can easily be devised.

One of the virtues of the relevance feedback and related query reformulation methods is the *local* nature of the operations involved; normally only the previously retrieved documents are used, rather than the whole document set. Such considerations lie at the root of a number of *local clustering* systems designed to improve the final search output. A standard inverted file search is used as a fast preliminary step, and the automatic classification procedures previously described then serve to cluster the (local) set of documents retrieved in response to particular search efforts in the hope of improving the final search output. The corresponding document classes can be used to determine a specific ranking order in which the output items can be brought to the user's attention. By displaying together whole groups of related documents and bringing them to the user's attention simultaneously, the choice of new terms to be incorporated into a feedback query may also be simplified [36].

Local clustering operations can be applied to terms as well as to documents. This produces a *local thesaurus* specifically applicable to each query which is obtained by grouping terms extracted from previously retrieved documents. A typical query reformulation process based on local term clustering might be carried out as follows [37,38]:

1 Relevance assessments are obtained for certain documents retrieved in earlier search operations.

2 Terms from the documents identified as relevant are ranked in decreasing order of relative frequency (frequency of occurrence in the relevant retrieved documents divided by total frequency in the collection), and terms with large relative frequency are used for query reformulation purposes.

3 Alternatively, or in addition, term similarity coefficients may be obtained for pairs of terms occurring in the relevant documents, the size of the coefficients being dependent on common occurrence patterns in the respective documents or document sentences.

4 Terms with large enough similarity coefficients are then clustered, and each original query term is considered as the kernel of a cluster of related terms to be used for query reformulation purposes.

It is not hard to generate extensions and refinements of the local clustering operations previously described. Thus local term association procedures can in principle also incorporate syntactic considerations where syntactic relationships between terms lead to the generation of term groups [39,40]. Document grouping methods, on the other hand, might utilize the similarities not only between the assigned terms but also between bibliographic citations shared jointly by a number of documents [41]. As the use of computers for document processing becomes more widespread, one may expect that the query refinement procedures based on user-system cooperation and on local clustering operations will also become more widespread.

REFERENCES

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

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

[3] G. Salton and M.E. Lesk, The SMART Automatic Document Retrieval System—An Illustration, Communications of the ACM, Vol. 8, No. 6, June 1965, pp. 391–398.

[4] G. Salton, A. Wong, and C.S. Yang, A Vector Space Model for Automatic Indexing, Communications of the ACM, Vol. 18, No. 11, November 1975, pp. 613–620.

[5] 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.

[6] G. Salton, Recent Studies in Automatic Text Analysis and Document Retrieval, Journal of the ACM, Vol. 20, No. 2, April 1973, pp. 258–278.

[7] G. Salton, A New Comparison between Conventional Indexing (MEDLARS) and Automatic Text Processing (SMART), Journal of the ASIS, Vol. 23, No. 2, March–April 1972, pp. 75–84.

[8] J.B. Lovins, Development of a Stemming Algorithm, Mechanical Translation and Computational Linguistics, Vol. 11, No. 1–2, March and June 1968, pp. 11–31.

[9] A. Tars, Stemming as a System Design Consideration, ACM SIGIR Forum, Vol. 11, No. 1, Summer 1976, pp. 9–16.

[10] G. Salton, C.S. Yang, and C.T. Yu, A Theory of Term Importance in Automatic Text Analysis, Journal of the ASIS, Vol. 26, No. 1, January–February 1975, pp. 33–44.

[11] K. Sparck Jones, A Statistical Interpretation of Term Specificity and Its Application in Retrieval, Journal of Documentation, Vol. 28, No. 1, 1972, pp. 11–21.

[12] G. Salton and C.S. Yang, On the Specifications of Term Values in Automatic Indexing, Journal of Documentation, Vol. 29, No. 4, December 1973, pp. 351–372.

[13] G. Salton and A. Wong, On the Role of Words and Phrases in Automatic Text Analysis, Computers and the Humanities, Vol. 10, 1976, pp. 69–87.

[14] G. Salton, Experiments in Automatic Thesaurus Construction for Information Retrieval, Information Processing 71, North Holland Publishing Company, Amsterdam, 1972, pp. 115–123.

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

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

[17] W.B. Croft, Clustering Large Files of Documents Using the Single Link Method, Journal of the ASIS, Vol. 28, No. 6, November 1977, pp. 341–344.

[18] R.E. Williamson, Real Time Document Retrieval, Doctoral Thesis, Cornell University, Ithaca, New York, June 1974.

[19] G. Salton and A. Wong, Generation and Search of Clustered Files, ACM Transactions on Data Base Systems, Vol. 3, No. 4, December 1978, pp. 321–346.

[20] E. Ide and G. Salton, Interactive Search Strategies and Dynamic File Organization, in The SMART Retrieval System—Experiments in Automatic Document Processing, G. Salton, editor, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971, Chapter 16.

[21] J.J. Rocchio, Jr., Relevance Feedback in Information Retrieval, in The SMART Re-

trieval System—Experiments in Automatic Document Processing, G. Salton, editor, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971, Chapter 14.

[22] G. Salton, Dynamic Document Processing, Communications of the ACM, Vol. 15, No. 7, July 1972, pp. 658–668.

[23] T. Brauen, Document Vector Modification, in The SMART Retrieval System—Experiments in Automatic Document Processing, G. Salton, editor, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975, Chapter 24.

[24] N.S. Malthouse, Indexgen—Index Term Generation Heuristics, Oak Ridge National Laboratory, Report ORNL-EIS-104, Oak Ridge, Tennessee, June 1978.

[25] Informatics Inc., RADC Automatic Classification On-Line (RADCOL)—User's Manual, Final Technical Report, Rome Air Development Center, Rome, New York, May 1975.

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

[27] M.J. McGill, L. Smith, S. Davidson, and T. Noreault, Syracuse Information Retrieval Experiment (SIRE): Design of an On-Line Bibliographic Retrieval System, SIGIR Forum, Vol. 10, No. 4, Spring 1976, pp. 37–44.

[28] M.J. McGill and T. Noreault, Syracuse Information Retrieval Experiment (SIRE): Rationale and Basic System Design, Report, School of Information Studies, Syracuse University, Syracuse, New York, May 1977.

[29] J.H. Williams, BROWSER, An Automatic Indexing On-Line Text Retrieval System, IBM Federal Systems Division Report, Gaithersburg, Maryland, 1969.

[30] T.E. Doszkocs and B.A. Rapp, Searching MEDLINE in English: A Prototype User Interface with Natural Language Query, Ranked Output and Relevance Feedback, Proceedings of the ASIS Annual Meeting, Minneapolis, Minnesota, October 1979, R.D. Tally, editor, Knowledge Industry Publications, White Plains, New York, 1979, pp. 131–139.

[31] 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.

[32] W.G. Waller and D.H. Kraft, A Mathematical Model of a Weighted Boolean Retrieval System, Information Processing and Management, Vol. 15, No. 5, 1979, pp. 235–245.

[33] A. Bookstein, Fuzzy Requests: An Approach to Weighted Boolean Searches, Journal of the American Society for Information Science, Vol. 31, No. 4, July 1980, pp. 240–247.

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

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

[36] D.S. Becker and S.R. Pyrce, Enhancing the Retrieval Effectiveness of Large Information Systems, IIT Research Institute, Report PB 266 008, Chicago, Illinois, 1977.

[37] 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.

[38] T.E. Doszkocs, AID—An Associative Interactive Dictionary for On-Line Searching, On-Line Review, Vol. 2, No. 2, 1978, pp. 163–173.

[39] D.J. Hillman, Customized User Services via Interactions with LEADERMART, Information Storage and Retrieval, Vol. 9, No. 11, 1973, pp. 587–596.

[40] D. Taeuber, CONDOR—Ein Integriertes Datenbank und Informationssystem, Nachrichten für Dokumentation, Vol. 29, No. 3, 1978, pp. 127–130.
[41] H. Small, Cocitation 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.

BIBLIOGRAPHIC REMARKS

Many materials dealing with modern information retrieval appear as user manuals and reports issued by the sponsoring organizations. A summary of many existing, advanced systems is included in:

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

Additional information about the SMART system can be obtained from:

G. Salton, editor, The SMART Retrieval System—Experiments in Automatic Document Processing, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

Advanced bibliographic retrieval services are also examined in various survey papers. The following references may serve as a convenient introduction:

D.B. McCarn and J. Leiter, On Line Services in Medicine and Beyond, Science, Vol. 181, July 27, 1973, pp. 318–324.
T.E. Doszkocs, B.A. Rapp, and H.M. Schoolman, Automated Information Retrieval in Science and Technology, Science, Vol. 208, April 4, 1980, pp. 25–30.
G. Salton, Progress in Automatic Information Retrieval, Computer, Vol. 13, No. 9, September 1980, pp. 41–57.

EXERCISES

4-1 It has been claimed that the generation of ranked retrieval output enhances retrieval effectiveness and user satisfaction.
- **a** Justify the basic argument.
- **b** Contrast the methods used to rank documents at the output in the SMART and SIRE systems.
- **c** Boolean query formulations and inverted file technologies do not ordinarily produce ranked retrieval output; describe two methods that are usable to produce ranked output in a Boolean query environment.

4-2 Relevance feedback and dynamic document space modification are dual operations: in the former the query formulations are enhanced by using terms from previously retrieved documents, and in the latter the document vectors are altered by using terms from queries submitted by the user population.
- **a** How could relevance feedback and dynamic document space modification be implemented in a retrieval environment based on Boolean query formulations

and inverted file technologies; prepare a flowchart-like description for both processes in the new environment.

b Do you feel that relevance feedback and dynamic document space modification are well adapted to an inverted file organization? Carefully explain your reasons.

4-3 Prepare a program to compute the cosine similarity between a given document and query [expression (1)] given the file organization for the SIRE system shown in Fig. 4-15.

4-4 What is the purpose of cluster splitting during the generation of a clustered document collection? In the cluster splitting system represented in Fig. 4-11 a new level of centroids may be created, thereby causing an upward expansion of the cluster tree. Can you think of a cluster splitting method in which the cluster hierarchy expands in a downward direction instead of upward? What are the advantages of both systems of cluster modification for cluster generation and search?

4-5 Consider a sample document collection consisting of six documents, each represented by six terms. Assume that three of the documents are relevant to a given query and three are nonrelevant. Choose the term weights so as to model the situations shown in Fig. 4-14a, b, and c, respectively. Compute the centroid vector for the relevant and the nonrelevant items in each case and show that the distance between the centroids is largest for the case of Fig. 4-14a and smallest for the situation of Fig. 4-14c. What are the consequences in a relevance feedback system?

4-6 What problems arise in implementing a relevance feedback system when a given query does not succeed in retrieving any relevant items? How could one modify feedback equation (7) to handle such a situation?