INTRODUCTION TO MODERN INFORMATION RETRIEVAL



COMPUTER SCIENCE SERIES

EX. 1027
APPLE INC. / Page 1 of 18

Introduction to Modern Information Retrieval

Gerard Salton
Professor of Computer Science
Cornell University

Michael J. McGill
Associate Professor of Information Studies
Syracuse University

McGraw-Hill Book Company

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

This book was set in Times Roman by Progressive Typographers.
The editors were Charles E. Stewart and James E. Vastyan;
the production supervisor was John Mancia.
The drawings were done by VIP Graphics.
R. R. Donnelley & Sons Company was printer and binder.

INTRODUCTION TO MODERN INFORMATION RETRIEVAL

Copyright © 1983 by McGraw-Hill, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

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

ISBN 0-07-054484-0

Library of Congress Cataloging in Publication Data

Salton, Gerard.
Introduction to modern information retrieval.

(McGraw-Hill computer science series)
Includes indexes.
1. Information storage and retrieval systems.
I. McGill, Michael J. II. Title. III. Series.
Z699.S313 025'.04 81-20843
ISBN 0-07-054484-0 AACR2

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-

EX. 1027
APPLE INC. / Page 4 of 18

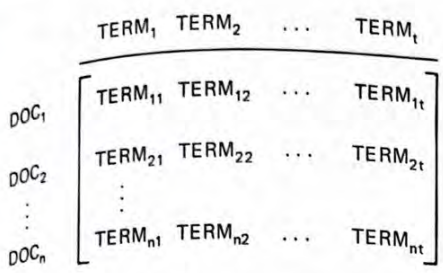


Figure 4-1 Term assignment array (n documents, t terms).

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

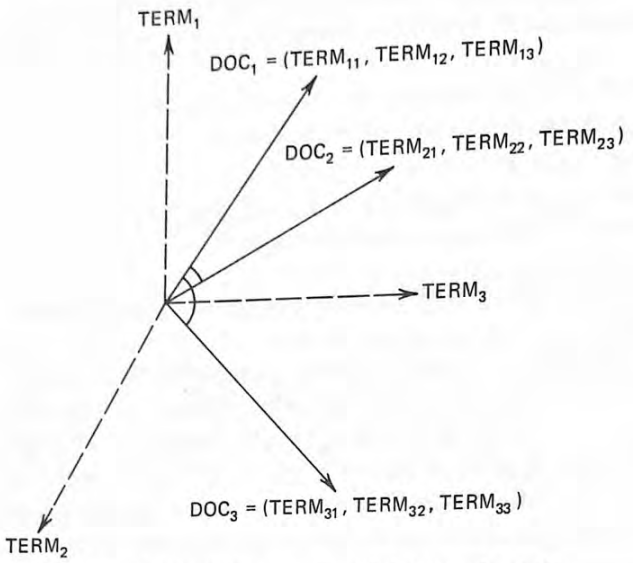


Figure 4-2 Vector representation of document space.

(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.

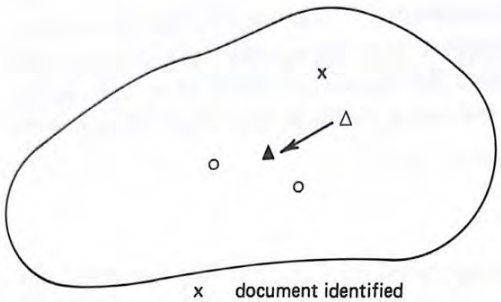
The query reformulation process is then based on the following complementary operations:

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-



x document identified as nonrelevant

o document identified as relevant

Δ original query

▲ reformulated query

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.

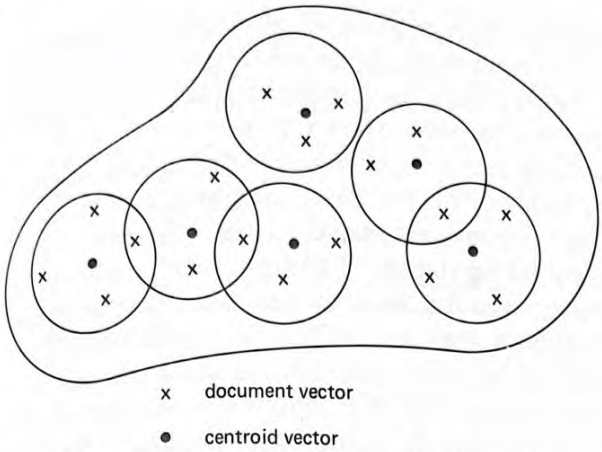


Figure 4-4 Sample clustered document collection.

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

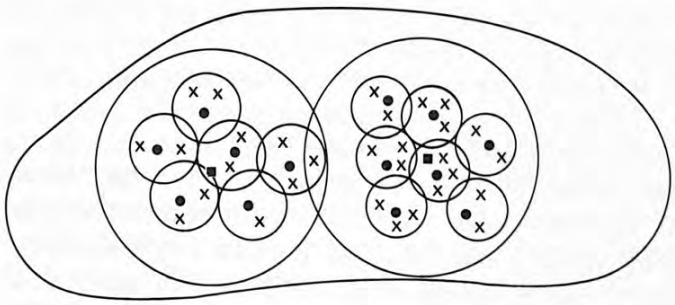


Figure 4-5 Introduction of superclusters.

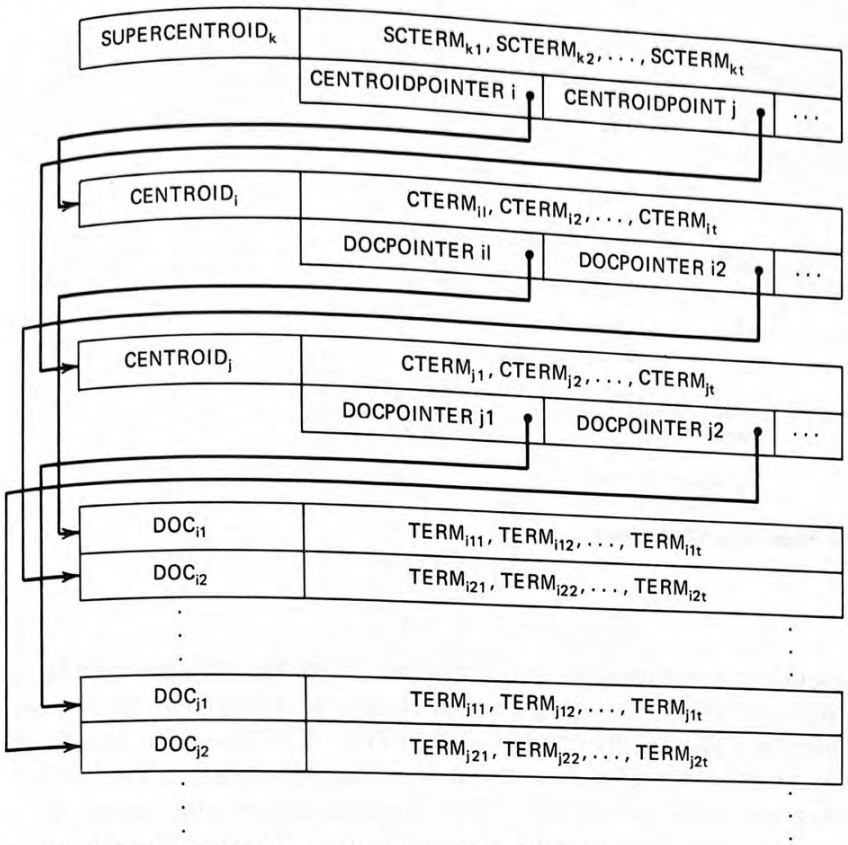
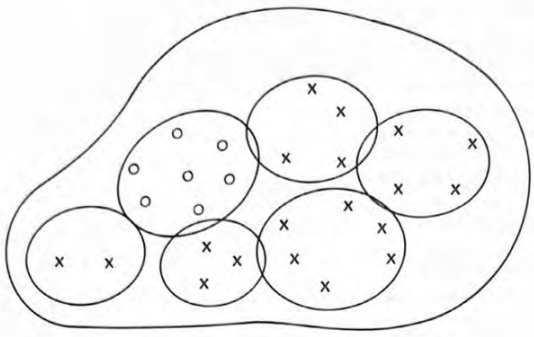


Figure 4-6 Hierarchical vector arrangement for clustered file.

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

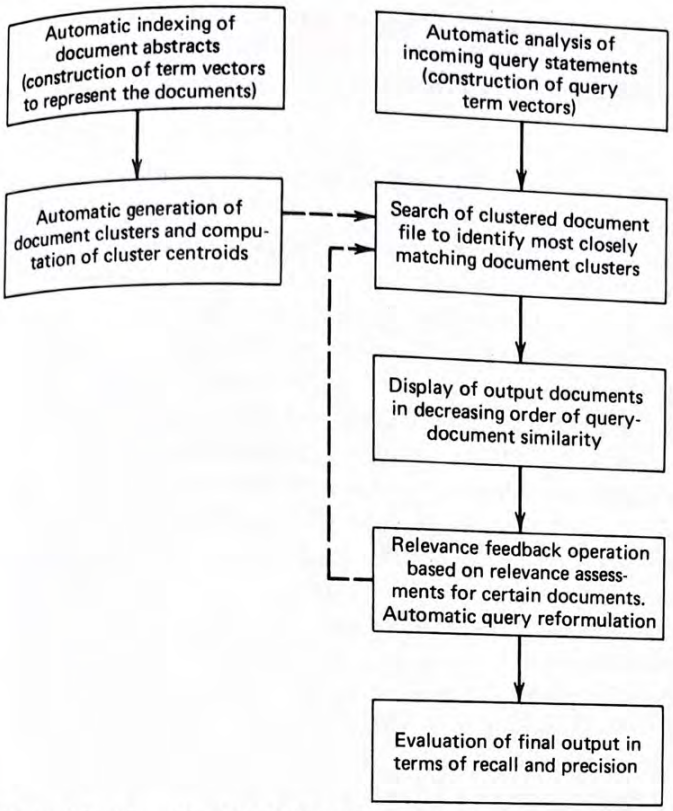


Figure 4-8 Simplified SMART system flowchart.

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-

EX. 1027
APPLE INC. / Page 14 of 18

*2 VECTOR SIMILARITY FUNCTIONS

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

tors. In a retrieval environment, the query usually contains low-weighted terms; hence the query-document correlations using the overlap measure SIM_4 will be larger in magnitude than those of the cosine SIM_3.

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

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

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

3 TERM WEIGHTING SYSTEMS

A Principal Weighting Strategies

In principle, the retrieval environment is simplest when the information items are characterized by unweighted properties and the indexing operation is binary. In this case, the degree to which a given property (term) may be useful to represent the content of an item is not a consideration. Any property that appears relevant is assigned to the information item and rejected when it appears extraneous. While the indexing is simplified, the task of evaluating the output of a search operation may be complicated because distinctions among the retrieved items, or for that matter among the items that are not retrieved, are more difficult to make for binary than for weighted vectors. When weighted

EX. 1027
APPLE INC. / Page 18 of 18