Concepts and Effectiveness of the Cover-Coefficient-Based Clustering Methodology for Text Databases

FAZLI CAN
Miami University
and
ESEN A. OZKARAHAN
The Pennsylvania State University

A new algorithm for document clustering is introduced. The base concept of the algorithm, the cover coefficient (CC) concept, provides a means of estimating the number of clusters within a document database and relates indexing and clustering analytically. The CC concept is used also to identify the cluster seeds and to form clusters with these seeds. It is shown that the complexity of the clustering process is very low. The retrieval experiments show that the information-retrieval effectiveness of the algorithm is compatible with a very demanding complete linkage clustering method that is known to have good retrieval performance. The experiments also show that the algorithm is 15.1 to 63.5 (with an average of 47.5) percent better than four other clustering algorithms in cluster-based information retrieval. The experiments have validated the indexing–clustering relationships and the complexity of the algorithm and have shown improvements in retrieval effectiveness. In the experiments, two document databases are used: TODS214 and INSPEC. The latter is a common database with 12,684 documents.

Categories and Subject Descriptors: H.2.2 [**Database Management**]: Physical Design—*access methods*; H.3.2 [**Information Storage and Retrieval**]: Information Storage—*file organization*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*clustering*; *retrieval models*; *search process*

General Terms: Algorithms, Design, Performance, Theory, Verification

Additional Key Words and Phrases: Clustering–indexing relationships, cluster validity, cover coefficient, decoupling coefficient, document retrieval, retrieval effectiveness

1. INTRODUCTION

The term *information retrieval* (*IR*) in this paper is equivalent to *document retrieval*. An *information retrieval system* (*IRS*) is a system that tries to locate and retrieve documents that are relevant to user queries. Textual data (e.g., journal and newspaper articles, reports, books, all of which are referred to as

Authors' addresses: F. Can, Department of Systems Analysis, Miami University, Oxford, OH 45056; E. A. Ozkarahan, School of Business, The Pennsylvania State University, Erie, PA 16563.

documents or *text*), are stored in a document database, and their management and retrieval characteristics differ from that of a database management system (DBMS). An IRS determines relevance with a similarity measure, whereas a DBMS returns facts that satisfy a Boolean qualification evaluation, that is, true or false.

Because document databases are often very large, IRSs normally perform retrievals on document representatives. Various models exist for document representation, one of which is the vector space model [23]. In the vector space model, a document collection is represented by a document-by-term matrix, which is referred to as the *D matrix*. In this matrix each row represents a single document as a collection of terms. Each row element is called an *index term*, or just *term* for brevity. The D matrix can be generated manually or by automatic indexing [19, 25, 27]. The individual entries of D, d_{ij} ($1 \leq i \leq m$, $1 \leq j \leq n$), indicate the importance of term j (t_j) in document i (d_i), where t_j is a member of the indexing vocabulary T, $T = \{t_1, t_2, \ldots, t_n\}$. If indexing is binary, d_{ij} will be either 0 or 1, indicating the presence or absence of t_j in d_i. Otherwise (i.e., in the case of weighted indexing), d_{ij} may indicate the number of occurrences of t_j in d_i. As we shall see later, term weights are normalized to achieve more effective retrieval.

In this representation a document is modeled by a vector in an n-dimensional space defined by n terms. These descriptors (i.e., vectors) can be clustered to improve search for retrieval. (Documents can also be represented in more traditional ways such as by using inverted files constructed for terms. Alternatively, we can map, by hashing, documents into what are called *signature files* [18, 23].)

Query processing retrieves documents that are relevant to the user request. Because an exact document match (in full text) rarely occurs in document retrieval, relevance of documents is determined by a similarity (matching) function. Documents that are most similar to the query according to the matching function are presented to the user. The actual relevance of the document is then decided by the user examining the retrieved documents.

In IR, search strategies vary from brute force, that is, *full search* (*FS*), to indexed and cluster-based retrievals. FS, which involves comparison of a query vector with the vectors of individual documents, is inefficient unless the database is rather small. However, it can be emulated by inverted index searching, or *index search* for short. An index provides postings for each term. A posting consists of the numbers of the documents in which the term appears.

Efficiency and effectiveness of IR can be increased by clustering documents based on their representatives. In the context of IR, a *cluster* indicates a homogeneous group of documents that are more strongly associated with each other than with those in different groups. The process of forming these groups is referred to as *clustering*. In the IR literature, there is a well-known hypothesis [27], known as the "clustering hypothesis," which states that "closely associated documents tend to be relevant to the same request." It is this hypothesis that justifies clustering of documents in a database [28].

In the search strategy, to be referred to as *clustered-based retrieval* (*CBR*), the queries are first compared with the clusters, or more accurately, with cluster

representatives called *centroids*. Detailed query-by-document comparison is performed only within selected clusters. It is also possible to arrange the clusters in a hierarchical structure. In this case, CBR takes place either as a top-down or a bottom-up search through the hierarchy [10, 12, 21, 26, 29, 32].

CBR facilitates browsing and easy access to complete document information [23, p. 345]. As with any such algorithm, the efficiency of CBR is important. To increase the efficiency of CBR, documents within a cluster can be stored in close proximity to each other within a disk medium to minimize I/O delays [22]. We can also use an index to search the centroids. In addition to being efficient, CBR should be effective in the sense of meeting user needs.

Clustering algorithms can be classified in various ways [1, 15]. One possible classification is according to the manner in which documents are distributed to clusters:

(1) *Partitioning type*: Clusters cannot have common documents.
(2) *Overlapping type*: Clusters can have common documents.
(3) *Hierarchical type*: Clustering structure is represented by a tree where the leaves of the tree represent documents. Upper levels of the tree represent clusters of clusters and/or documents. The root of the tree represents the cluster containing all of the document databases.

Another classification of clustering algorithms is according to the clustering methodology used:

(1) *Single-pass algorithms*: In these algorithms documents are handled only once.
(2) *Iterative algorithms*: Each iteration improves the value of an objective function measuring the quality of clustering. The algorithm terminates when an acceptable quality level is reached.
(3) *Graph theoretical algorithms*: In these algorithms, concepts like single link, group average link (in short, average link), and maximally complete graph (complete linkage method) are used. Here, *link* means a similarity value greater than or equal to a threshold between two documents. By changing the similarity threshold from higher to lower values, one may construct a hierarchical organization known as a dendrogram [27].

In this paper we introduce a new clustering methodology based on the cover coefficient (CC) concept. We present the CC concept, along with the various concepts and methodologies that are used with it. The CC methodology has a formal base and is useful in IR applications. In Section 2 the CC concept and methodologies, including the CC-based Clustering Methodology (C^3M), its complexity analysis, and the indexing–clustering relationships, are introduced. Section 3 covers our experimental design and evaluation. The experiments were designed and carried out to validate the CC-concept-related indexing–clustering relationships and to evaluate performance of C^3M in clustering and CBR. The experiments have been carried out using two document databases: the INSPEC database of 12,684 documents and 77 queries, and the TODS214 database of 214 documents and 58 queries. The retrieval evaluations are based on seven different similarity (matching) functions. The conclusion is given in Section 4.

2. CONCEPTS OF C^3M

The C^3M algorithm presented in this section is of the partitioning type. A generally accepted strategy to generate a partition is to choose a set of documents as the seeds and to assign the ordinary (nonseed) documents to the clusters initiated by seed documents to form clusters [1]. This is the strategy used by C^3M. *Cover coefficient, CC,* is the base concept of C^3M clustering. The CC concept serves to

(1) identify relationships among documents of a database by use of a matrix, the CC matrix, which will be defined shortly;
(2) determine the number of clusters that will result in a document database;
(3) select cluster seeds using a new concept, cluster seed power;
(4) form clusters with respect to C^3M, using concepts (1)–(3);
(5) correlate the relationships between clustering and indexing.

Each of these items will be dealt with in this section.

2.1 The CC Concept

In previous work we have introduced the CC concept for clustering document databases [2–5]. In this paper a probabilistic interpretation of this concept is introduced.

Before we proceed any further, however, we would like to point out a few distinct features of the CC concept to the reader who is not familiar with it. C^3M falls in the category of nonhierarchical clustering. It is therefore seed based. Unlike hierarchical clustering, where inclusion relationships and, therefore, similarities would suffice, in C^3M the seed must attract relevant documents onto itself. To achieve this properly, one must determine document relationships of coverage and similarities in a multidimensional space. Using the CC concept, these relationships are reflected in the C matrix, whose elements convey document/term couplings of a symmetrical as well as an asymmetrical nature.

Definition. A D matrix that represents the document database $\{d_1, d_2, \ldots, d_m\}$ described by the index terms $T = \{t_1, t_2, \ldots, t_n\}$ is given. The CC matrix, C, is a document-by-document matrix whose entries c_{ij} $(1 \leq i, j \leq m)$ indicate the probability of selecting any term of d_i from d_j.

In other words, the C matrix indicates the relationship between documents based on a two-stage probability experiment. The experiment randomly selects terms from documents in two stages. The first stage randomly chooses a term t_k of document d_i; then the second stage chooses the selected term t_k from document d_j. The motivation for this experimental method and for the CC concept are introduced here using binary indexing. Subsequently, the CC concept will be extended to incorporate weighted indexing.

To apply the CC concept, the entries of the D matrix, d_{ij} $(1 \leq i \leq m, 1 \leq j \leq n)$, must satisfy the following conditions:

(1) Each document must have at least one term.
(2) Each term must appear at least in one document.

From the definition of CC, it may initially seem that individual entries of the C matrix, c_{ij}, would be equal to the ratio (number of terms common to both d_i and d_j)/(sum of the document frequencies of terms in d_i). However, this is not true. For the calculation of c_{ij}, one must first select an arbitrary term of d_i, say, t_k, and use this term to try to select document d_j from this term, that is, to check if d_j contains t_k. In other words, we have a two-stage experiment [13, p. 94]. Each row of the C matrix summarizes the results of this two-stage experiment.

This can be better understood by analogy. Suppose we have many urns, and each urn contains balls of different colors. Then what is the probability of selecting a ball of a particular color? To find this probability experimentally, notice that first we have to choose an urn at random, and then we have to choose a ball at random. In terms of the D matrix, what we have is the following: From the terms (urns) of d_i, choose one at random. Each term appears in many documents, or each urn contains many balls. From the selected term try to draw d_j; or from the selected urn try to draw a ball of a particular color. What is the probability of getting d_j, or what is the probability of selecting a ball of a particular color? This is precisely the probability of selecting any term of d_i from d_j, since we are trying to draw the selected term of d_i from d_j at the second stage.

Let s_{ik} indicate the event of selecting t_k from d_i at the first stage, and let s'_{jk} indicate the event of selecting d_j from t_k at the second stage. In this experiment, the probability of the simple event "s_{ik} and s'_{jk}," that is, $P(s_{ik}, s'_{jk})$, can be represented as $P(s_{ik}) \times P(s'_{jk})$ [13]. To simplify the notation, we use s_{ik} and s'_{jk}, respectively, for $P(s_{ik})$ and $P(s'_{jk})$, where

$$s_{ik} = d_{ik} \times \left[\sum_{h=1}^{n} d_{ih} \right]^{-1}, \qquad s'_{jk} = d_{jk} \times \left[\sum_{h=1}^{m} d_{hk} \right]^{-1}$$

$$\text{for} \quad 1 \le i, \quad j \le m, \quad 1 \le k \le n.$$

By considering document d_i, we can represent the D matrix with respect to the two-stage probability model, as shown in Figure 1. Consider a path that starts from d_i and ends up at one of the documents, say, d_j, of the database. There are n possible ways $(s_{i1}, s_{i2}, \ldots, s_{in})$ to reach d_j $(1 \le j \le m)$ from d_i. Choose one of them, for example, s_{ik}, so that the intermediate stop is t_k. After this intermediate stop, in order to reach d_j we must follow s'_{jk}. Accordingly, the probability of reaching d_j from d_i via t_k becomes $s_{ik} \times s'_{jk}$. By using Figure 1, we can find c_{ij} (i.e., the probability of selecting a term of d_i from d_j) by summing the probabilities of individual paths from d_i to d_j.

Let us present an example by using the D matrix of Figure 2. To illustrate this concept let us follow the calculation of c_{12}. According to the D matrix, d_1 contains three terms $\{t_1, t_2, t_5\}$. According to the two-stage probability model, to calculate c_{12} we must first randomly select each one of the terms of d_1 and then try to select d_2 from the outcome (term) of the first stage. At the first stage, if we select t_1 or t_5, then d_2 has a chance of $\frac{1}{2}$. However, if t_2 is selected at the first stage, then the probability of selecting d_2 at the second stage is $\frac{1}{4}$. This is because t_1 and t_5 appear in d_1 and d_2. On the other hand, t_2 appears in d_1 and d_2 and in two other documents. At the first stage, the probability of selecting each element of $\{t_1, t_2, t_5\}$ from d_1 is $\frac{1}{3}$, and for the rest the probability is 0 (i.e., no term in

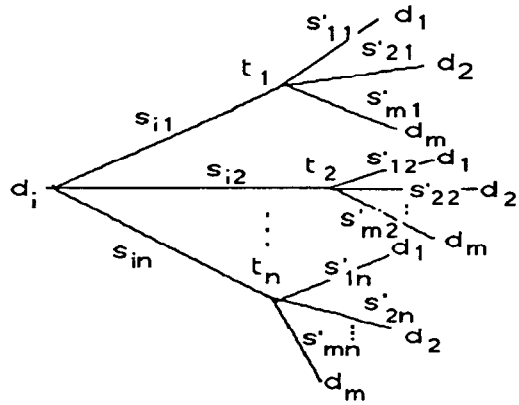

Fig. 1. Hierarchical representation of the two-stage probability model for d_i of the D matrix.

$$D = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{matrix}$$

Fig. 2. Example D matrix.

$\{t_3, t_4, t_6\}$ appears in d_1). Accordingly c_{12} is calculated as follows:

$$c_{12} = \sum_{k=1}^{6} s_{1k} \times s'_{2k}$$

$$= \tfrac{1}{3} \times \tfrac{1}{2} + \tfrac{1}{3} \times \tfrac{1}{4} + 0 \times 0 + 0 \times \tfrac{1}{2} + \tfrac{1}{3} \times \tfrac{1}{2} + 0 \times 0 = 0.417.$$

The C matrix can be formed from the matrices named S and S′, that is, C = S × S′$^{\mathrm{T}}$ (S′$^{\mathrm{T}}$ indicates the transpose of matrix S′), where matrix elements s_{ik} and s'_{jk} are as defined previously; s_{ik} and s'_{jk} indicate the probability of selecting t_k from d_i and the probability of selecting d_j from t_k, respectively. Using the definitions of the S and S′ matrices, the entries of the C matrix can be defined as follows:

$$c_{ij} = \sum_{k=1}^{n} s_{ik} \times s'^{\mathrm{T}}_{kj} = \sum_{k=1}^{n} (\text{probability of selecting } t_k \text{ from } d_i) \times (\text{probability of selecting } d_j \text{ from } t_k), \tag{2.1}$$

where $s'^{\mathrm{T}}_{kj} = s'_{jk}$. This can be rewritten as

$$c_{ij} = \alpha_i \times \sum_{k=1}^{n} d_{ik} \times \beta_k \times d_{jk}, \qquad 1 \le i, \ j \le m, \tag{2.2}$$

where α_i and β_k are the reciprocals of the ith row sum and the kth column sum, respectively, as shown below:

$$\alpha_i = \left[\sum_{j=1}^{n} d_{ij} \right]^{-1}, \qquad 1 \leq i \leq m, \tag{2.3}$$

$$\beta_k = \left[\sum_{j=1}^{m} d_{jk} \right]^{-1}, \qquad 1 \leq k \leq n. \tag{2.4}$$

2.2 Properties of the C Matrix

The following properties hold for the C matrix:

(1) For $i \neq j$, $0 \leq c_{ij} \leq c_{ii}$ and $c_{ii} > 0$.

(2) $c_{i1} + c_{i2} + \cdots + c_{im} = 1$ (i.e., sum of row i is equal to 1 for $1 \leq i \leq m$).

(3) If none of the terms of d_i is used by the other documents, then $c_{ii} = 1$; otherwise, $c_{ii} < 1$.

(4) If $c_{ij} = 0$, then $c_{ji} = 0$, and similarly, if $c_{ij} > 0$, then $c_{ji} > 0$; but in general, $c_{ij} \neq c_{ji}$.

(5) $c_{ii} = c_{jj} = c_{ij} = c_{ji}$ iff d_i and d_j are identical.

The proofs of properties (1)–(4) are given in [2] and [3]. The proof of property (5) is given in [7].

In property (1), $c_{ij} \geq 0$ means that it may ($c_{ij} > 0$) or may not ($c_{ij} = 0$) be possible to select the terms of d_i from d_j; $c_{ij} \leq c_{ii}$ is obvious; when one tries to select the terms of d_i from the database, d_i itself will have a higher chance of being selected than any other document. However, if d_j contains all the terms of d_i then $c_{ij} = c_{ii}$. We can, of course, always select a term of d_i from itself, $c_{ii} > 0$.

Property (2) means that the entries of each row are the outcomes of a two-stage experiment, and the sum of all the probabilities will be the universal space, that is, 1.

Property (3) means that if some of the terms of d_i appear in another document, say, d_j, then the corresponding c_{ij} entries will be nonzero for $j \neq i$.

Property (4) means that, if it is not possible to draw a term of d_i from d_j, then it is also not possible to draw a term of d_j from d_i (i.e., they do not have any common terms). If d_i and d_j have common terms but are not identical, then c_{ij} and c_{ji} will be greater than zero but not necessarily equal to each other. To put it another way, it can be observed that, unlike most other similarity measures (such as cosine), the measure of association between two documents indicated by the CC concept is asymmetric as expected. This comes from the probabilistic nature of the CC concept. (It is known that association measures based on conditional probabilities are often asymmetric [1, p. 71]. The CC concept is defined by a two-stage experiment, which is an example of conditional probability random experiments [13, p. 90].)

Property (5) means that, if two documents are identical, then the extent to which they cover each other and themselves are identical.

Another property of the C matrix is the following: Let μ be any positive real number. If a D matrix is mapped into a C matrix, then $\mu \times$ D defines the same

transformation as D. This property comes with the definition of the c_{ij} entries, eq. (2.2).

These properties indicate that, if a document has terms in common with many of the other documents in the database, then we will observe many nonzero values in the off-diagonal entries of the corresponding row. Since the row sum is equal to 1, the diagonal entry of the C matrix for the document will have a value less than 1. In the reverse case, that is, if a document has few terms in common with the rest of the database, then we will observe many zero values in the off-diagonal entries and a relatively higher value (close to 1) in the diagonal entry of the C matrix corresponding to the document. If a document does not share any of its terms with the rest of the documents, then the corresponding diagonal entry of the C matrix will have its maximum value (i.e., 1), and all the off-diagonal entries of that row will be equal to 0. From these properties of the C matrix and from the CC relationships between two document vectors, c_{ij} can be seen to have the following meaning:

$$c_{ij} = \begin{cases} \text{extent to which } d_i \text{ is covered by } d_j \text{ for } i \neq j \\ \text{(coupling of } d_i \text{ with } d_j\text{)}, \\ \text{extent to which } d_i \text{ is covered by itself for } i = j \\ \text{(decoupling of } d_i \text{ from the rest of the documents).} \end{cases}$$

To obtain a better understanding of the meaning of the C matrix, consider two document vectors d_i and d_j. For these document vectors, four possible relationships can be defined in terms of the C matrix entries (i.e., in terms of c_{ii}, c_{ij}, c_{jj}, and c_{ji}):

(1) *Identical documents*: Coupling and decoupling of any two such documents are equivalent. Furthermore, the extent to which these two documents are covered by other documents is also identical (i.e., $c_{ik} = c_{jk}$, where $1 \leq k \leq m$). Similarly, the extent to which these two documents cover other documents is the same (i.e., $c_{ki} = c_{kj}$, where $1 \leq k \leq m$).

(2) *Overlapping documents*: Each document will cover itself more than any other ($c_{ii} > c_{ij}$, $c_{jj} > c_{ji}$). However, this does not provide enough information to compare c_{ii} with c_{jj} and c_{ij} with c_{ji}. This is because these values are also affected by the couplings with the other documents of the database.

(3) *A document is a subset of another document*: Let d_i be a subset of d_j. Since d_i is a subset of d_j, the extent to which d_i is covered by itself (c_{ii}) will be identical to the extent to which d_i is covered by d_j (c_{ij}). Furthermore, since d_j contains all of the terms of d_i as well as some additional terms, then the extent to which d_j covers itself will be higher than the extent to which d_i covers itself (i.e., $c_{jj} > c_{ii}$). By similar reasoning, the extent to which d_j covers d_i is higher than the extent to which d_i covers d_j ($c_{ij} > c_{ji}$).

(4) *Disjoint documents*: Since d_i and d_j do not have any common terms, then they will not cover each other ($c_{ij} = c_{ji} = 0$). Obviously, the documents will cover themselves. However, because these documents may also be coupled with the others, c_{ii} and c_{jj} may be less than 1.

As can be seen from the foregoing discussions, in a D matrix, if d_i ($1 \leq i \leq m$) is relatively more distinct (i.e., if d_i contains fewer terms that are common with

$$C = \begin{bmatrix} 0.417 & 0.417 & 0.000 & 0.083 & 0.083 \\ 0.313 & 0.438 & 0.000 & 0.063 & 0.188 \\ 0.000 & 0.000 & 0.333 & 0.333 & 0.333 \\ 0.083 & 0.083 & 0.111 & 0.361 & 0.361 \\ 0.063 & 0.188 & 0.083 & 0.271 & 0.396 \end{bmatrix}$$

Fig. 3. C matrix corresponding to the example D matrix.

other documents), then c_{ii} will take higher values. Because of this, c_{ii} is called the *decoupling coefficient*, δ_i, of d_i. (Notice that δ_i is a "measure" of how much the document is not related to the other documents, and this is why the word *coefficient* is used.)

The sum of the off-diagonal entries of the ith row indicates the extent of coupling of d_i with the other documents of the database and is referred to as the coupling coefficient, ψ_i, of d_i. From the properties of the C matrix,

$$\delta_i = c_{ii}\text{: decoupling coefficient of } d_i;$$
$$\psi_i = 1 - \delta_i\text{: coupling coefficient of } d_i.$$

The ranges for δ_i and ψ_i are $0 < \delta_i \leq 1$ and $0 \leq \psi_i < 1$, for $1 \leq i \leq m$.

By using the individual δ_i and ψ_i values, the overall or average decoupling, δ, and coupling, ψ, coefficients of the documents can be defined as

$$\delta = \sum_{i=1}^{m} \delta_i/m \quad \text{and} \quad 0 < \delta \leq 1; \tag{2.5}$$

$$\psi = \sum_{i=1}^{m} \psi_i/m \quad \text{and} \quad 0 \leq \psi < 1. \tag{2.6}$$

The C matrix corresponding to the example D matrix given earlier has the values given in Figure 3, disregarding inexactness due to rounding. The entire C matrix is given for the sake of illustration. However, the implementation of C^3M- and CC-based concepts do not require complete construction of the C matrix. From this C matrix, the overall decoupling and coupling coefficients for the database are

$$\delta = \sum_{i=1}^{5} \delta_i/5 = 1.945/5 = 0.389, \qquad \psi = 1 - \delta = 0.611.$$

Looking at the D matrix, we can see that d_1 is a subset of d_2. Accordingly, $c_{11} = c_{12}$ (the extent to which d_1 covers itself is the same as the extent to which it is covered by d_2). The other relationships are $c_{22} > c_{21}$, $c_{22} > c_{11}$, and $c_{12} > c_{21}$, as can be verified with the C matrix.

2.3 The C′ Matrix

By following a methodology similar to the construction of the C matrix, we can construct a term-by-term C′ matrix of size n by n for index terms. C′ has the same properties as C. As with the C matrix, the C′ matrix summarizes the results of a two-stage experiment. C′ is defined as the product of S'^T and S matrices.

$$C' = \begin{bmatrix} 0.292 & 0.292 & 0.000 & 0.125 & 0.292 & 0.000 \\ 0.146 & 0.292 & 0.146 & 0.125 & 0.146 & 0.146 \\ 0.000 & 0.292 & 0.292 & 0.125 & 0.000 & 0.292 \\ 0.125 & 0.250 & 0.125 & 0.250 & 0.125 & 0.125 \\ 0.292 & 0.292 & 0.000 & 0.125 & 0.292 & 0.000 \\ 0.000 & 0.194 & 0.194 & 0.083 & 0.000 & 0.528 \end{bmatrix}$$

Fig. 4. C′ matrix corresponding to the example D matrix.

Its elements are defined as follows:

$$c'_{ij} = \sum_{k=1}^{m} s'^{\mathrm{T}}_{ik} \times s_{kj} = \sum_{k=1}^{m} (\text{probability of selecting } d_k \text{ from } t_i) \times (\text{probability of selecting } t_j \text{ from } d_k).$$

Notice that, in the case of C′, the stages of the two-stage experiment are interchanged with respect to the order for the C matrix. By using the definitions of the S and S′ matrices, the entries of the C′ matrix can be defined as follows:

$$c'_{ij} = \beta_i \times \sum_{k=1}^{m} d_{ki} \times \alpha_k \times d_{kj}. \tag{2.7}$$

The concepts of decoupling and coupling coefficients of t_j, δ'_j and ψ'_j, are the counterparts of the same concepts defined for documents. Hence, $\delta'_j = c'_{jj}$, and $\psi'_j = 1 - \delta'_j$. The concepts of overall or average coupling and decoupling are also valid for terms and are represented by δ' and ψ', respectively. The relationships between δ' and δ will be seen at the end of the next section.

The C′ matrix resulting from the example D matrix is (ignoring rounding errors) given in Figure 4.

2.4 Number-of-Clusters Hypothesis

The prediction of the number of clusters for a database (in general, for a multivariate data set) has been an unresolved problem in cluster analysis [1, 15]. In the hierarchical clustering methods, the number of clusters can vary from 1 (the entire database) to the number of documents (i.e., objects) in the database [1]. The outcome of these algorithms, *dendrogram*, can be cut at a certain level to obtain a predetermined number of clusters. The procedure for cutting a dendrogram is called the *stopping rule* [15]. Unfortunately, stopping rules are hard to apply [10], even for data collections of small sizes [15]. The cluster analysis literature does not provide much help in the IR context. In cluster analysis, the Euclidean distance measure is generally used as the similarity coefficient to predict and/or to validate the number of clusters [15]. However, Euclidean distance is not a good choice for document clustering. With Euclidean distance, two documents can be regarded as very similar to each other even though they do not share any common terms [32].

Let us have a close look at the number-of-clusters problem in an IR environment. In a document database, one obtains an extreme case for number of clusters

when all documents are the same. In this case there should be only one cluster. The other extreme for number of clusters can be observed when all documents are distinct (i.e., each term appears only in one document). For this extreme, the number of clusters should be equal to the number of documents. Typically, however, the number of clusters will be greater than 1 and less than the size of the database (m), since all documents will not be identical and will not be completely distinct. The following hypothesis accounts for these facts and/or observations:

HYPOTHESIS. *The number of clusters within a database should be high if individual documents are dissimilar, and low otherwise.*

The above hypothesis is obvious; however, the similarity concept is not much help in obtaining the number of clusters. This is because, as stated above, it is difficult to predict a similarity threshold (stopping rule) that will yield the desired number of clusters. In an effort to provide a solution to this problem, we have been able to use successfully the CC concept to predict the number of clusters, n_c, for a database. This prediction must agree with the number-of-clusters hypothesis. The diagonal entries of the C matrix yield n_c as follows:

$$n_c = \sum_{i=1}^{m} \delta_i = \delta \times m. \tag{2.8}$$

Equation (2.8) is consistent with the number-of-clusters hypothesis. This is because a database with similar documents will have a low δ (decoupling) and few clusters. On the other hand, a database with dissimilar documents will have a high δ and many clusters. The previous statement also holds true for terms, with δ being replaced with δ'. Hence, the number of term clusters implied by the C′ matrix, n_c', would be

$$n_c' = \sum_{j=1}^{n} \delta_j' = \delta' \times n. \tag{2.9}$$

It is known that the classification of documents implies the classification of terms and vice versa, since as we classify documents, the clustering process will group documents by the terms most frequently used within their clusters. The reverse is also true. The idea of classifying terms and then using term clusters to form document clusters [8] or vice versa [16] has been used in the literature by various researchers. In our context this means that n_c and n_c' should be identical, and indeed this identity has proved to hold [2, 3].

Let us show the agreement between eqs. (2.8) and (2.9) and the number-of-clusters hypothesis with the use of the following theorems and corollaries:

THEOREM 1. $n_c = 1$ *iff all documents of D are identical.*

PROOF. The proof is given in [7]. □

THEOREM 2. *For a binary D matrix, the minimum value of* c_{ii} (*i.e.*, δ_i) *for* $1 \le i \le m$ *is* $1/m$.

PROOF. If $c_{ii} < 1/m$, then this means that $c_{ij} < 1/m$ for $i \neq j$ (since $c_{ii} \ge c_{ij}$). Then $c_{i1} + c_{i2} + \cdots + c_{im} < 1$, which contradicts property (2) of the C matrix; that is, $c_{i1} + c_{i2} + \cdots + c_{im} = 1$. Hence, the minimum value of $c_{ii} = 1/m$. □

Intuitively, if there is at least one document distinct from the others in a document database, then the number of clusters must be greater than 1. This is stated in the following corollary:

COROLLARY 1. *In a binary D matrix,* $n_c > 1$ *if we have at least two distinct documents in D.*

PROOF. Consider two documents d_i and d_j. If they are identical, then $c_{ii} = c_{jj}$. If d_i and d_j are distinct, then $c_{ii} > c_{ij}$ and $c_{jj} > c_{ji}$, due to properties (1) and (5) of the C matrix. Consider d_i. Since the minimum value of c_{ii} is $1/m$, then, in order to have the inequality hold, c_{ii} must be greater than $1/m$. (Notice that we cannot have both $c_{ii} = 1/m$ and $c_{ii} > c_{ij}$ at the same time, since in such a case the row sum becomes less than 1, which contradicts property (2) of the C matrix; i.e., if $c_{ii} = 1/m$, then none of the off-diagonal entries can assume a value greater than $1/m$.) Furthermore, if we have at least one c_{ii} value greater than $1/m$, then it implies that n_c is greater than 1, since the lowest value that can be assumed by c_{ii} is $1/m$. □

COROLLARY 2. *The value range of* n_c *is* $1 \leq n_c \leq min(m, n)$.

PROOF. The minimum value of n_c is stated in Theorem 2. We know that $n_c = n_c'$ and that n_c, n_c' are the summation of the diagonal entries of C and C′ matrices, respectively. C and C′ are square matrices of sizes $m \times m$ and $n \times n$, respectively. Hence, the maximum value of n_c is m and of n_c' is n. On the other hand, $n_c = n_c'$. This implies that $\max(n_c) = \min(m, n)$. □

After determining n_c, the average number of documents, d_c, within a cluster follows from $d_c = m/n_c = 1/\delta$. The concept of decoupling coefficient implies that an increase in δ or in individual δ_i $(1 \leq i \leq m)$ will increase the number of clusters $(n_c = \delta \times m)$ and hence decrease the average size of clusters. The opposite is also true. The range of d_c is $\max(1, m/n) \leq d_c \leq m$.

Let us compute the number of clusters that will result from the example D matrix:

$$n_c = \sum_{i=1}^{5} \delta_i = (0.417 + 0.438 + 0.333 + 0.361 + 0.396) = 1.945,$$

$$n_c = \delta \times m = 0.389 \times 5 = 1.945 \cong 2.$$

If we use the C′ matrix, n_c' would come out as 1.946, which agrees with n_c (the difference is due to rounding).

The a priori knowledge of the number of clusters enables us to estimate the space requirements of clustering (i.e., there will be n_c centroids). This would lead to the estimation of both the vector and full text storage space in the document database. If we impose arbitrary restrictions on cluster size, the result may be detrimental to retrieval effectiveness. We may be forced to create a few fat clusters by forcing marginally related documents into clusters. We may alternatively be forced to create too many artificially split clusters whose intercluster separation is ill-defined. The impact of either too fat or too thin clusters on retrieval effectiveness will be counterproductive. The CC concept provides the theoretical knowledge of the number of clusters. n_c is a function of the D matrix.

Table I. Properties of the C Matrix

Property ($1 \le i, j \le m, i \ne j$)	D matrix	
	Binary	Weighted
$0 < c_{ii} \le 1, 0 \le c_{ij} < 1$	√	√
$c_{ij} \le c_{ii}$	√	(1)
$c_{ij} > c_{ii} \rightarrow c_{jj} > c_{ji}$	(2)	√
$c_{i1} + c_{i2} + \cdots + c_{im} = 1$	√	√
d_i is distinct $\leftrightarrow c_{ii} = 1$	√	√
$c_{ij} = 0 \leftrightarrow c_{ji} = 0, c_{ij} > 0 \leftrightarrow c_{ji} > 0$	√	√
$\min(c_{ii}) = 1/m$	√	(3)
$1 \le n_c \le \min(m, n)$	√	√
$c_{ii} = c_{jj} = c_{ij} = c_{ji} \leftrightarrow d_i$ and d_j are identical	√	√
$n_c = 1 \leftrightarrow$ all documents are identical	√	√
d_i, d_j are identical $\rightarrow c_{ik} = c_{jk}, c_{ki} = c_{kj}$ $(1 \le k \le m)$	√	√
D and $u \times$ D $(u > 0)$ implies the same C matrix	√	√

Notes: (1) $c_{ij} > c_{ii}$ is possible; (2) $c_{ij} > c_{ii}$ is impossible; (3) $\min(c_{ii}) < 1/m$ is possible.

If, however, we are forced to vary cluster size due to space restrictions of our computer, the indexing–clustering relationships expressed with the CC concept provide us the know-how of achieving that in a manner that does not sacrifice retrieval quality.

Before proceeding, the peculiarities of the C matrix corresponding to a weighted D matrix will be pointed out. A weighted D matrix may lead to $c_{ij} > c_{ii}$ [7]. Within the context of the CC concept, the inequality $c_{ij} > c_{ii}$ intuitively implies the inequality $c_{jj} > c_{ji}$. In other words, if d_j covers d_i more than the extent to which d_i covers itself, then the extent to which d_j covers itself must be greater than the extent to which d_j is covered by d_i.

THEOREM 3. *In the C matrix corresponding to a weighted D matrix, if* $c_{ij} > c_{ii}$ *then* $c_{jj} > c_{ji}$.

PROOF. The proof is given in [7]. □

Hence, for a weighted D matrix $\delta_i < 1/m$ is possible. However, $1 \le n_c \le \min(m, n)$ is still valid, since the minimum value of $n_c = 1$ even for a weighted D matrix.

THEOREM 4. *In the C matrix corresponding to a weighted D matrix, the minimum value of* n_c *is* 1.

PROOF. The proof is by contradiction and is given in [7]. □

For easy reference, the properties of the C matrix are summarized in Table I. (Because of the mirror image definitions of C and C′ matrices, all of the properties of the C matrix are valid for the C′ matrix.)

For added interest, we can state the relationship between δ and δ'. From $n_c = n_c'$, we have $\delta = (n/m) \times \delta'$ and $\delta' = (m/n) \times \delta$. In a real-life environment, one would expect to observe $\delta < \delta'$, since m would be greater than n.

2.5 Cluster Seed Power

The C^3M is a seed-oriented document-clustering methodology; that is, n_c documents are selected as cluster seeds, and nonseed documents are grouped around the seeds to form clusters. Seed documents must be well separated from each other and at the same time must be able to pull nonseed documents to themselves. Therefore, seed documents cannot be too general (containing many terms) or too specific (containing very few terms). To satisfy these constraints, we introduce the *cluster seed power*, P_i, of d_i $(1 \leq i \leq m)$, which is defined as

$$P_i = \delta_i \times \psi_i \times \sum_{j=1}^{n} d_{ij} \tag{2.10}$$

and

$$P_i = \delta_i \times \psi_i \times \sum_{j=1}^{n} (d_{ij} \times \delta_j' \times \psi_j'). \tag{2.11}$$

Equations (2.10) and (2.11) pertain to a binary and weighted D matrix, respectively. In these equations δ_i provides the separation of clusters (intercluster dispersion), ψ_i provides the connection among the documents within a cluster (intracluster cohesion), and the third term (summation) provides normalization. In a weighted D matrix, we need to pay attention to the weights of terms in a document. By eq. (2.11), a term with a high weight in d_i, but with a skewed frequency (i.e., a very frequent or a very rare term), will not contribute significantly to the seed power of d_i. The first n_c documents with the highest seed power are selected as the seed documents.

Note that some seeds may be almost identical, since they may be described by nearly identical sets of terms. To eliminate the identical (false) seeds, the following method is introduced: First we have to determine the documents with almost the same seed power value by using a threshold value. To do this we sort the documents according to their seed power values. In the sorted list, if two consecutive cluster seed powers are significantly different, then the corresponding documents are different. However, documents within close range of seed power values can have false (identical) seeds. Such documents will be grouped in the sorted list. In each group (i.e., the set of documents with "equal" seed power) we compare the candidate seeds with each other. From the properties of the C matrix, if two candidate seeds of the same group, d_i and d_j, are almost identical, then the entries c_{ii}, c_{jj}, c_{ij}, and c_{ji} will be almost identical, that is, the absolute value of $(c_{ii} - c_{jj}) < \epsilon$, $(c_{ii} - c_{ij}) < \epsilon$, $(c_{jj} - c_{ji}) < \epsilon$, $(c_{ij} - c_{ji}) < \epsilon$, and where ϵ is the threshold. (For each false seed, we consider another document from the sorted list of documents.) In our experiments, ϵ was the real constant 0.001. We never had identical seeds; that is, the first n_c documents of the sorted list did not contain any identical seeds.

In the experiments, it has been observed that documents with a medium number of terms tend to have higher seed power. This makes sense for a seed selection method, since, as explained in the beginning of this section, general or special documents would not be appropriate for a cluster seed. General documents do not provide intercluster dispersion, and special documents do not attract other documents.

Using the example D matrix, the seed powers of the documents are calculated according to eq. (2.10) and are listed in decreasing order as follows: $p_2 = 0.985$, $p_5 = 0.957$, $p_1 = 0.729$, $p_4 = 0.692$, and $p_3 = 0.222$. Since $n_c = 2$, then d_2 and d_5 become candidate seed documents. Our false seed elimination procedure determines that they are distinct. (Notice that the values $c_{22} = 0.438$ and $c_{55} = 0.396$ are significantly different.) Hence, d_2 and d_5 are selected as the cluster seeds. Notice that in this example no further check is needed; we can decide on the basis of only c_{22} and c_{55}, since they are significantly different.

2.6 The C³M Algorithm

C³M is a partitioning-type clustering algorithm that operates in a single pass. A brief description of the algorithm is as follows [2, 3]:

C³M:
[a] *Determine the cluster seeds of the database.*
[b] $i = 1$;
repeat; /* *construction of clusters* */
if d_i *is not a cluster seed*
then
begin;
Find the cluster seed (if any) that maximally covers d_i; *if there is more than one cluster seed that meets this condition, assign* d_i *to the cluster whose seed power value is the greatest among the candidates*;
end;
$i = i + 1$;
until $i > m$.
[c] *If there remain unclustered documents, group them into a ragbag cluster (some nonseed documents may not have any covering seed document).*

Notice that in the above algorithm we try to concentrate nonseed documents around seed documents. Some nonseed documents may not find any seed document to join. Such documents are inserted into a ragbag cluster in step (c) of the algorithm. The upper bound for the size of the ragbag cluster is $(m - n_c)$. The better selection of seeds leads to a lower size for the ragbag cluster. In our experiments we obtained a ragbag cluster only once with the D_{I7} matrix of the INSPEC database (see Section 3.2). This ragbag cluster contained just one document.

2.6.1 *Implementation of the Algorithm.* The construction of clusters occurs in step (b) of C³M, and this step determines the computational complexity of the algorithm [2, 3]. For the construction of clusters, we can use

I1: the D matrix; or
I2: an inverted index for the terms that appear in cluster seeds.

In I1, for a nonseed document d_i and c_{ij} (where d_j is a seed) values of each seed document are computed one after the other. In I2, the inverted index of a term includes the seed document numbers that contain the term and the weight of the term in each seed [20], and c_{ij} values are concurrently computed for those cluster seeds that have terms common with the nonseed document d_i. In both I1 and I2, each document is represented by a sorted term vector containing the term numbers and the associated weights (for a weighted D matrix).

The complexities of implementations with I1 or I2 are as follows:

I1: $O((m - n_c) \times x_d \times n_c)$; and
I2: $O((m - n_c) \times x_d \times t_{gs} + n_c \times x_d) \Rightarrow O((m - n_c) \times x_d \times t_{gs})$.

In the above expressions, x_d is the average number of distinct terms per document, and t_{gs} is the average number of seed documents per term. As will be explained later, t_{gs} accounts only for terms having a generality of two or more, since only such terms can appear both in seed and nonseed documents.

Consider the significance of each term in the complexities. There are $(m - n_c)$ documents to cluster, and in each document, on the average there are x_d terms to consider. In I1, for each document, there are n_c clusters to consider. On the other hand, in I2 there is approximately a list of length t_{gs} for each term of a nonseed document. Here we are ignoring the existence of terms with term generality one in nonseed documents. The second term of I2, $(n_c \times x_d)$, accounts for the construction of an inverted index for terms appearing in seed documents. This second term can be ignored in the complexity, since it is much less than the first term, $(m - n_c) \times x_d \times t_{gs}$.

In passing, it should be noted that in I2 the product $(m - n_c) \times x_d \times t_{gs}$ is an approximation and can be rewritten as

$$(m - n_c) \times \left(x_d \times \frac{s}{n_2}\right) \times \frac{t_s}{s} \quad \text{or} \quad (m - n_c) \times x_d \times \frac{t_s}{n_2}.$$

In this formula, s is the number of distinct terms contained in seed documents (these terms will be referred to as select terms), and t_s is the sum of seed document frequencies of the select terms. Since our seed selection approach returns well-separated seed documents, we can ignore the fact that some select terms would appear only in seed documents. Accordingly, the probability of occurrence of a nonseed document term in a seed document is roughly equal to s/n_2, where n_2 is the number of terms that appear in at least two documents. Notice that only such terms of nonseed documents can appear in seed documents. (This means that for each document to be clustered an s/n_2 portion of x_d terms can be select terms, and for each select term, we have to consider an index of length roughly equal to t_s/s. Notice that t_{gs} is equal to $(s/n_2) \times (t_s/s) = (t_s/n_2)$, and we can assign the meaning of "probable number of seed documents per term" to this product. The word *probable* is due to the factor s/n_2, which indicates a probability. However, for t_s/n_2 we prefer to attach the meaning "average (expected) number of seed documents per term."

The computational cost of I2 is a fraction of I1 and is t_{gs}/n_c, although I2 involves the construction of an inverted index for terms that appear in seed documents [7]. Our experiments showed that the typical cost of I2 is approximately 5 percent and 1 percent of I1, respectively, for the document databases TODS214 and INSPEC.

The foregoing discussion shows that the expected complexity of C^3M is $O(m \times x_d \times t_{gs})$, since $m \gg n_c$, and that the complexity of the other steps of the algorithm can be ignored with respect to the complexity of step (b). Obviously, this is better than most other clustering algorithms whose complexity ranges from $O(m^2)$ to $O(m^3)$ [22, 30, 32]. In Section 3.3, C^3M's complexity will be experimentally verified.

Consider the construction of clusters for the example D matrix. We have determined earlier that $\{d_1, d_3, d_4\}$ is the set, D_0, of documents to be clustered and $\{d_2, d_5\}$ is the set, D_s, of seed documents. To construct the clusters, we need only to calculate c_{ij}s, where $d_i \in D_0$ and $d_j \in D_s$. In this calculation, we consider only the terms common between nonseed and seed documents. For example, for the nonseed document d_1, $c_{12} = 0.417$ and $c_{15} = 0.083$. Since $c_{12} > c_{15}$, d_1 will join the cluster initiated by d_2. If we proceed in this manner, the generated clusters will be $C_1 = \{d_1, d_2\}$ and $C_2 = \{d_3, d_4, d_5\}$.

2.6.2 *Characteristics of the C^3M Algorithm.* C^3M satisfies the desirable characteristics of a good clustering algorithm in the following ways:

(1) It has been shown experimentally [2, 17] that the clusters produced are stable. That is, small errors in the description of documents lead to small changes in clustering, since small changes in the D matrix lead to small changes in the C matrix. Instability is a problem in some graph theoretical hierarchical algorithms (e.g., average link) [29, p. 28].
(2) From the definition of the C matrix, the algorithm is independent of the order of documents and produces a unique classification. The hierarchical clustering methods of complete linkage and average link, for example, do not necessarily define a unique dendrogram [29, p. 27].
(3) The algorithm requires a very small memory space for the data structures needed in the implementation. The memory space needed for the inverted index of the select terms is proportional to $n_2 \times t_{gs}$, which is very small. For example, for the INSPEC database our n_2 and t_{gs} values are 7,435 and 3.10, respectively. The space requirement for the other data structures is proportional to m or n [7]. Assignment of documents is performed efficiently by computing CC values between nonseed documents and the seed documents with the use of inverted indexes. On the other hand, a graph theoretical approach to clustering would require calculation of similarity coefficients that requires $(m^2 - m)/2$ memory entries plus the cost of cluster generation resulting in a time complexity of $O(m^2)$ [30, 32].
(4) It has been experimentally observed that this algorithm distributes documents evenly among clusters. In other words, it does not create a few "fat" clusters and a lot of singletons, a classical problem encountered in clustering.

2.7 Indexing–Clustering Relationships Obtainable from the CC Concept

The CC concept indicates some relationships between indexing and clustering. The indexing–clustering relationships show us, in a predictable manner, that change in the size of indexing vocabulary impacts the number of clusters. This can be used to tailor the number of clusters and hence the size of the clusters according to the parameters of the computer system that will be used to search the clusters. The computer system can be a multiprocessor or a uniprocessor with set memory or partition sizes. To obtain an acceptable cluster size, the clustering–indexing relationships provide tools (i.e., analytical relationships) to achieve the requirements. For example, as we will see in this section, the ratio (size of indexing vocabulary)/(average indexing exhaustivity) gives us the average

number of clusters, whereas the average size of a cluster is given by average term generality.

In this section we show analytical derivation of these relationships using binary indexing. In Section 3.2 these relationships are experimentally verified using both binary and weighted indexing.

For the derivation of the indexing–clustering relationships, consider eqs. (2.2) and (2.8), and find n_c:

$$n_c = \sum_{i=1}^{m} \delta_i = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij}^2 \times \alpha_i \times \beta_j. \tag{2.12}$$

In the case of binary indexing, $d_{ij}^2 = d_{ij}$. By substituting the values of α_i (eq. (2.3)) and β_j (eq. (2.4)) in eq. (2.12), we obtain the following:

$$n_c = \sum_{i=1}^{m} \sum_{j=1}^{n} \left[d_{ij} \times \left[\sum_{k=1}^{n} d_{ik} \right]^{-1} \times \left[\sum_{k=1}^{m} d_{kj} \right]^{-1} \right]. \tag{2.13}$$

In the IR literature the summations

$$\sum_{k=1}^{n} d_{ik} \quad \text{and} \quad \sum_{k=1}^{m} d_{kj}$$

are called, respectively, the depth of indexing x_{d_i} for document d_i and term generality t_{g_j} for term t_j. With these definitions, eq. (2.13) becomes

$$n_c = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} \times [x_{d_i} \times t_{g_j}]^{-1}. \tag{2.14}$$

In order to proceed, we need to define the average values for depth of indexing x_d and term generality t_g for the database:

$$x_d = \sum_{i=1}^{m} \frac{x_{d_i}}{m}, \qquad t_g = \sum_{j=1}^{n} \frac{t_{g_j}}{n}. \tag{2.15}$$

$x_{d_i} \times t_{g_j}$ can then be roughly approximated by $x_d \times t_g$, and (2.14) can be rewritten as

$$n_c = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} \times [x_d \times t_g]^{-1} = t \times [x_d \times t_g]^{-1}. \tag{2.16}$$

Equation (2.16) indicates the relationships among the quantities of the number of clusters, n_c; the total number of term assignments, t; the average depth of indexing, x_d; and the average term generality, t_g.

In eq. (2.16) if we substitute t/n for t_g and t/m for x_d, then n_c could be written as follows:

$$n_c = \frac{m \times n}{t} = \frac{m}{t_g} = \frac{n}{x_d}. \tag{2.17}$$

This relationship is intuitively justifiable. If the number of nonzero entries in the D matrix is increased, the similarity among documents is increased; hence, a smaller n_c results. The reverse, that is, more zero entries, would imply higher n_c.

Therefore, the relationship of eq. (2.17) can be used to check the clustering tendency of a document collection. Testing of clustering tendency in IR is a measure of the extent to which the use of a clustering method on a document database is likely to lead to an increase in retrieval effectiveness with respect to FS [9, p. 361; 32].

Using d_c and d_c' to indicate the average size of a document cluster and term cluster, respectively, we can write the following equations:

$$d_c = \frac{m}{n_c} = \frac{1}{\delta} = \frac{m}{(m/t_g)} = t_g, \tag{2.18}$$

$$d_c' = \frac{n}{n_c'} = \frac{1}{\delta'} = \frac{n}{(n/x_d)} = x_d. \tag{2.19}$$

Equations (2.18) and (2.19) show that t_g and x_d are the basic determinants of document cluster and term cluster sizes, respectively. The relationship (2.18) indicates that, on the average, individual clusters would contain a reasonable number of documents. On the contrary, clusters with extreme sizes would not be appropriate for browsing. This has been verified in our experiments of the INSPEC database, where only 3.5 percent of the database (448 clusters) needed to be searched based on our clustering methodology. Each cluster had an average of 28 documents. These values were obtained from the indexing–clustering relationships and were subsequently verified by the experiments (see Table III, Section 3.2).

The value range of n_c, indicated by the indexing–clustering relationships, is consistent with the theoretical expectation; that is, $1 \leq n_c \leq \min(m, n)$ (see Section 2.4). This is because $\max(t)$ is equal to $(m \times n)$ and $\min(t)$ is equal to $\max(m, n)$ [7].

If we apply eq. (2.16) to the example D matrix of Section 2.1, we obtain $n_c = 15/(3 \times 2.5) = 2$. Similarly, by using the expression in eq. (2.17), $n_c = (5 \times 6)/15 = 2$. In other words, under the CC, the number of clusters depicted by the indexing–clustering relationships is very close to the theoretically expected value, that is, 1.95 (see Section 2.4). The same is also true for eqs. (2.18) and (2.19).

So far we have derived the indexing–clustering relationships (eqs. (2.16)–(2.19)) indicated by the CC concept by using a binary D matrix. However, as shown in Section 3.2, these relationships are also valid for weighted indexing, with the exception of a little distortion introduced by term weights.

3. EXPERIMENTAL DESIGN AND EVALUATION

In this section we present three sets of experiments:

(1) validity experiments to test the validity of indexing–clustering relationships,
(2) usability experiments to verify time efficiency of C^3M for very large databases and to verify validity of the generated clustering structures, and
(3) IR experiments to measure the performance of C^3M by using various term-weighting approaches in query–document matching.

Item (3) involves evaluation of effectiveness as well as of efficiency. However, the goal of this paper is limited to the evaluation of the validity and effectiveness

of the concepts and/or methodologies that have been introduced via the CC concept.

3.1 Document Databases

In this study we use two document databases: TODS214 and INSPEC. The TODS214 database contains the papers published by the Association for Computing Machinery in the journal *Transactions on Database Systems* during March 1976 to September 1984. The database consists of 214 documents. Each document of TODS214 contains the title, keywords given by the author(s), and the abstract. The indexing vocabulary is generated automatically from within the documents of the database. The details of the indexing software can be found in [19].

The second database, INSPEC, contains 12,684 documents, covering computer and electrical engineering areas. The D matrix and the queries of this collection are provided by Cornell University with the permission of INSPEC. The characteristics of the databases and the queries are provided in Sections 3.2 and 3.4.3.

3.2 Experimental Validation of Indexing–Clustering Relationships: Validity Experiments

For the experiments in this section, we use the binary and weighted versions of various D matrices of TODS214 and INSPEC. In the experiments the entries of a weighted D matrix, d_{ij}, indicate the number of occurrences of t_j in d_i. The binary version of the same D matrix converts all nonzero d_{ij} values to one.

Table II provides the information pertaining to the generation of D matrices for the INSPEC database. In the table the frequency pair (Min, Max) indicates the frequency constraints that establish a (word) stem as a term. For example, the first row of the table indicates that a stem must occur at least in one and at most in 5,851 documents to be used as a term. Actually, for this case all stems are used as a term, since the maximum observed stem frequency is 5,851. This makes the cardinality of the indexing vocabulary 14,573, as indicated by n; t is the number of nonzero entries in the corresponding D matrix.

The same approach is applied to the TODS214 database. The equivalent of D_{I1} for the TODS214 database is $D_{1\text{-}180}$. Like D_{I1}, the indexing vocabulary of $D_{1\text{-}180}$ contains all stems. For this case, the minimum and maximum term frequency constraints are 1 and 180, respectively, n is 2,064, and t is 11,048. Nine other D matrices of TODS214 are obtained using the nine combinations of the three minimum term frequencies (2, 3, 4) with the three maximum term frequencies (20, 30, 40) [7].

Table III summarizes the experiments on indexing–clustering relationships for the INSPEC database. The second column gives the estimated number of clusters (see eq. (2.17)); n_{cb} and n_{cw} indicate the number of clusters generated using the CC concept (eq. (2.8)) for the binary and weighted versions of the corresponding D matrices, respectively. Similarly, d_{cb} and d_{cw} indicate the average size of a document cluster for the binary and weighted matrices, respectively. The average size of a term cluster is given by d'_{cb} and d'_{cw} for the respective (binary and weighted) term clusters.

As would be expected, the sparsity of the D matrix determines the number of clusters: The more sparse the D matrix is, the more clusters it contains. For

Table II. Characteristics of the D Matrices for the INSPEC Database

D matrix	Term frequency		n	t
	Min	Max		
D_{I1}	1	5,851	14,573	412,255
D_{I2}	2	5,851	7,435	405,117
D_{I3}	2	3,000	7,431	387,811
D_{I4}*	2	1,000	7,382	308,886
D_{I5}	3	5,851	5,677	401,601
D_{I6}	3	3,000	5,673	384,295
D_{I7}*	3	1,000	5,624	305,370

An asterisk indicates that for that matrix m is equal to 12,682.

Table III. Results of the Indexing–Clustering Relationships Experiments for the INSPEC Database

Matrix	$m \times n$ / t	n_{cb}	n_{cw}	t_g	d_{cb}	d_{cw}	x_d	d'_{cb}	d'_{cw}
D_{I1}	448	439	475	28.29	28.89	26.70	32.50	33.20	30.68
D_{I2}	233	227	275	54.49	55.88	46.12	31.94	32.75	27.04
D_{I3}	243	238	294	52.19	53.29	43.14	30.58	31.22	25.28
D_{I4}	303	294	364	41.84	43.14	34.84	24.36	25.11	20.28
D_{I5}	179	175	218	70.74	72.48	58.18	31.66	32.44	26.04
D_{I6}	187	183	233	67.74	69.31	54.44	30.30	31.02	24.35
D_{I7}	234	227	289	54.30	55.88	43.88	24.08	24.78	19.46

example, the original description of the INSPEC database, shown by matrix D_{I1} of Table II, has a size of 12,684 by 14,573 and contains 412,255 nonzero entries; the estimated n_c is 448. Matrix D_{I2} has a size of 12,684 by 7,435 and contains almost the same number of nonzero entries (405,117) as D_{I1}. This means that the sparsity of D_{I2} is half that of D_{I1} (or its density is twice that of D_{I1}). As a result, the estimated n_c drops from 448 to 232; that is, the estimated n_c of D_{I2} is only 52 percent of the estimated n_c of D_{I1}.

The results of the experiments show that the indexing–clustering relationships hold very closely in the case of binary indexing. In the weighted case, the weights have slightly perturbed the indexing–clustering relationships. For example, the value of n_c, estimated by the relationships and expressed by eqs. (2.16) and (2.17), is very close to the n_{cb} value: On the average, n_{cb} and n_{cw} values are, respectively, 2.37 percent less than and 19.37 percent greater than the estimated n_c values. That is, the estimated n_c and n_{cb} assume very close values. Similarly, as indicated by eq. (2.18), t_g and the average size of clusters (d_{cb}) assume very close values. For example, the t_g and d_{cb} entries of the D_{I1} matrix of INSPEC assume the following respective values: 28.29 and 28.89. The validity of eq. (2.19), that is, the relationship between the average size of a term cluster (d'_{cb}) and x_d, can also be seen from Table III. For example, the x_d and d'_{cb} entries of the D_{I1} matrix of INSPEC assume the following respective values: 32.50 and 33.20. The

relationships between (t_g, d_{cw}) and (x_d, d'_{cw}) are still valid; however, they are perturbed by the weights of terms.

The experimental observations for the 10 D matrices of the TODS214 database are very similar [7]. For the binary D matrices of the TODS214 database, the estimated n_c and n_{cb} values are identical in 6 of the 10 experiments and differ only by 1 in the remaining 4 cases. In other words, on the average, n_{cb} and n_{cw} values are, respectively, 1.03 percent less than and 12.87 percent greater than the estimated n_c values. The average size of document clusters, d_{cb} and d_{cw}, are, respectively, 1.42 percent greater and 10.92 percent less than their values predicted via t_g. The same kind of relationships between (d'_{cb}, d'_{cw}) and x_d are observed because of the interconnection of the indexing–clustering relationships.

3.3 Usability of the Clustering Algorithm and the Clusters Generated: Usability Experiments

Before employing a clustering algorithm, we have to show that the algorithm and its clustering structure are usable in the intended environment. Due to huge sizes of document collections, to be useful, a clustering algorithm for IR must be time efficient and must produce valid (meaningful) clusters. As we will see shortly, C^3M satisfies both of these constraints. We will also see that the running time of C^3M is proportional to the complexity of the algorithm. In these analyses we use the INSPEC database to enable us to relate to experiments published by other researchers.

3.3.1 *The First Usability Constraint: Time Efficiency.* The clustering algorithm was coded in FORTRAN77 and run on an IBM 4381 Model 23 computer using the VM/SP operating system. The execution times needed to cluster various D matrices of the INSPEC database are shown in Table IV. The execution times vary between 189 s and 74 s, depending on the characteristics of the corresponding D matrices. The results indicate that C^3M is time efficient and can easily be used for very large databases. The clustering time needed for the INSPEC (D_{11}) database by various graph theoretical algorithms on an IBM 3083 BX environment is reported in [10, Tab. 1]. This IBM computer is a faster model than the IBM 4381 computer we have used in our experiments [14]. The reported execution times of the graph theoretical algorithms vary between 840 to 3,416 s. These values are 4.4 to 18.1 times larger than the execution times of C^3M. El-Hamdouchi and Willett [10] have tried using a version of the complete linkage method, which is known to have good IR performance [29, 31]. Due to the huge time and memory requirements encountered, they were unable to use that particular version of the complete linkage algorithm. Voorhees describes the implementation details of the same algorithm for the INSPEC database; however, no accurate clustering time performance statistics are available [30].

Based on the results just shown, the execution time of the C^3M algorithm is lower than those of most other algorithms recently used and/or referenced in the IR literature.

Now let us look at the consistency between our complexity analysis and the execution times of the C^3M algorithm for various matrices of the INSPEC database. Our complexity analysis indicates that the run time is proportional to ($m \times x_d \times t_{gs}$). In other words, the ratio ($m \times x_d \times t_{gs}$)/(execution time) would

Table IV. Execution Time Behavior of the C^3M Algorithm for the INSPEC Database

D matrix	D_{I1}	D_{I2}	D_{I3}	D_{I4}	D_{I5}	D_{I6}	D_{I7}
Execution time (e.t.), s	189.00	126.00	114.00	85.00	105.00	95.00	74.00
t_{gs}	3.10	1.73	1.80	1.86	1.81	1.86	1.94
x_d	32.50	31.94	30.58	24.36	31.66	30.30	24.08
$r = (t_{gs} \times x_d)/(\text{e.t.})$	0.53	0.44	0.48	0.53	0.55	0.59	0.63

be a constant across different D matrices. For all of the D matrices, D_{I1}–D_{I7}, m is practically the same (either 12,684 or 12,682; see Table II). Therefore, factor m in this ratio can be ignored. Table IV shows t_{gs}, x_d, and the ratio $r = (x_d \times t_{gs})/(\text{execution time})$.

The value of r in Table IV varies between 0.44 and 0.63 and has a mean of 0.54, with a coefficient of variation (standard deviation/mean value) of 0.12. The low value of the coefficient of variation confirms that the expected running time of the C^3M algorithm is proportional to $(m \times x_d \times t_{gs})$, which is considerably less than those of most other clustering algorithms.

3.3.2 *The Second Usability Constraint: Validity of the Clustering Structure.* The term *cluster validity* refers mainly to objective ways of deciding whether a clustering structure represents the intrinsic character of a clustered data set. The two other related problems, clustering tendency and validity of individual clusters, are beyond the scope of this study. A comprehensive coverage of the cluster validity problem and an extensive list of references are given in [15]. A brief overview of the problem from the viewpoint of IR is presented in [32].

A typical cluster validity approach tries to show that the clustering structure is significantly different from random clustering. Most of the known methods of cluster validation are inapplicable in the IR environment. This is mostly due to the very large size of document databases. The nature of the IR problem also introduces some conceptual problems [7]. In IR partitioning-type clustering environments, we can use the following approach to test validity: Given a query, let a target cluster be defined as a cluster that contains at least one relevant document for the query. Let n_t indicate the average number of target clusters for a set of queries for a given clustering structure. If we preserve the same clustering structure and assign documents randomly to the clusters, then we obtain random clustering. Intuitively, for a valid clustering structure, n_t should be significantly less than the average number of target clusters under random clustering, n_{tr}.

The number of target clusters for individual queries in the case of random clustering can be obtained by using the following theorem:

THEOREM. *Consider a partition of m documents with n_c number of clusters and with each cluster having a size of $|C_j|$ for $1 \le j \le n_c$. If k documents are randomly selected from m documents, the probability P_j that cluster C_j will be selected is given by*

$$P_j = \left[1 - \prod_{i=1}^{k} \frac{m_j - i + 1}{m - i + 1}\right],$$

where $m_j = m - |C_j|$.

Table V. The Comparison of C³M and Random Clustering in Terms of Average Number of Target Clusters for All Queries for the INSPEC Database

D matrix	D_{I1}	D_{I2}	D_{I3}	D_{I4}	D_{I5}	D_{I6}	D_{I7}
n_t	24.455	23.299	23.636	24.364	22.961	23.039	23.662
n_{tr}	30.807	29.580	29.715	30.379	28.769	29.005	29.765

PROOF. The proof follows steps similar to those in the proof of Yao's theorem [33] for estimating block accesses. Due to limited space, we refer the reader to [33]. □

Accordingly, in random clustering for a query with k relevant documents the number of target clusters is equal to the summation $(P_1 + P_2 + \cdots + P_{n_c})$. Thus, finding n_{tr}, the average number of target clusters for all queries, is straightforward. The case $n_t \geq n_{tr}$ suggests that tested clustering structure is invalid, since it is unsuccessful in placing the documents relevant to the same query into fewer number of clusters than that of the average random case. The case, $n_t < n_{tr}$, is an indication of the validity of the clustering structure. Table V gives the n_t and n_{tr} values for all D matrices of the INSPEC database. The observations for the TODS214 database are similar and are given in [7]. The characteristics of the queries for both databases can be found in Section 3.4.3. For all the D matrices of both databases, the n_t values are lower than the n_{tr} values. However, to decide the validity of a clustering structure we must show that the n_t value is significantly lower than the corresponding n_{tr} value. In other words, we need to show that the clustering structure is significantly nonrandom. For this purpose we generated 10,000 random clustering structures for all the D matrices of the TODS214 database and obtained the average number of target clusters for each random case, $n_t(r)$. From this we constructed a histogram of the observed $n_t(r)$ values. The same was done for the D_{I1} matrix of the INSPEC database. Due to the long computational requirements of the INSPEC database, we generated 5,000 random cases.

The 5,000 $n_t(r)$ values of INSPEC were grouped into 10 bins. Figure 5 shows the percentage counts of each bin. That is, Figure 5 shows the approximate baseline distribution (probability density function) of the $n_t(r)$ values. The plot shows that the n_t value of 24.455 for the INSPEC database is significantly different from the random case, since all of the observations have a value greater than n_t. The same is observed for all of the D matrices of the TODS214 database. This shows that the clusters generated by C³M are not an artifact of the algorithm. On the contrary, they are valid. The answer to the remaining question about the effectiveness of CBR is given next.

3.4 IR Experiments

3.4.1 *Evaluation Measures for Retrieval Effectiveness.* In this paper we assess the effectiveness of C³M by comparing its CBR performance with that of FS and with the CBR performance of other clustering algorithms used in the current IR literature. The effectiveness measures used in this study are the average precision[1]

[1] *Precision* is defined as the ratio of the number of retrieved relevant documents to the number of retrieved documents.

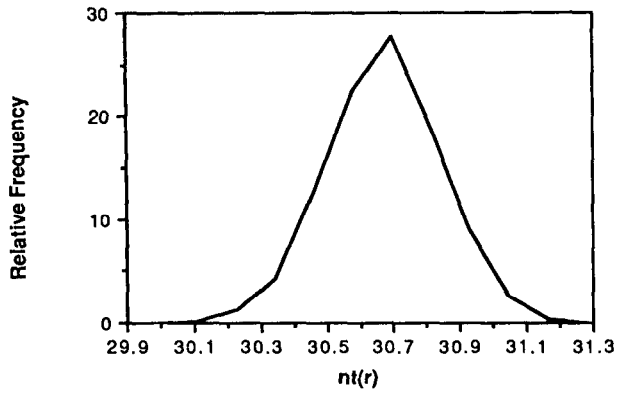

Fig. 5. Histogram of the $n_t(r)$ values for the INSPEC database (for D_{11}) (min = 30.051; max = 31.221; average = 30.659; standard deviation = 0.172; bin length = 0.117).

for all queries; the total number of relevant documents retrieved for all queries, T; and the total number of queries with no relevant documents, Q [10, 11, 29]. The effectiveness measures are obtained after retrieving 10 and 20 documents. Typically, a user will evaluate the relevance of the first 10–20 documents returned by the system, and after that, either he or she is satisfied or the query is reformulated. An effective system yields higher precision and T and lower Q. The effectiveness measures precision and T provide similar results; this is shown analytically in [7]. However, both effectiveness measures are used to compare our results with those of [10], [29], and [31].

In CBR, first the clusters are ranked according to their similarity with the user query. Then the d_s (10 or 20) documents of the first n_s clusters are selected according to their similarity to the query. The selected clusters and documents must have nonzero similarity to the query.

3.4.2 *Term Weighting or Query Matching Function Selection.* Term weighting has basically three components: the term frequency component (*TFC*), the collection frequency component (*CFC*), and the normalization component (*NC*) [24]. The weights of the terms of a document and a query (denoted by w_{d_j} and w_{q_j}, $1 \leq j \leq n$, respectively) are obtained by multiplying the respective weights of these three weighting components. After obtaining the term weights, the matching function for a query and a document is defined as the following vector product:

$$\text{similarity (Q, D)} = \sum_{k=1}^{n} w_{d_k} \times w_{q_k}.$$

A brief description of *TFC*, *CFC*, and *NC* follows, and more details can be found in [24].

Consider d_{ij}. The weight w_{d_j} of term t_j in d_i will be ($TFC \times CFC \times NC$). The three possibilities for *TFC* are symbolized by b, t, and n: b is binary weight; in this case, ignore the term frequency, and take $TFC = 1$; t is term frequency, which means that *TFC* is equal to the number of occurrences of t_j in d_i; n is the

augmented normalized term frequency and is defined as $(0.5 + 0.5 \times t/\max t)$, where $\max t$ is $\max(d_{i1}, d_{i2}, \ldots, d_{in})$, that is, the maximum number of times any term appears in d_i.

The three possibilities for *CFC* are denoted by *x*, *f*, and *p*: *x* indicates no change. Hence, take $CFC = 1$; *f* is inverse document frequency, and in this study it is taken as $\ln(m/t_{g_j}) + 1$ for document and query terms. As defined earlier, t_{g_j} is the number of documents containing t_j. For centroids it is defined as $\ln(n_c/x_j) + 1$; x_j indicates the number of centroids containing t_j. In centroids it is possible to have $n_c = x_j$ for some terms, which leads to $\ln(n_c/x_j) = 0$. We therefore have to add the constant 1 to the formula; *p* is the probabilistic inverse collection frequency factor and is similar to *f* both in terms of definition and performance [24]. We did not use it in our experiments.

For normalization, that is, the *NC* component, there are two possibilities, denoted by *x* and *c*: *x* means no change; that is, take $NC = 1$; *c* means cosine normalization, where each term weight ($TFC \times CFC$) is divided by a factor representing Euclidean vector length. The normalization of query terms is insignificant, since it does not change the relative ranking of documents (and centroids).

The various combinations of the term-weighting components yield different matching functions. For example, $TW1$, that is, the combination *txc* ($TFC = t$, $CFC = x$, and $NC = c$) and *txx*, respectively, for documents and queries yields the well-known cosine function used in the IR literature. The combinations *tfc* and *nfc* for documents and *nfx*, *tfx*, and *bfx* for queries have been determined to result in better IR performance [24]. These provide us with the six different matching functions (in our experiments the term-weighting components for documents and centroids are the same): *tfc* · *nfx*, *tfc* · *tfx*, *tfc* · *bfx*, *nfc* · *nfx*, *nfc* · *tfx*, and *nfc* · *bfx*. In this paper we use TW_i $(1 \leq i \leq 7)$ to represent the cosine coefficient and the other six term-weighting approaches. Table VI shows these seven combinations, that is, the query-matching functions used in our experiments. Each of these matching functions enables us to test the performance of C^3M under a different condition.

3.4.3 *Generation of Cluster Centroids and Query Characteristics.* To perform CBR we need cluster representatives or centroids. The terms with the highest total number of occurrences within the documents of a cluster are chosen as centroid terms. The maximum length (i.e., number of distinct terms) of centroids is provided as a parameter. The weight of a centroid term is defined as its total number of occurrences in the documents of the corresponding cluster. As would be expected, centroid length affects the effectiveness of CBR [29, 31]. In this study, our concern is not to find the best centroid length for CBR, but to use one that will give us some indication of IR effectiveness.

In our experiments, we have used the weighted version of $D_{1\text{-}180}$ and D_{11} matrices of the TODS214 and the INSPEC databases, respectively. The reason for this choice is that the D_{11} matrix is the original description of INSPEC, which is commonly used in the IR literature, and enables us to relate to the works of other researchers. In D_{11} all the stems are chosen as terms, as are those in the matrix $D_{1\text{-}180}$. This provides a common characteristic for both matrices.

Table VI. Term-Weighting Approaches Used in the Experiments

Abbreviation	TW1	TW2	TW3	TW4	TW5	TW6	TW7
Meaning	*txc · txx*	*tfc · nfx*	*tfc · tfx*	*tfc · bfx*	*nfc · nfx*	*nfc · tfx*	*nfc · bfx*

Table VII. Characteristics of the Centroids

Database	Centroid length (max x_c)	Average centroid length x_c	t_{gc}	$\%X_c$	$\%n$	%D
TODS214	50	49.47	2.83	24	30	16
	100	94.81	3.06	46	54	30
	150	135.22	3.39	65	70	44
	200	163.69	3.51	79	81	53
INSPEC	250	237.09	12.96	51	60	27
	500	399.06	14.74	86	88	46

For the TODS214 database, we have used the maximum centroid lengths 50, 100, 150, and 200; and for the INSPEC database, 250 and 500. The characteristics of the centroids generated for both databases are shown in Table VII. In Table VII, the symbol x_c indicates the average number of distinct terms per centroid (average centroid length), t_{gc} indicates the average number of centroids per term, $\%x_c$ indicates the percentage of the distinct cluster terms used in the centroids, and $\%n$ indicates the percentage of D matrix terms that appear in at least one centroid (i.e., centroid terms). The last entry, %D, indicates the total size of centroid vectors as a percentage of t of the corresponding D matrix.

For the INSPEC database, the effectiveness results of CBR with the centroid length 250 are slightly better than those with 500. For the TODS214 database, the IR performance increases slightly from the centroid length of 50 to 150, and after this point it almost remains the same. These observations lead us to the following conclusion: The increase in centroid length first increases the IR effectiveness, but after, a threshold provides no further improvement. This is also consistent with the findings of Voorhees [31, Tab. 4]. In this study the performance figures for TODS214 and INSPEC are presented using the centroid lengths of 150 and 250, respectively. With this choice, the disk space requirement for the centroids in the INSPEC database is only 27 percent of the original D matrix. Considering the storage requirements of the various hierarchical clustering algorithms implemented for the same database [29, pp. 110–112; 31], this percentage is quite reasonable. For example, the centroid file external storage requirement of a complete linkage algorithm is 2.85 times more than that of the D matrix of the INSPEC database [31, Tab. 2].

The query characteristics are presented in Table VIII. In both of the databases, query vectors are created in the same manner as the D matrices. The queries of the INSPEC database were collected at Cornell University and Syracuse

Table VIII. Characteristics of the Queries

Database	Number of queries	Average terms per query	Average relevant documents per query	Total relevant documents	Number of distinct documents retrieved	Number of distinct terms for query definition
TODS214	58	13.36	5.26	305	126	291
INSPEC	77	15.82	33.03	2,543	1,940	577

University. Further information for TODS214 queries is given in [7]. The query vectors of TODS214 are binary, since most of the terms appear only once. In the INSPEC case, the information on term frequencies is available and is incorporated with the query-matching functions whenever needed.

3.4.4 *The Effectiveness of CBR*

3.4.4.1 *INSPEC Database.* The first step in CBR is to determine the number of clusters n_s to be selected. In general, retrieving more clusters will be more effective, but also more expensive. The increase in effectiveness would be expected to increase up to a certain n_s, and after this (saturation) point, the retrieval effectiveness remains the same or improves very slowly. In the IR experiments of INSPEC, for the term-weighting strategies of TW1–TW7 the saturation point is observed at $n_s = 50$, which corresponds to 10.5 percent of the total number of clusters (for the INSPEC database $n_{cw} = 475$). For all of the queries and for all of the matching functions TW1–TW7, the average percentage of the matched documents is 12.5 percent, which is slightly higher than 10.5 percent, the percentage of the selected clusters. These comparable values also indicate that documents are evenly distributed among clusters.

The performance figures for each TW2, TW3, and TW4 are similar, with TW2 being the best and TW4 being the worst. The same is true for each TW5, TW6, and TW7, with TW5 being the best and TW7 being the worst. This is valid for all of the observed n_s values. Therefore, TW2 and TW5 qualify as the best examples of their groups.

The changes in precision when we retrieve 10 documents for INSPEC are shown in Figure 6 for TW1, TW2, and TW5 with different values of n_s. The horizontal lines indicate the precision values for the corresponding FS. The results indicate that the increase in n_s improves effectiveness. The same trend of improvement is also valid for the other effectiveness measures (i.e., the increase in T and decrease in Q). The plot also indicates that the best term-weighting strategy (i.e., matching function) is TW2. CBR with TW2 results in better effectiveness than that of FS for all of the other matching functions beginning at the number of selected clusters value, n_s, of 40. TW2 outperforms TW1 and TW5 beginning at $n_s = 30$. The case with $n_s = 50$ returns 12.5 percent of the documents to examine, which is very low with respect to FS and is the retrieval saturation point. Because of these and to decrease the volume of the presented data, the CBR effectiveness will be analyzed at $n_s = 50$ for the INSPEC database. Figure 6 shows the general picture for the other n_s values.

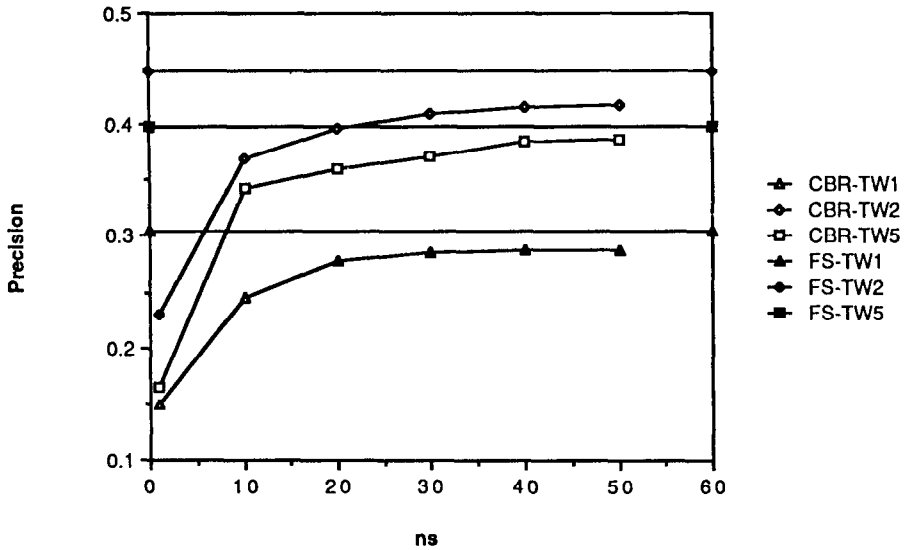

Fig. 6. Precision with 10 documents ($d_s = 10$) for different values of n_s (INSPEC).

Table IX shows the effectiveness figures for the INSPEC database. The first and second rows of each item of Table IX are for d_s values of 10 and 20, respectively. Since the effectiveness figures T and precision result in relatively identical effectiveness [7], it suffices to consider only one of them (i.e., either precision or T). We will consider precision.

The experimental results indicate that the matching function is very critical to retrieval effectiveness. The decrease of the CBR precision with respect to the FS precision varies from 1.0 percent ($d_s = 10$ of TW6) to 6.9 percent ($d_s = 10$ of TW2).

For all cases, the average percentage decrease in precision of CBR with respect to FS is 3.9. TW2 gives the highest percentage decrease (6.9 percent) in precision with respect to its own FS; however, when $d_s = 10$, the CBR precision (.418) is greater than those of FS for the other matching functions (between 37.1 percent to 0.2 percent with an average of 11.5 percent). For $d_s = 20$, the CBR precision (.336) of TW2 is greater than those of other FS results, except for TW4 (whose precision value is .342).

The difference (average Q values for CBR) − (average Q values for FS) for 14 observations is 1.1, which is very low. The effectiveness in terms of Q can be assumed to be almost identical to that of FS.

Let us compare our results with the other algorithms tested on the INSPEC database. Voorhees reports effectiveness experiments in [29] and [31]. Since the results of [31, Tab. 4] are better, we consider them first. This work [31] reports the results of a complete linkage algorithm with three different centroid lengths. The Q values of [31] are similar to our case. The only effectiveness measure that remains to be compared is precision. The precision values for FS are .384 and .315, respectively, for $d_s = 10$ and 20. The CBR precision values for $d_s = 10$ are

Table IX. Effectiveness of FS and CBR for the INSPEC Database

Effectiveness measures	TW1		TW2		TW3		TW4		TW5		TW6		TW7	
	FS	CBR	FS	CBR	FS	CBR	FS	CBR	FS	CBR	FS	CBR	FS	CBR
Precision	.305	.287	.449	.418	.412	.396	.417	.401	.399	.388	.386	.382	.357	.352
	.253	.230	.358	.336	.334	.323	.342	.329	.321	.310	.319	.313	.292	.288
T	235	221	346	322	317	305	321	309	307	299	297	294	275	271
	390	354	552	518	515	497	526	506	494	477	491	482	450	443
Q	18	17	5	6	7	8	5	6	2	5	6	9	6	7
	11	13	2	3	4	3	3	3	2	4	4	4	2	4

(.387, .397, .373) and for $d_s = 20$ are (.308, .314, .284) for three different centroid lengths. Even though there is no exact definition, the matching function of the referenced study is similar to our TW6. The best CBR result (.397, when $d_s = 10$) and the worst CBR result (.284, when $d_s = 20$) are 3.4 percent better and 9.8 percent worse than that of FS, respectively. The worst case of our experiments in terms of relative percentage difference in precision (with respect to FS) is 6.9 percent for TW2. Interestingly, as we have stated before, it gives the best CBR results, that is, .418 (for $d_s = 10$) and .336 (for $d_s = 20$), which are 5.3 percent (100 × .418/.397) and 7.0 percent (.336/.314) better than the best results of Voorhees for $d_s = 10$ and 20, respectively. Obviously, if Voorhees had changed the matching function, better results may have been obtained, although such improvement is not guaranteed. This discussion indicates that our results are compatible with the results of [31] on the INSPEC database.

We should also point out that all of our CBR results are better than those performed by using the single link and average link hierarchical clustering methods reported in [29, p. 88]. However, the matching functions of the compared experiments are not precisely the same.

Next, let us compare our experiments with the results of El-Hamdouchi and Willett [10], performed on the INSPEC database. Their study used the binary version of the INSPEC database to create four different types of hierarchical clustering structures. The algorithms used are single link, complete linkage, average link (in [10] this is called group average), and the Ward method. Their complete linkage algorithm is different from the one reported in [29] and [31], since, as we have mentioned before, they were unable to use that version due to its excessive CPU time and main memory requirements.

The study reports the results of three different CBR search strategies [10, Tab. 3]. The effectiveness measures that are common with our study are T and Q. In [10] the cosine similarity coefficient is used as the matching function, with the query terms weighted using inverse document frequency. Both the D matrix and query vectors are treated as binary. To obtain a comparable CBR environment, we have incorporated the same matching function in our experiments.

The "best" T and Q results for single link (SL), complete linkage (CL), average link (GA), and the Ward method (WM) versus our C^3M results are shown in Table X and are based on the number of selected clusters, n_s, of 10 and 50. All of the values of Table X are obtained after examining 10 documents ($d_s = 10$). The results of the table indicate that C^3M always outperforms the hierarchical algorithms. For $n_s = 10$, the percentage improvement in T ranges from 15.1 to 63.5, with an average of 47.5; and the decrease in Q (the number of queries with no relevant documents retrieved) ranges from 44.4 percent to 60.5 percent, with an average of 54.7. Notice that $n_s = 10$ is only 2.1 percent of all the clusters ($n_c = 475$), and the number of documents within these selected clusters is 2.4 percent of the INSPEC database. (As a sidelight, we should also point out that in the comparisons we have used the "best" results of hierarchical clustering against our cases for only $n_s = 10$, a very modest comparison!)

3.4.4.2 *TODS214 Database.* The queries of TODS214 are binary. By definition, the matching functions TW2, TW3, and TW4 become equivalent, as is true for

Table X. Best T and Q Values for Various Hierarchical-Clustering Algorithms (taken from [10]) and C^{3}M with $n_s = 10$ and 50

Effectiveness measures	Hierarchical-clustering algorithms				C^{3}M	
	Single link (SL)	Complete linkage (CL)	Average link (GA)	Ward method (WM)	$n_s = 10$	$n_s = 50$
T	127	126	179	138	206	226
Q	37	38	27	33	15	12

Table XI. Effectiveness of FS and CBR for the TODS214 Database ($n_{cw} = 36$)

Effectiveness measures	TW1				TW4 (TW2, TW3)				TW7 (TW5, TW6)			
	FS	$n_s = 4$	$n_s = 6$	$n_s = 8$	FS	$n_s = 4$	$n_s = 6$	$n_s = 8$	FS	$n_s = 4$	$n_s = 6$	$n_s = 8$
Precision	.166	.160	.155	.159	.179	.186	.181	.179	.179	.193	.188	.186
	.110	.116	.116	.114	.128	.127	.124	.124	.124	.129	.128	.126
Q	18	25	25	21	19	23	21	21	21	20	23	21
	16	23	19	18	16	20	19	18	11	19	17	15

TW5, TW6, and TW7. Table XI shows the effectiveness measures (precision and Q) for the TODS214 database. (T is not shown because it results in relatively similar effectiveness as precision.) The first and second rows of each item indicate the cases for $d_s = 10$ and $d_s = 20$, respectively. As in the INSPEC case, TW1 is the worst matching function. For example, its FS precision value, for $d_s = 10$, is 0.166. This value is less than the precision of FS with TW4 and TW7 (for these the FS precision is 0.179). For TODS214 the best matching function is TW7. In the case of INSPEC, the best matching function is TW2. The difference can be attributed to the different nature of the queries used in the databases. As specified earlier, the INSPEC queries are weighted, and weights are used whenever they are needed. Table XI shows that CBR provides an increase in effectiveness. For example, CBR with TW7 improves effectiveness in terms of precision (the percentage improvement ranges from 1.6 to 7.8 with an average of 4.3).

4. CONCLUSION

In this study a new clustering methodology called C^{3}M has been introduced. C^{3}M relies on its heuristic, the cover coefficient (CC) concept, which indicates relationships among documents (or terms) based on a two-stage probability experiment. In this paper a hypothesis is introduced that gives a method of estimating the number of clusters in a document database. The hypothesis states that the number of clusters within a collection should be inversely proportional to the similarity among documents. This means that higher similarity among documents implies fewer clusters, or conversely, lower similarity implies more clusters. The CC concept satisfies this hypothesis and can be used to find the number of clusters indicated by the hypothesis. The CC concept is also used for the selection

of cluster seeds and to relate indexing and clustering analytically. In the experiments conducted, the relationships indicated by the CC concept are validated.

The complexity of C^3M is shown to be better than most other clustering algorithms, whose complexities range from $O(m^2)$ to $O(m^3)$. The experiments show that C^3M is time efficient and suitable for very large databases. Its low complexity is experimentally validated. C^3M has all the desirable properties of a good clustering algorithm.

In this paper a methodology of checking the validity of partitioning structure in IR query processing is introduced and used to validate the partitions produced by C^3M before using them for IR. The retrieval experiments show that the IR effectiveness of C^3M is compatible with a complete linkage method that is known to have good performance. It is known that the implementation of this version of the complete linkage method is very demanding in terms of computation time and memory, and can only be used by some approximations in very large database environments [30]. Our experiments also show that C^3M is 15.1 to 63.5 (with an average of 47.5) percent better than four other clustering algorithms in CBR. The experiments show that the computational cost of C^3M is several times less than that of these algorithms. It is also shown that C^3M improves retrieval effectiveness with respect to FS in the TODS214 environment. We have shown that the CC concept has good use in IR. Before concluding we provide a list of problems as a pointer toward future research:

(1) Regarding the indexing–clustering relationships shown in this paper, there remains the question: What is the effect of variations of indexing on CBR performance [9]?
(2) In an IR system, it may be worth combining FS and CBR, since they may return different sets of documents to the user [11]. Although this would increase the cost, this combination may be worth considering in order to increase effectiveness.
(3) The efficiency comparison of inverted index search and CBR may be worth pursuing [29, 31]. The simple clustering structure and the small sizes of centroids with respect to the D matrices of our experiments indicate that CBR with C^3M would be very efficient.
(4) Another study worth undertaking is the incorporation of the CC concept into incremental cluster maintenance based on static as well as dynamic indexing. Dynamic indexing is important for document databases. Our early findings on the maintenance issue are reported in [6] for the TODS214 database.

ACKNOWLEDGMENTS

The authors are grateful to the anonymous referees for suggesting improvements to the paper. We thank our colleagues Dr. Jon M. Patton for streamlining our proofs of the theorems given in [7] and Dr. James D. Kiper for commenting on readability. The authors are also indebted to Dr. Gerard Salton of Cornell University for helping us to acquire the INSPEC database and to Dr. Peter Willett of the University of Sheffield for providing timely results before they were actually published in the media.

REFERENCES

1. ANDERBERG, M. R. *Cluster Analysis for Applications.* Academic Press, New York, 1973.
2. CAN, F. A new clustering scheme and its use in an information retrieval system incorporating the support of a database machine. Ph.D. dissertation, Dept. of Computer Engineering, Middle East Technical Univ., Ankara, Turkey, 1985.
3. CAN, F., AND OZKARAHAN, E. A. A clustering scheme. In *Proceedings of the 6th Annual International ACM–SIGIR Conference.* (Bethesda, Md., June 1983) ACM, New York, 1983, pp. 115–121.
4. CAN, F., AND OZKARAHAN, E. A. Two partitioning type clustering algorithms. *J. Am. Soc. Inf. Sci. 35*, 5 (Sept. 1984), 268–276.
5. CAN, F., AND OZKARAHAN, E. A. Concepts of the cover coefficient based clustering methodology. In *Proceedings of the 8th Annual International ACM–SIGIR Conference* (Montreal, Quebec, June 1985). ACM, New York, 1985, pp. 204–211.
6. CAN, F., AND OZKARAHAN, E. A. Dynamic cluster maintenance. *Inf. Process. Manage. 25*, 3 (1989), 275–291.
7. CAN, F., AND OZKARAHAN, E. A. Effectiveness assessment of the cover coefficient based clustering methodology. Working Paper 89-002, Dept. of Systems Analysis, Miami Univ., Oxford, Ohio, Oct. 1989.
8. CROUCH, D. B. A file organization and maintenance procedure for dynamic document collections. *Inf. Process. Manage. 11*, 1 (1975), 11–21.
9. EL-HAMDOUCHI, A., AND WILLETT, P. Techniques for the measurement of clustering tendency in document retrieval systems. *J. Inf. Sci. 13*, 6 (1987), 361–365.
10. EL-HAMDOUCHI, A., AND WILLETT, P. Comparison of hierarchical agglomerative clustering methods for document retrieval. *Comput. J. 32*, 3 (June 1989), 220–227.
11. GRIFFITHS, A., LUCKHURST, C., AND WILLETT, P. Using interdocument similarity information in document retrieval systems. *J. Am. Soc. Inf. Sci. 37*, 1 (Jan. 1986), 3–11.
12. GRIFFITHS, A., ROBINSON, L. A., AND WILLETT, P. Hierarchical agglomerative clustering methods for automatic document classification. *J. Doc. 40*, 3 (Sept. 1984), 175–205.
13. HODGES, J. L., AND LEHMANN, E. L. *Basic Concepts of Probability and Statistics.* Holden-Day, San Francisco, Calif., 1964.
14. IBM. IBM 3083 processor complex. IBM Doc. G221-2417-0, IBM Corp., Armonk, N.Y., Mar. 1982.
15. JAIN, A. K., AND DUBES, R. C. *Algorithms for Clustering Data.* Prentice-Hall, Englewood Cliffs, N.J., 1988.
16. KUSIAK, A., AND CHOW, W. S. An efficient cluster identification algorithm. *IEEE Trans. Syst. Man Cybern. SMC-17*, 4 (July–Aug. 1987), 696–699.
17. KUTLUAY, M. S. A validity analysis of the cover coefficient concept on cluster analysis. M.S. thesis, Dept. of Electrical and Electronics Engineering, Middle East Technical Univ., Ankara, Turkey, 1986.
18. OZKARAHAN, E. *Database Machines and Database Management.* Prentice-Hall, Englewood Cliffs, N.J., 1986.
19. OZKARAHAN, E. A., AND CAN, F. An automatic and tunable indexing system. In *Proceedings of the 9th Annual International ACM–SIGIR Conference* (Pisa, Italy Sept., 1986). ACM, New York, 1986, pp. 234–243.
20. RASMUSSEN, E. M., AND WILLETT, P. Non-hierarchic document clustering using the ICL distributed array processor. In *Proceedings of the 10th Annual International ACM–SIGIR Conference* (New Orleans, La., June 1987). ACM, New York, 1987, pp. 132–139.
21. SALTON, G. Cluster search strategies and the optimization of retrieval effectiveness. In *The Smart Retrieval System—Experiments in Automatic Document Processing*, G. Salton, Ed. Prentice-Hall, Englewood Cliffs, N.J., 1971, pp. 223–242.
22. SALTON, G. *Dynamic Information and Library Processing.* Prentice-Hall, Englewood Cliffs, N.J., 1975.
23. SALTON, G. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, Reading, Mass., 1989.
24. SALTON, G., AND BUCKLEY, C. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage. 24*, 5 (1988), 513–523.

25. SALTON, G., AND MCGILL, M. J. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.
26. SALTON, G., AND WONG, A. Generation and search of clustered files. *ACM Trans. Database Syst. 3*, 4 (Dec. 1978), 321–346.
27. VAN RIJSBERGEN, C. J. *Information Retrieval.* 2nd ed. Butterworths, London, 1979.
28. VOORHEES, E. M. The cluster hypothesis revisited. In *Proceedings of the 8th Annual International ACM–SIGIR Conference* (Montreal, Quebec, June 1985). ACM, New York, 1985, pp. 188–196.
29. VOORHEES, E. M. The effectiveness and efficiency of agglomerative hierarchical clustering in document retrieval. Ph.D. dissertation, Dept. of Computer Science, Cornell Univ., Ithaca, N.Y., 1986.
30. VOORHEES, E. M. Implementing agglomerative hierarchical clustering algorithms for use in document retrieval. *Inf. Process. Manage. 22*, 6 (1986), 465–476.
31. VOORHEES, E. M. The efficiency of inverted index and cluster searches. In *Proceedings of the 9th Annual International ACM–SIGIR Conference* (Pisa, Italy, Sept. 1986). ACM, New York, 1986, pp. 164–174.
32. WILLETT, P. Recent trends in hierarchical document clustering: A critical review. *Inf. Process. Manage. 24*, 5 (1988), 577–597.
33. YAO, S. B. Approximating block accesses in database organizations. *Commun. ACM 20*, 4 (Apr. 1977), 260–261.

Received December 1987; revised November 1989; accepted January 1990