# EFFECTIVENESS ASSESSMENT

# OF THE

# COVER COEFFICIENT BASED CLUSTERING METHODOLOGY

by

Fazli Can
Systems Analysis Department
Miami University
Oxford, Ohio 45056

Esen Ozkarahan
School of Business
The Pennsylvania State University
Erie, Pennsylvania 16563

Working Paper #89-002

89/10

# EFFECTIVENESS ASSESSMENT OF THE COVER COEFFICIENT BASED CLUSTERING METHODOLOGY

Fazli CAN*
Esen A. OZKARAHAN+

09/15/89

**Abstract**

An algorithm for document clustering is introduced. The base concept of the algorithm, Cover Coefficient (CC) concept, provides 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 the seeds. It is shown that the complexity of the clustering process is very low. The retrieval experiments show that the IR effectiveness of the algorithm is compatible with a very demanding complete linkage clustering method which is known to have good 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: TODS and INSPEC, the later is a common database with 12684 documents.

Categories and Subject Descriptors: H.3.3. [**Information Storage and Retrieval**]: Information Search and Retrieval - *clustering, search process*

General Terms: Design, Performance, Theory

Additional Key Words and Phrases: cover coefficient, decoupling coefficient, clustering-indexing relationships, cluster validity

* Dept. of Systems Analysis, Miami University, Oxford, OH 45056

+ School of Business, The Pennsylvania State University, Erie, PA 16563

## 1. INTRODUCTION

The phrase "Information Retrieval" represents document retrieval. An Information Retrieval System (IRS) is a system that tries to locate and retrieve documents that are relevant to user queries. Because textual data (e.g., journal and newspaper articles, reports, books, all of which are referred to as "documents" or "text") are stored in a document database, 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, i.e., true or false. In a contemporary information system both types of services are needed. The recent efforts [10, 37, 38, 48] of integrating DBMS and IRS justify such a necessity.

Because document databases are often very large IRSs normally perform retrievals on document representatives. Various models exist for document representation. One document representation model is the vector space model [36]. 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 is a vector and the collection of its elements describe the corresponding document. Each vector element is called an index term, or just term for brevity. The D matrix can be generated manually or by automatic indexing [39, 45, 53]. The individual entries of D, $d_{ij}$ ($1 \le i \le m$, $1 \le j \le n$), indicate the importance of term-j ($t_j$) in document-i ($d_i$), where $t_j$ is a member of the indexing vocabulary T, $T = \{t_1, t_2, ..., t_n\}$. If indexing is binary $d_{ij}$ will be either 0 or 1 indicating the presence or absence of a term in a document. 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 will see later on, however, term weights are normalized to achieve more effective retrieval.

In the essence of vector representation we are modelling a document by a vector in an n-dimensional space defined by n terms. These descriptions (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 [38, 47].

The document representatives are stored in secondary storage, using a structure that facilitates query processing [38, 45, 53, 60]. Query processing retrieves documents that are relevant to the user request. Because exact document (in full text) match rarely occurs in document retrieval relevance of documents is determined by use of a similarity function. In the vector space model the relatedness or similarity of a query to documents, or of two documents, of the database is determined by a similarity function. In document retrieval some similarity functions are also used as a matching function in determining the documents to be retrieved. A well-known similarity or matching function is the cosine function given in Eq. (1.1).

$$\text{Cosine } (X, Y) = \left[ \sum_{i=1}^{n} x_i \cdot y_i \right] \cdot \left[ \sum_{i=1}^{n} x_i^2 \cdot \sum_{i=1}^{n} y_i^2 \right]^{-\frac{1}{2}} \qquad (1.1)$$

In Eq. (1.1) X and Y are vectors of length n, and Cosine (X, Y) is the cosine of the angle formed by the vectors X and Y in the n-dimensional space and therefore takes a value between zero and one. One vector represents the query

and the other the document being compared. Documents that are most similar (with cosine value closer to one) to the user query according to the matching function are presented to the user. In document retrieval, usually documents whose similarity exceed a given threshold (between 0 and 1 if Eq. (1.1) is used) are retrieved by the system. The actual relevance of the document is then decided by the user examining the retrieved documents. Relevance judgment often depends on various factors such as the user's background, document type, depth and style of document, author, etc.

In Information Retrieval (IR) search strategies vary from brute force, that is, full search (FS), which involves comparison of a query vector with the vectors of individual documents, to indexed and cluster based retrievals. (In the IR literature FS is also referred to as best-match [21].) FS is inefficient unless the database is rather small. However, it can be emulated by inverted index searching (IIS). The IIS approach requires a term list for each term. A term list contains document numbers of documents that contain the term as well as the weight of the term in the corresponding document. IIS is as effective as FS and very efficient [12, 47, 55]. The efficiency comes from the fact that only the documents that have at least one common term with the query participate in the similarity calculations.

Efficiency and effectiveness of FS 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 [53], known as the "clustering hypothesis" which states that "closely associated documents tend to be relevant to the same request," and this justifies clustering of documents in a database. There are also disclaimers [54, 55 against this hypothesis. 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 in a top-down or bottom-up search in the hierarchy [12, 55, 59].

CBR facilitates browsing and easy access to complete document information[47, p. 345]. The efficiency of CBR is important. To increase the efficiency of CBR, documents within a cluster can be stored in close proximity within a disk medium to minimize I/O delays [16, 43]. We can also use the IIS approach to first search the centroids and then the member documents of the selected clusters. In addition to being efficient, CBR should be effective in the sense of meeting user needs.

In this paper we will introduce a new clustering methodology based on the Cover Coefficient (CC) concept. We will first present the CC concept along with the various concepts and methodologies that are used with it. In contrast to most other clustering methods, the CC methodology has a formal base and is useful in IR applications.

In the following section an overview of clustering algorithms is presented. The CC concept and methodologies including the CC-based Clustering Methodology ($C^3M$), its complexity analysis, and the indexing-clustering relationships are introduced in Section 3. In the fourth and last section we cover 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 CBR. The experiments



have been carried out using two document databases: the INSPEC database of 12684 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.

## 2. OVERVIEW OF CLUSTERING ALGORITHMS

Clustering or cluster analysis has a widespread use in various disciplines [1]. The vast amount of literature on cluster analysis [1, 17, 23, 27, 49] supports this fact. To see the diversity of the subject the reader may refer to [17] which cites more than three hundred publications (over two hundred and fifty articles from seventy seven journals, forty books, and eighteen reports). Another good source with the same diversity is [30].

At first sight one may think that an expert in a field would be able to judge clustering of the observations at hand by enumerating all possibilities and simply choosing the ones which look the best. Unfortunately, the problem is not that simple. The number of ways to cluster m observations (documents) into $n_c$ nonempty subsets is a Stirling number of second type, given as follows [1, p.3]

$$S_m^{(n_c)} = 1 / n_c! \sum_{k=0}^{n_c} (-1)^{n_c - k} \binom{n_c}{k} k^m$$

For m= 25, and $n_c$ = 5 the number of possibilities is approximately $2.44 \times 10^{13}$. If we do not know the number of clusters $n_c$ the number of possibilities becomes the sum of the Stirling numbers [1]. For m= 25

$$\sum_{j=1}^{25} S_{25}^{(j)} > 4 \times 10^{18}$$

which is a very large number. Indeed, the number of possible clusters cannot be described by an expression having a length bounded by a polynomial function of the input length.

In information systems clustering is used for various purposes. For example, record clustering problem deals with allocation of records to blocks of a direct access file such that the total number of blocks that need to be accessed for a given set of queries is minimized. It is assumed that the set of records that satisfy the queries is known. It is shown that the record clustering problem is NP-complete [36, pp. 161-167; 63]. The clustering problem in IR is different, it tries to group documents to yield a retrieval environment that is both effective and efficient. Clustering in IR is not well defined in the sense that it cannot be defined with a problem statement.

There are three elements of clustering: the document representation matrix, the clustering algorithm, which detects the association among documents, and the resulting set of nonempty clusters.

In judging suitability of a clustering algorithm usually the following questions must be answered:

(a) Are the clusters stable? That is, are they unlikely to change when new documents are added and/or are the clusters unaffected by small errors made in the description of documents [5]?

(b) Is the composition of clusters independent of the order in which documents are processed?

(c) Are the clusters well defined? That is, for a given set of data, does the algorithm produce a single classification or a small number of compatible classifications?

(d) Is document distribution in the clusters as uniform as possible (i.e., are the sizes of clusters close to each other) ?

(e) Is maintenance of the clusters practical and efficient? In other words, is the clustering algorithm able to handle document growth efficiently [8, 13, 52]?

(f) Do the clusters produced result in an effective and efficient retrieval environment?

The answers for all of these questions must, of course, be affirmative for a good clustering algorithm.

Clustering algorithms can be classified in different ways. One possible classification is according to the document distribution in clusters, as in the following:

(a) Partitioning type: Clusters cannot have common documents, i.e., $C_i \cap C_j = \varnothing$ for $1 \le i, j \le n_c$, and $i \ne j$, where $\varnothing$ indicates the null set.

(b) Overlapping type: Clusters can have common documents, i.e., $C_i \cap C_j \ne \varnothing$ for some $1 \le i, j \le n_c$, and $i \ne j$.

(c) Hierarchical type: Clustering structure is represented by a tree where the leaves of the tree represent documents and the interior nodes are non-singleton clusters. (A non-singleton cluster contains more than one member.) The documents of a cluster correspond to leaves of the node corresponding to the cluster. Upper levels of the tree represent clusters of clusters, or clusters of clusters and documents. The root of the tree represents the cluster containing all of the document database [55].

Another classification of clustering algorithms is by the number of iterations needed for clustering as in the following:

(a) Single-pass algorithms: In these algorithms documents are handled only once. The first document will form the first cluster. Then the consecutive documents will be compared with the clusters that have been formed, or more correctly with their centroids. When a document is found close enough to a centroid it is assigned to the corresponding cluster and then the centroid of the cluster is modified accordingly [44].

(b) Iterative algorithms: A typical algorithm starts with the assignment of documents to existing initial clusters or seeds. (There should be a way of creating the seeds.) The centroid vectors of the receiving clusters are then modified. Next, the documents are reassigned to the related clusters iteratively. Each iteration improves the value of an objective function measuring the quality of clustering. The algorithm terminates when an acceptable quality level is reached.

(c) Graph theoretical algorithms: In these algorithms the concepts like "single link", group average link (for short, "average link"), and "maximally complete graph" (complete linkage method) are used. Here, a link means a similarity value greater than or equal to a threshold between two documents. In the single link method, the documents of a cluster are connected to each other by at least one link. In the average link, the number of links of a document to the other documents in the same cluster is greater than or equal to a minimum number. In the complete linkage method (clique) all the documents within a cluster are linked to each other. By changing the



similarity threshold from higher to lower values one may construct a hierarchical organization known as a dendrogram [53].

Typically, the graph theoretical algorithms have a worst time complexity of $O(m^2)$ and space complexity of $O(m)$, for m documents. These algorithms use an inverted file of the document database to avoid unnecessary computations [55, 56]. Even though the single link method is theoretically attractive, its IR performance has usually been found less effective than the other graph theoretical algorithms. In the performance literature [55, 56, 59] the complete linkage method has been found to be the most effective clustering method. Voorhees [55] reports an implementation of the complete linkage method due to C. Buckley. In the worst case, the algorithm requires space and time complexity of, respectively, $O(m^2)$ and $O(m^3)$. Such worst case behavior has been observed in some IR experiments [21].

### 2.1 Partitioning Type Clustering Algorithms

The $C^3M$ clustering algorithm presented in this paper is of the partitioning type. We will therefore analyze partitioning type clustering algorithms in more detail. A partitioning type clustering algorithm generates a partition P, where

$$P = \left\{ C_1, C_2, \ldots, C_{n_c} \right\}, \text{ and } \sum_{i=1}^{n_c} |C_i| = m$$

$|C_i|$ indicates the cardinality of cluster $C_i$. The set P can be formed in different ways. For example, use of a "suitable" threshold value with the single link, average link, or complete linkage graph concept can lead to an acceptable partitioning. However, it is very hard, if not impossible, to estimate the "suitable" similarity threshold value [21, 35].

A generally accepted strategy to generate a partition is to choose a set of documents as the seeds and assign the ordinary (non-seed) documents to the seed documents to form clusters. $C^3M$ uses this strategy.

In the implementation of a seed based approach, there exists a nonempty subset $D_s$ of D called the set of seed documents, and a relation $R_M$ called "member of the same cluster". $R_M$ is an equivalence relation, i.e., it is symmetric, transitive, and reflexive. This relation has the following properties:

(a) No two distinct seeds are members of the same cluster, i.e.,

$$d_i \ne d_j \text{ and } d_i \, R_M \, d_j \text{ ---> } d_i \in (D - D_s) \oplus d_j \in (D - D_s)$$

where, - and $\oplus$ indicate the "set difference" and Boolean "exclusive-or" operator, respectively.

(b) For each $d \in (D - D_s)$, there exists a seed document which is the member of the same cluster.

*Corollary:: For $d_k \in (D - D_s)$, there exists exactly one seed document, $d_l$, satisfying $d_k \, R_M \, d_l$.*

*Proof*: It is stated in (b) that $d_k$ has at least one seed $d_l$ satisfying $d_k \, R_M \, d_l$. Let us consider two seeds $d_l$ and $d_o$. To test the above case, we will have $d_k \, R_M \, d_l$

and $d_k$ $R_M$ $d_o$. Since $R_M$ is an equivalence relation, $d_k$ $R_M$ $d_l$ and $d_k$ $R_M$ $d_o$--> $d_l$ $R_M$ $d_o$. However, this contradicts (a). □

Because seeds determine clusters, the selection or creation of seed documents is an important task. To contrast our seed selection method, that will be presented in Section 3.5, with those of the earlier studies, a non-exhaustive list of known seed selection methods is provided below, where each list item corresponds to a different method:

(a) Select the first $n_c$ documents of a database as cluster seeds, where $n_c$ is the number of clusters to be generated.

(b) Select the seeds randomly from the database [1, 30, 38]

(c) Generate the initial cluster seeds by a random process [1, 38].

(d) Divide the database into partitions, then form the partition centroids as the seeds [1, 38, 41].

(e) For each term, the term generality, i.e., the number of documents containing the term, is calculated. Then for each document the sum of the term generalities of its terms is calculated. The documents with the largest sum are chosen as the seeds [14].

(f) Use an inverted file structure which provides a list of documents per term contained in the documents. Then select the documents related with the term as a cluster. Construct the centroid for each resulting cluster and then use this centroid as the seed. In the above process, terms containing too many or too few documents may be eliminated [43, 58].

(g) Sort the documents in descending order according to their similarity with the database centroid, then choose the documents at the ranks 1, $(1+ m/n_c)$, $(1 + 2m/n_c)$, etc. as the seeds [30].

It should be noticed that the methods c, d, and f do not use documents as seeds.

### 2.2 Unresolved Problems of Clustering Algorithms

The clustering research in IR has produced various methodologies [11, 13-15, 37, 43, 44, 53, 55, 56, 58, 62, 63]. However, as discussed in [22] clustering has yet many unresolved problems. The following list summarizes some of these problems:

(a) Time (i.e., execution time) and space (i.e., memory) complexity of the algorithms is high; time is $O(m^2)$ for m documents in the database.

(b) Graph theoretical based algorithms depend on determination of a similarity threshold to generate cluster links and partitions, this has a time complexity of $O(m^2)$. A suitable threshold is hard to predict and determining it by computing correlations in the database is costly in time [40].

(c) Any attempts of reducing $O(m^2)$ or even $O(m^3)$ time complexity using inverted lists [26, 47] usually works fine [55, 56]. However, the efficiency is not certain. For example, the complete linkage method results in very demanding storage and extended execution times [21]. Furthermore, additional storage requirement is substantial, and it may be even more depending on the similarity function used to calculate similarity between documents [47, pp. 337-338].

(d) Nongraph theoretical based clustering algorithms use concepts that are mostly arbitrary in the way of determining cluster seeds, category vectors (i.e., centroids), or use of inverted file structures. The result is often dependence of clustering on the input order of documents (order dependence) or skewed distribution of documents in resulting clusters [59].



(e) In addition to order dependence and skewed distribution of documents among clusters, it is not possible to predict beforehand the number of clusters to be formed, nor it is possible to establish any relationship between clustering and indexing.

(f) In some algorithms, clusters can be generated independently of the document description matrix D. For example, the relevant documents of a query can be made into a cluster. However, such an approach may lead to clusters containing documents with unsimilar document description vectors. This may be detrimental for document retrieval in a general purpose environment [15, 62].

## 3. CONCEPTS OF $C^3M$

Cover Coefficient, CC, is the base concept of $C^3M$ clustering. The CC concept serves to:

(a) identify relationships among documents of a database by use of the C matrix (or the C' matrix for relationships among terms), - the matrices will be defined shortly;

(b) determine the number of clusters that will result in a document database;

(c) select cluster seeds using a new concept called cluster seed power;

(d) form clusters with respect to $C^3M$, using the concepts of (a) through (c);

(e) correlate the relationships between clustering and indexing;

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

### 3.1 The CC Concept

The CC concept has been introduced for clustering document databases [3, 6, 7]. In this paper, a probabilistic interpretation of this concept will be introduced.

*Definition*: Given is a D matrix representing the document database $\{d_1, d_2, \ldots, d_m\}$ described by the index terms $T= \{t_1, t_2, \ldots, t_n\}$. The cover coefficient matrix, C, is a document by document matrix whose entries $c_{ij}$ $(1 \le i, j \le m)$ indicate the probability of selecting any term of $d_i$ from $d_j$.

An explanation for 'why and how is a term selected' will follow with an example. Briefly, to find the relationship between two documents $d_i$ and $d_j$ we first randomly choose a term of $d_i$ then we try to choose the selected term from $d_j$. Therefore, the answer for 'why' is to find relationships between documents, and the answer for 'how' is random selection of terms in two stages. These answers will be much more clear by the end of this section.

The entries of the D matrix, $d_{ij}$, $(1 \le i \le m, 1 \le j \le n)$ must satisfy the following conditions:

$$\text{(a)} \sum_{j=1}^{n} d_{ij} > 0, \quad 1 \le i \le m \quad \text{(each document has at least one nonzero term)}$$

(b) $\sum_{i=1}^{m} d_{ij} > 0, \quad 1 \le j \le n$ (each term is assigned to at least one document).

We will introduce the CC concept using binary indexing. Subsequently, we will also include weighted indexing.

From the definition of CC, at first glance, it seems that individual entries of the C matrix, $c_{ij}$, would be equal to

$$\frac{\text{(number of terms common to both } d_i \text{ and } d_j)}{\text{(the sum of the document frequencies of terms in } d_i)}$$

However, this is not true. For the calculation of $c_{ij}$ one must first select an arbitrary term (or column) of $d_i$ (say $t_k$) and try to select document $d_j$ from this term, i.e., check if $d_j$ contains $t_k$. In other words, we have a two-stage experiment [28, p. 94] and each row of the C matrix summarizes the results of this two-stage experiment. Let $s_{ik}$ indicate the event of selecting $t_k$ from $d_i$ at the first stage, and $s'_{jk}$ indicate the event of selecting $d_j$ from $t_k$ at the second stage.

An equivalent interpretation is the following: Suppose we have many urns, and each urn contains balls of different colors. Then what is the probability of selecting a ball of particular color? To find this probability notice that first we have to choose an urn at random, then have to choose a ball at random. In terms of D matrix what we have is the following. We have many terms (urns) of $d_i$, choose one of them 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 nothing but 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.

In this experiment, the probability of the simple event "$s_{ik}$ and $s'_{jk}$", i.e., $P(s_{ik}, s'_{jk})$ can be represented as $P(s_{ik}) \times P(s'_{jk})$ [28]. To simplify the notation we will use $s_{ik}$ and $s'_{jk}$, respectively, for $P(s_{ik})$ and $P(s'_{jk})$, where

$$s_{ik} = d_{ik} / \left(\sum_{h=1}^{n} d_{ih}\right), \quad s'_{jk} = d_{jk} / \left(\sum_{h=1}^{m} d_{hk}\right) \quad \text{for } 1 \le i \le m, \ 1 \le k \le n$$

Considering the entire database we can represent the D matrix with respect to the two-stage probability model, as shown in Table 1. Figure 1 shows a hierarchical interpretation of this model. We can consider a path that starts from $d_i$ and ends up at one of the documents, say $d_j$, of the database. In reaching $d_j$ ($1 \le j \le m$) from $d_i$ there are n possible ways ($s_{i1}, s_{i2}, \ldots, s_{in}$). Choose one of them, e.g., $s_{ik}$, 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 $s_{ik} \times s'_{jk}$.



Table 1. Two-stage Probability Model of the D matrix, where $1 \le i \le m$

| 1 | 2 | ... | j | ... | m |
|---|---|---|---|---|---|
| $s_{i1} \times s'_{11}$ | $s_{i1} \times s'_{21}$ | .... | $s_{i1} \times s'_{j1}$ | ... | $s_{i1} \times s'_{m1}$ |
| $s_{i2} \times s'_{12}$ | $s_{i2} \times s'_{22}$ | .... | $s_{i2} \times s'_{j2}$ | ... | $s_{i2} \times s'_{m2}$ |
| ... | ... | | ... | | ... |
| $s_{in} \times s'_{1n}$ | $s_{in} \times s'_{2n}$ | ... | $s_{in} \times s'_{jn}$ | ... | $s_{in} \times s'_{mn}$ |

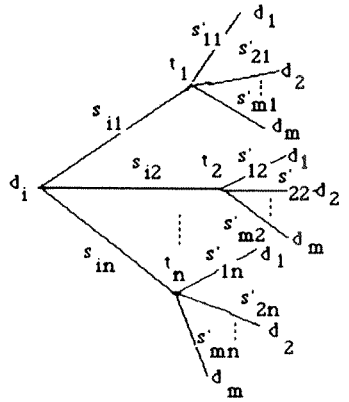


Figure 1. Hierarchical representation of the two-stage probability model for $d_i$ of the D matrix.

By using Table 1 we can find $c_{ij}$ (i.e., the probability of selecting a term of $d_i$ from $d_j$) by summing the probabilities in the jth column of the table.

Let us present an example by using the following D matrix.

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

$$c_{12} = \sum_{k=1}^{6} s_{1k} \times s'_{2k} = 1/3 \times (1/2 + 1/4 + 1/2) = 0.417$$

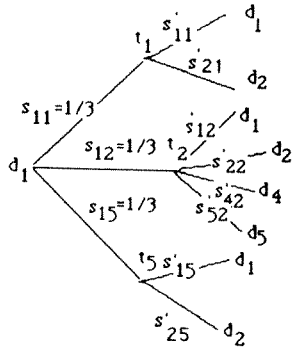


Figure 2. Hierarchical representation of the two-stage probability model for $d_1$ of the example D matrix (non-zero elements only).

$$S = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} \end{bmatrix} \qquad S' = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{4} & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{4} & \frac{1}{2} & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{3} \end{bmatrix}$$

Figure 3. The S and S' matrices for the example D matrix.

To illustrate the concept let us follow the calculation of $c_{12}$. According to the D matrix, $d_1$ contains three terms $\{t_1, t_2, t_5\}$. There are a total of eight



occurrences of these terms with three of this total appearing in $d_2$. An enthusiastic devotee of "equally likely cases" might argue as follows. There are eight occurrences of the terms of $d_1$, any one of which may be drawn; since three of them are in $d_2$, the probability of getting a term of $d_1$ from $d_2$ is 3/8= 0.375 [28, p.92]. As we have stated previously, however, this is not the case. This is because we cannot treat the eight terms as equally likely candidates. According to the two-stage experiment 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 1/2. However, if $t_2$ is selected at the first stage, then the probability of selecting $d_2$ at the second stage is 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 $d_1$, $d_2$, and two other documents. At the first stage, the probability of selecting each element of $\{t_1, t_2, t_5\}$ from $d_1$ is 1/3 and for the rest the probability is 0 (i.e., no term in $\{t_3, t_4, t_6\}$ appears in $d_1$). At this point, for clarity, let us refer back to the urn analogy. $d_1$ has three terms (urns), so the probability of selecting any of them, i.e., one of $\{t_1, t_2, t_5\}$, is 1/3. If we choose term (urn) $t_1$, then the probability of selecting $d_2$ is 1/2, since $t_1$ appears in two documents, or this urn contains two balls.

The pictorial representation of this experiment is provided in Figure 2 (the zero $s_{1j}$ or $s'_{ij}$ values for $1 \le i \le 5$, $1 \le j \le 6$ are not shown in the figure).

The C matrix can be formed from the matrices named S and S', i.e., $C = S \times S'^{T}$ ($S'^{T}$ 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. (In the $s'_{jk}$ case we consider the term definition vector, i.e., the jth column of the D matrix.) For easy reference the S and S' matrices of the example D matrix are provided in Figure 3. Using the definitions of the S and S' matrices, the entries of the C matrix can be defined as follows:

$$c_{ij} = \sum_{k=1}^{n} s_{ik} \times s'^{T}_{kj} = \sum_{k=1}^{n} (\text{probability of selecting } t_k \text{ from } d_i) \times (\text{probability of selecting } d_j \text{ from } t_k) \qquad (3.1)$$

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

$$c_{ij} = \alpha_i \times \sum_{k=1}^{n} d_{ik} \times \beta_k \times d_{jk} \qquad (1 \le i, j \le m) \qquad (3.2)$$

where $\alpha_i$ and $\beta_k$ are the reciprocals of the ith row sum and the kth column sum, respectively, as shown below:

$$\alpha_i = 1 \;/\; \left(\sum_{j=1}^{n} d_{ij}\right) \qquad (1 \le i \le m) \qquad (3.3)$$

$$\beta_k = 1 \;/\; \left(\sum_{j=1}^{m} d_{jk}\right) \qquad (1 \le k \le n) \qquad (3.4)$$

After these definitions it is easy to obtain the $\alpha$ and $\beta$ values. Table 2 provides these values for the example D matrix. Notice the correspondence between the $\alpha$ values and the S matrix, and the $\beta$ values and the S' matrix.

Table 2. The $\alpha$ and $\beta$ values for the example D matrix

$\alpha$ values

| $\alpha_1$ | $\alpha_2$ | $\alpha_3$ | $\alpha_4$ | $\alpha_5$ |
|---|---|---|---|---|
| 1/3 | 1/4 | 1 | 1/3 | 1/4 |

$\beta$ values

| $\beta_1$ | $\beta_2$ | $\beta_3$ | $\beta_4$ | $\beta_5$ | $\beta_6$ |
|---|---|---|---|---|---|
| 1/2 | 1/4 | 1/2 | 1/2 | 1/2 | 1/3 |

### 3.2 Properties of the C Matrix

The following properties hold for the C matrix:

(a) For $i \neq j$ $0 \le c_{ij} \le c_{ii}$ and $c_{ii} > 0$ (i.e., the values of the off-diagonal entries vary between 0 and $c_{ii}$, with $c_{ii}$ always greater than zero).

(b) $c_{i1} + c_{i2} + \ldots + c_{im} = 1$ (i.e., sum of row-i is equal to 1 for $1 \le i \le m$).

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

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

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

The proofs of the properties (a) through (d) are given in [3, 6]. The proof of property (e) follows.

*Theorem-1.* $c_{ii} = c_{jj} = c_{ij} = c_{ji}$ *iff* $d_i$ *and* $d_j$ *are identical.*

*Proof.* If $d_i$ and $d_j$ are identical then $c_{ii} = c_{jj} = c_{ij} = c_{ji}$, this is by definition of the C matrix entries. Now assume $c_{ii} = c_{jj} = c_{ij} = c_{ji}$.



$$\sum_{k=1}^{n} \beta_k \times (d_{ik} - d_{jk})^2 = \sum_{k=1}^{n} \beta_k \times (d_{ik}^2 - 2 \times d_{ik} \times d_{jk} + d_{jk}^2)$$

$$= \sum_{k=1}^{n} \beta_k \times (d_{ik}^2 - d_{ik} \times d_{jk}) + \sum_{k=1}^{n} \beta_k \times (d_{jk}^2 - d_{ik} \times d_{jk}) \qquad (1)$$

For $1 \le j \le m$ $c_{ii} = c_{ij}$ which implies $c_{ii} - c_{ij} = 0$. This can be written as follows:

$$\alpha_i \times \sum_{k=1}^{n} \beta_k \times (d_{ik}^2 - d_{ik} \times d_{jk}) = 0$$

$$\alpha_i > 0 \;\Longrightarrow\; \sum_{k=1}^{n} \beta_k \times (d_{ik}^2 - d_{ik} \times d_{jk}) = 0 \qquad (2)$$

Likewise we have the following.

$$c_{jj} = c_{ji} \;\Longrightarrow\; \sum_{k=1}^{n} \beta_k \times (d_{jk}^2 - d_{ik} \times d_{jk}) = 0 \qquad (3)$$

Substitute (2) and (3) into (1).

$$\sum_{k=1}^{n} \beta_k \times (d_{ik} - d_{jk})^2 = 0$$

Since $\beta_k > 0$ for $1 \le k \le n$, $(d_{ik} - d_{jk}) = 0$ for $1 \le i, j \le m$, therefore $d_{ik} = d_{jk}$ for $1 \le k \le n$; therefore $d_i$ and $d_j$ are identical. □

Here we present the interpretation of these properties.

In property (a), $c_{ij} \ge 0$ means that it may ($c_{ij} > 0$) or may not ($c_{ij} = 0$) be possible to select the terms of $d_i$ from $d_j$. $c_{ij} \le c_{ii}$ is obvious; when one tries to select the terms of $d_i$ from the database, $d_i$ itself will have higher chance 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 (b) 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, i.e., 1.

Property (c) 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 (d) 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 they are not identical, then $c_{ij}$ and $c_{ji}$ will be greater than zero but not necessarily equal to each other.

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

Another property of the C matrix is the following. If a D matrix is mapped into a C matrix, then $\mu$xD, where "$\mu$" is a positive real number defines the same transformation. This comes with the definition of the $c_{ij}$ entries, Eq. (3.2).

To have a better intuition about the meaning of the C matrix consider two document vectors $d_i$ and $d_j$. For these document vectors we can define five possible relationships in terms of the C matrix entries (i.e., in terms of $c_{ii}$, $c_{ij}$, $c_{jj}$, and $c_{ji}$). These are depicted in Figure 4 and listed in the following:

(a) $d_i$ and $d_j$ are identical: i.e.,

$c_{ik} = c_{jk}$, $c_{ki} = c_{kj}$ for $1 \le k \le m$

(b) $d_i$ and $d_j$ have some common terms but neither document "contains" the other:

i.e.,

$c_{ii} > c_{ij}$, $c_{jj} > c_{ji}$, (either $c_{ii} = c_{jj}$ or $c_{ii} \neq c_{jj}$) and (either $c_{ij} = c_{ji}$ or $c_{ij} \neq c_{ji}$)

(c) $d_i$ is a subset of $d_j$: i.e.,

$c_{ii} = c_{ij}$, $c_{jj} > c_{ji}$, $c_{jj} > c_{ii}$, $c_{ij} > c_{ji}$

(d) $d_j$ is a subset of $d_i$ (the reverse of (c)):

$c_{ii} > c_{ij}$, $c_{jj} = c_{ji}$, $c_{jj} < c_{ii}$, $c_{ij} < c_{ji}$

(e) $d_i$ and $d_j$ have no common terms: i.e.,

$c_{ij} = c_{ji} = 0$, if $d_i$ is disjoint with the rest of the database then $c_{ik} = 0$ for $1 \le k \le m$, and $k \neq i$, and $c_{ii} = 1$

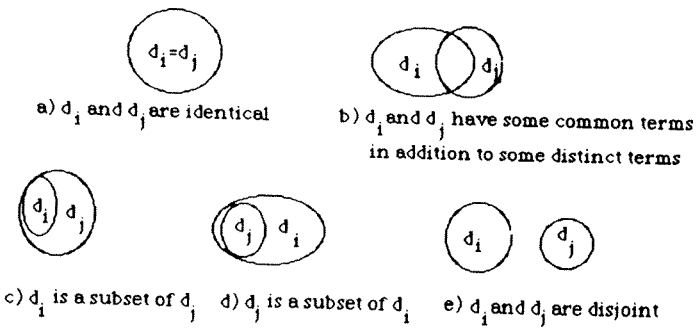


Figure 4. The possible relationships between two document vectors $d_i$ and $d_j$.

These properties say that if a document has common terms with many of the other documents in the database then we start to observe many non-zero values in the off-diagonal entries of the corresponding row; and since the row sum is equal to one, the diagonal entry of the C matrix for the document will have a value comparatively less than one. In the reverse case, i.e., if a document has less common terms with the rest of the database then we will observe many zero values in the off-diagonal entries and relatively higher value (close to 1) at 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 the CC-relationships between two document vectors (refer to Figure 3), $c_{ij}$ can be seen to have the following meaning:

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

Now let us revisit the interpretation of the individual entries of Figure 3.

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

(b) 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.

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

(d) Similar to the discussion in (c).

(e) 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 $d_i$ ($1 \le i \le m$) is relatively more distinct (i.e., if $d_i$ contains fewer terms that are common with other documents), then $c_{ii}$ will take higher values. Because of this, $c_{ii}$ is called the decoupling coefficient, $\delta_i$, of $d_i$. (Notice that $\delta_i$ is a "measure" of how much the document is not related with the other documents and that is why the word "coefficient" is used.) If none of the terms of $d_i$ is contained in any other document, then $\delta_i = 1$ (i.e., $d_i$ is completely decoupled from the other documents in the database). In terms of the two-stage experimentation, it is not possible to select a term of $d_i$ from the other documents of the database.

Contrary to this would mean nonzero $c_{ij}$ which in turn implies nondisjointness of $d_i$ with the other documents.

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, $\psi_i$, of $d_i$. From the properties of the C matrix

$$\psi_i = 1 - \delta_i \quad : \text{coupling coefficient of } d_i \text{ ; and}$$

$$\delta_i = c_{ii} \quad : \text{decoupling coefficient of } d_i$$

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

By using the individual $\delta_i$ and $\psi_i$ values, the overall or average decoupling and coupling coefficients $\delta$ and $\psi$, respectively, of the documents can be defined as

$$\delta = \sum_{i=1}^{m} \delta_i / m \qquad \text{and} \quad 0 < \delta \leq 1 \qquad (3.5)$$

$$\psi = \sum_{i=1}^{m} \psi_i / m = 1 - \delta \qquad \text{and} \quad 0 \leq \psi < 1 \qquad (3.6)$$

The C matrix corresponding to the example D matrix given earlier has the following values, disregarding inexactness due to rounding.

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

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 decoupling coefficient of $d_1$ is $\delta_1 = c_{11} = 0.417$. Accordingly, its coupling with the rest of the database, i.e., coupling coefficient is, $\psi_1 = 1 - \delta_1 = 0.583$. 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 $d_2$, which corresponds to case (c) in Figure 3. As stated in the interpretation of Figure 3 $c_{11} = c_{12}$, $c_{22} > c_{21}$, $c_{22} > c_{11}$, and $c_{12} > c_{21}$ hold, as can be verified with the C matrix.



### 3.3 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 the elements of which are obtained as

$$c'_{ij} = \sum_{k=1}^{m} s'^T_{ik} \times s_{kj} = \sum_{k=1}^{m} (\text{probability of selecting } d_k \text{ from } t_j) \times (\text{probability of selecting } t_j \text{ from } d_k)$$

where $s'^T_{ik} = s'_{ki}$.

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

$$c'_{ij} = \beta_i \times \sum_{k=1}^{m} d_{ki} \times \alpha_k \times d_{kj} \qquad (1 \leq i, j \leq n) \qquad (3.7)$$

The concepts of decoupling and coupling coefficients of $t_j$, $\delta'_j$ and $\psi'_j$, are the counterparts o
f 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 represented by $\delta'$ and $\psi'$, respectively.

The C' matrix resulting from the example D matrix is (ignoring rounding errors)

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

From this C' matrix the decoupling and coupling coefficients of $t_1$ are $\delta'_1 = c'_{11} = 0.292$ and $\psi'_1 = 1 - \delta'_1 = 0.708$. The overall decoupling, $\delta'$, and coupling, $\psi'$, of terms are

$$\delta' = \sum_{j=1}^{6} \delta'_j / 6 = 1.946 / 6 = 0.324$$

$$\psi' = 1 - \delta' = 0.676$$

Notice that in the example D matrix $t_1$ and $t_5$ are identical, i.e., $d_{i1} = d_{i5}$ $(1 \le i \le 5)$. Accordingly, from the properties of the C' matrix $c'_{1j} = c'_{5j}$ and $c'_{j1} = c'_{j5}$ for $1 \le j \le 5$ which can be easily observed in the C' matrix (i.e., the first and fifth rows are identical, the same is valid for the first and fifth columns).

There are some relationships between $\delta'$ and $\delta$, that will be seen at the end of the next section.

### 3.4 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, 18, 22, 30]. In the hierarchical clustering methods, the number of clusters can vary from one (the entire database) to the number of documents (i.e., objects) in the database [1]. As stated previously, this clustering structure is called a dendrogram. A 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 [35]. Unfortunately, stopping rules are hard to apply [21] even for data collections of small sizes [35]. The cluster analysis literature does not provide much help in the IR context. In cluster analysis Euclidian distance measure is generally used as the similarity coefficient to predict and/or validate number of clusters [18, 30]. However, Euclidian distance is not a good choice for document clustering. With Euclidian distance two documents can be regarded very similar to each other even though they do not share any common terms [59].

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, obviously, 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 would be greater than one and less than the size of the database (m) since all documents would not be identical nor would they be completely distinct. The following hypothesis accounts for these facts and/or observations.

*Hypothesis. The number of clusters within a database, $n_c$, 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 it is difficult to predict a similarity threshold (stopping rule) that will yield the desired number of clusters. As stated earlier, stopping rules are hard to find and apply, especially if we consider the size of a document database. This is intuitively obvious since if we do not have the extreme cases (all documents are the same or all are distinct), then similarity provides no additional information about the unknown number of clusters. In an effort to provide a solution to this problem we have been able to successfully use 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:



$$n_c = \sum_{i=1}^{m} \delta_i = \delta \times m \qquad (3.8)$$

The equation (3.8) is consistent with the number of clusters hypothesis. This is because a database with similar documents will have a low $\delta$ (decoupling) and few clusters. On the other hand, a database with dissimilar documents will have a high $\delta$ and many clusters. The previous statement also holds true for terms, with $\delta$ being replaced with $\delta'$. Hence the number of term clusters implied by the C' matrix, $n'_c$, would be

$$n'_c = \sum_{i=1}^{n} \delta'_j = \delta' \times n \qquad (3.9)$$

It is known that classifying documents implies classifying terms and vice versa [50]. This is intuitively evident 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 [13] or vice versa [31] 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 been proven [3, 6] to hold.

Let us show the agreement between Eq.s (3.8) and (3.9) and the number of clusters hypothesis with the use of the following theorems and corollaries:

*Theorem-2:* $n_c = 1$ *iff all documents of D are identical.*

*Part-1.* If all documents of D are identical then $n_c = 1$.

*Proof:* From the properties of the D matrix $d_{ik} > 0$ for $1 \le i \le m$, $1 \le k \le n$. This is because a term exists if and only if it appears in at least one document. Since all documents are identical $d_{ik} = d_{jk}$ for $1 \le i, j \le m$ and $1 \le k \le n$. For simplicity let us use $d_k$ for $d_{ik}$ and $d_{jk}$, $1 \le k \le n$. Then from Eq. (3.2)

$$\delta_i = \alpha_i \times \sum_{k=1}^{n} d_k^2 \times \beta_k \qquad \text{for } 1 \le i \le m$$

where $\beta_k = 1 / (m \times d_k)$, hence $d_k^2 \times \beta_k = d_k / m$. Accordingly,

$$\sum_{k=1}^{n} d_k^2 \times \beta_k = (1/m) \times (d_1 + d_2 + \ldots + d_n)$$

On the other hand $\alpha_i = 1 / (d_1 + d_2 + \ldots + d_n)$. Hence $\delta_i = 1 / m$ for $1 \le i \le m$. This implies that $n_c$, which is the summation of all $\delta_i$, is equal to $m \times 1/m = 1$.

*Part-2.* If $n_c = 1$ then all documents of D are identical .

*Proof:* $n_c = 1$ implies

$$\sum_{i=1}^{m} c_{ii} = 1 \implies \sum_{i=1}^{m} \alpha_i \times \sum_{k=1}^{n} \beta_k \times d_{ik}^2 = 1 \qquad (1)$$

$$\sum_{i,j=1,\, j\neq i}^{m} c_{ij} = (m-1) \implies \sum_{i=1}^{m} \alpha_i \times \sum_{j=1,\, i\neq j}^{m} \sum_{k=1}^{n} \beta_k \times d_{ik} \times d_{jk} = (m-1) \qquad (2)$$

Multiply both sides of (1) by (m-1).

$$\sum_{i=1}^{m} \alpha_i \times (m-1) \times \sum_{k=1}^{n} \beta_k \times d_{ik}^2 = \sum_{i=1}^{m} \alpha_i \sum_{j=1,\, j\neq i}^{m} \sum_{k=1}^{n} \beta_k \times d_{ik}^2 = (m-1) \qquad (3)$$

Subtracting (2) from (3)

$$\sum_{i=1}^{m} \alpha_i \sum_{k=1}^{n} \beta_k \times \left[ \sum_{j=1,\, j\neq i}^{m} (d_{ik}^2 - d_{ik} \times d_{jk}) \right] = 0$$

$$\sum_{i=1}^{m} \alpha_i \sum_{k=1}^{n} \beta_k \times \left[ \sum_{j=2,\, j>i}^{m} (d_{ik}^2 - 2 \times d_{ik} \times d_{jk} + d_{jk}^2) \right] = 0$$

$$\sum_{i,j=1,\, j>i}^{m} \alpha_i \sum_{k=1}^{n} \beta_k \times (d_{ik} - d_{jk})^2 = 0$$

Therefore $d_{ik} = d_{jk}$ for $1 \le i, j \le m$ and $1 \le k \le n$, therefore all documents are identical. □

*Theorem-3: For a binary D matrix the minimum value of $c_{ii}$ (i.e., $\delta_i$) 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 $c_{i1} + c_{i2} + \ldots + c_{im} < 1$ which contradicts the property (b) of the C matrix, i.e., $c_{i1} + c_{i2} + \ldots + c_{im} = 1$. Hence, the minimum value of $c_{ii} = 1 / m$. □

*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 $c_{ii} > c_{ij}$ or $c_{jj} > c_{ji}$, due to property (a). Assume that $c_{ii} > c_{ij}$. 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 (b) of the C matrix. That is, if $c_{ij} = 1/m$, then none of the off diagonal entries can assume a value greater than 1/m.) Further, 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 $1 \le n_c \le min(m, n)$.*

*Proof:* We know that $n_c = n'_c$, and $n_c$ and $n'_c$ are the summation of the diagonal entries of C and C' matrices, respectively. C and C' are square matrices of sizes m x m and n x 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 $\max(n_c) = \min(m, n)$. □

Notice that $n_c \le \min(m, n)$ is logical, since $n_c \ge \min(m, n)$ would violate the partitioning property of $C^3M$ that will be introduced later. For instance if $n_c > m$ then some document must be the member of more than one cluster, which contradicts the definition of partitioning, i.e., no overlap.

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 increase in $\delta$ or in individual $\delta_i$ ( $1 \le i \le m$) 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 relationship between $d_c$ and $\delta$ is shown on a logarithmic scale in Figure 5. The value range of $d_c$ is $\max(1, m/n) \le d_c \le m$.

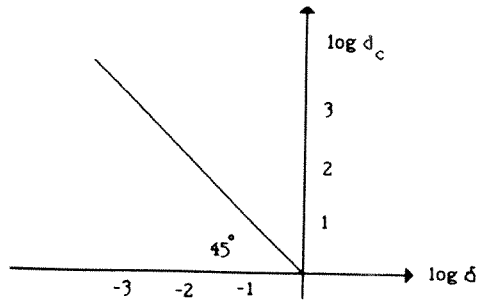


Figure 5 The relationship between $d_c$ and $\delta$ on a logarithmic scale.
$\log d_c = -\log \delta \quad (0 < \delta \le 1)$.

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

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

$$n_c = \delta \times m = 0.389 \times 5 = 1.945 \approx 2$$

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).

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 $c_{ii} < c_{ij}$, which is very easy to prove. Consider $c_{ii}$ and $c_{ij}$:

$$c_{ii} = \alpha_i \times \sum_{k=1}^{n} d_{ik}^2 \times \beta_k \qquad c_{ij} = \alpha_i \times \sum_{k=1}^{n} d_{ik} \times d_{jk} \times \beta_k$$

If $\min(d_{jk}) = d_{ik}$ for $1 \le j \le m$ and if $d_{jk} > d_{ik}$ for at least one k ($1 \le k \le n$) then $c_{ij} > c_{ii}$. Intuitively $c_{ij} > c_{ii}$ implies $c_{jj} < c_{ji}$, and in the following it is proven that this intuitive inequality holds.

*Theorem-4. In the C matrix corresponding to a weighted D matrix $c_{ij} > c_{ii}$ implies $c_{jj} > c_{ji}$.*

*Proof.* Assume $c_{ij} > c_{ii}$ then $c_{ij} - c_{ii} > 0$.

$$c_{ij} - c_{ii} = \alpha_i \sum_{k=1}^{n} \beta_k \times d_{ik} \times d_{jk} - \alpha_i \sum_{k=1}^{n} \beta_k \times d_{ik}^2$$

$$c_{ij} - c_{ii} = \alpha_i \times \left[ \sum_{k=1}^{n} \beta_k \times d_{ik} \times (d_{jk} - d_{ik}) \right] > 0$$

$$c_{jj} - c_{ji} = \alpha_j \times \left[ \sum_{k=1}^{n} \beta_k \times (d_{jk}^2 - d_{jk} \times d_{ik}) \right]$$

$$= \alpha_j \times \left[ \sum_{k=1}^{n} \beta_k \times (d_{jk}^2 - 2 \times d_{jk} \times d_{ik} + d_{ik}^2 + d_{jk} \times d_{ik} - d_{ik}^2) \right]$$

$$= \alpha_j \times \left[ \sum_{k=1}^{n} \beta_k \times (d_{jk} - d_{ik})^2 + \sum_{k=1}^{n} \beta_k \times d_{ik} \times (d_{jk} - d_{ik}) \right]$$

But $\alpha_j > 0$,



$$\sum_{k=1}^{n} \beta_k \times (d_{jk} - d_{ik})^2 \ge 0 \qquad \text{(since } \beta_k > 0\text{), and}$$

$$\sum_{k=1}^{n} \beta_k \times d_{ik} \times (d_{jk} - d_{ik}) > 0$$

□

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

*Theorem-5. In the C matrix corresponding to a weighted D matrix the minimum value of $n_c$ is 1.*

*Proof.* The proof is by contradiction.

$$\sum_{i=1}^{m} c_{ii} = \sum_{i=1}^{m} \alpha_i \sum_{k=1}^{n} \beta_k \times d_{ik}^2 < 1$$

Multiply both sides of the above inequality by (m-1), then it can be rewritten as follows.

$$\sum_{i=1}^{m} \alpha_i \sum_{j=1,\, j \ne i}^{m} \sum_{k=1}^{n} \beta_k \times d_{ik}^2 < (m - 1) \qquad (1)$$

$$\sum_{i=1}^{m} \alpha_i \sum_{j=1,\, j \ne i}^{m} \sum_{k=1}^{n} \beta_k \times d_{ik} \times d_{jk} > (m - 1) \qquad (2)$$

Subtract inequality (2) from inequality (1).

$$\sum_{i=1}^{m} \alpha_i \sum_{j=1, j \ne i}^{m} \sum_{k=1}^{n} \beta_k \times d_{ik}^2 - \sum_{i=1}^{m} \alpha_i \sum_{j=1, j \ne i}^{m} \sum_{k=1}^{n} \beta_k \times d_{ik} \times d_{jk} < 0$$

$$\sum_{i=1}^{m} \alpha_i \sum_{k=1}^{n} \beta_k \times \left[ \sum_{j=1, j \ne i}^{m} (d_{ik}^2 - d_{ik} \times d_{jk}) \right] < 0$$

$$\sum_{i=1}^{m} \alpha_i \sum_{k=1}^{n} \beta_k \sum_{j=2, j>i}^{m} (d_{ik}^2 - 2 \times d_{ik} \times d_{jk} + d_{jk}^2) < 0$$

$$\sum_{i,i=1, i>i}^{m} \alpha_i \sum_{k=1}^{n} \beta_k \times (d_{ik} - d_{jk})^2 < 0$$

$\alpha_i > 0, \beta_k > 0$ and $(d_{ik} - d_{jk})^2 \ge 0$, this implies

$$\sum_{i,j=1, j>i}^{m} \alpha_i \sum_{k=1}^{n} \beta_k \times (d_{ik} - d_{jk})^2 \geq 0$$

This is a contradiction. □

In passing it should be noted that all of the properties of the C matrix are exactly valid for the C' matrix. The necessary complementary proofs follow similar steps as those in the proofs introduced so far. This is because the Eq.s (3.2) and (3.7) are conceptually the same due to conceptual equality of αs and βs. For easy reference properties of the C matrix are summarized in Table 3.

Table 3. The properties of C matrix

| Propery ($1 \leq i, j \leq m, i \neq j$) | D Matrix | |
|---|---|---|
| | Binary | Weighted |
| $0 < c_{ij} \leq 1,\ 0 \leq c_{ij} < 1$ | + | + |
| $c_{ij} \leq c_{ii}$ | + | (1) |
| $c_{ij} > c_{ii}$ --> $c_{ij} > c_{ji}$ | (2) | + |
| $c_{i1} + c_{i2} + \ldots + c_{im} = 1$ | + | + |
| $d_i$ is distinct <--> $c_{ii} = 1$ | + | + |
| $c_{ij} = 0$ <--> $c_{ji} = 0$, $c_{ij} > 0$ <--> $c_{ji} > 0$ | + | + |
| min $(c_{ii}) = 1/m$ | + | (3) |
| $1 \leq n_c \leq \min(m, n)$ | + | + |
| $c_{ii} = c_{ij} = c_{ji} = c_{jj}$ <--> $d_i$ and $d_j$ are identical | + | + |
| $n_c = 1$ <--> all documents are identical | + | + |
| $d_i$, $d_j$ are identical --> $c_{ik} = c_{jk}$, $c_{ki} = c_{kj}$ $(1 \leq k \leq n)$ | + | + |
| D and uxD ($u > 0$) implies the same C matrix | + | + |

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

For added interest we can state the relationship between δ and δ'. From $n_c = n'_c$ we have

$$\delta = (n / m) \times \delta' \quad \text{and} \quad \delta' = (m / n) \times \delta$$

These relationships lead to the following conclusions. If m and n are equal, then average decoupling of documents (δ) will be equal to average decoupling of terms (δ'). In such an environment, an average document and an average term are defined with the same average number of terms and documents, respectively. Accordingly, expected average similarity of documents and terms would almost be the same and hence it is natural to have identical values for δ and δ'. On the other hand, $m > n$ implies $\delta < \delta'$. This means that the average number of common items (terms) among documents is greater than the average number of common items (documents) among terms. More common terms means more similarity, or more coupling, and hence less decoupling. The case of $n < m$ means the reverse. In a real life environment, one would expect to observe $\delta < \delta'$, since m would be greater than n.



### 3.5 Cluster Seed Power

The $C^3M$ is seed oriented, i.e., $n_c$ documents are selected as cluster seeds and non-seed documents are grouped around the seeds to form clusters. The seed selection process depends on the cluster seed power, $p_i$, of $d_i$ $(1 \leq i \leq m)$ which is a concept introduced in $C^3M$. The cluster seed power of $d_i$ is defined as

$$p_i = \delta_i \times \psi_i \times \sum_{j=1}^{n} d_{ij} \quad \text{and} \tag{3.10}$$

$$p_i = \delta_i \times \psi_i \times \sum_{j=1}^{n} (d_{ij} \times \delta'_j \times \psi'_j) \tag{3.11}$$

The Eqs. (3.10) and (3.11) pertain to a binary and weighted D matrix, respectively. In these equations $\delta_i$ provides the separation of clusters (inter cluster dispersion), $\psi_i$ provides the connection among the documents within a cluster (intra cluster cohesion) and the third term (i.e., $d_{i1} + d_{i2} + \ldots + d_{in}$, or $d_{i1} \times \delta'_1 \times \psi'_1 + d_{i2} \times \delta'_2 \times \psi'_2 + \ldots + d_{in} \times \delta'_n \times \psi'_n$) provides normalization. In a weighted D matrix, the normalization factor (i.e., $d_{i1} + d_{i2} + \ldots + d_{in}$, which is the number of terms used for the description of documents) should be normalized further, and this is provided by the product $\delta'_j \times \psi'_j$ for individual terms, since there might be some terms with high weights in $d_i$. With such an approach a term with a high weight, but with a skewed frequency (i.e., a very frequent or a very rare term) will not contribute too much to the summation, i.e., the seed power of $d_i$.

It should be noticed that some seeds might be almost identical, since they may be described by nearly identical sets of terms. To eliminate the similar (false) seeds the following algorithm is provided.

**An algorithm to eliminate identical (false) cluster seeds:**

[a] *Calculate cluster seed power of all documents and sort documents in descending order according to their seed power*
$N_c = 0$; /* $N_c$ *indicates the number of equivalence classes in the seed document set; in this algorithm we want to use only one of the identical documents in each class as a seed* */

[b] ***repeat;***
**if** $N_c < n_c$ **then**
**do;**
*Consider the next* $(n_c - N_c)$ *documents with the maximum cluster seed power as the new cluster seeds;*
**end;**
*Determine the number of equivalence classes within this cluster seed collection, and set* $N_c$ *to this number;*
**until** $N_c = n_c$ ;

The equivalence classes within the set of candidate cluster seeds are found by using the relation "equivalent seed", $R_e$. Two seeds, $d_i$ and $d_j$, are related to each other with respect to the relation $R_e$, if $c_{ii} = c_{jj}$, $c_{ii} = c_{ij}$, $c_{jj} = c_{ji}$. It is obvious that the relation $R_e$ holds the requirements of an equivalence relation since it is reflexive, symmetric, and transitive. This relation is illustrated in the following:

(a) $R_e$ is reflexive, i.e., $d_i$ $R_e$ $d_i$ is trivial.

(b) $R_e$ is symmetric by definition (its definition does not consider ordering).

(c) $R_e$ is transitive, since $d_i$ $R_e$ $d_j$ and $d_j$ $R_e$ $d_k$ imply that $d_i$ $R_e$ $d_k$. $d_i$ $R_e$ $d_j$ implies $c_{ii} = c_{jj}$, $c_{ii} = c_{ij}$, $c_{jj} = c_{ji}$; $d_j$ $R_e$ $d_k$ implies $c_{jj} = c_{kk}$, $c_{jj} = c_{jk}$, $c_{kk} = c_{kj}$. Together these equalities imply that $c_{ii} = c_{kk}$, $c_{ii} = c_{ik}$, $c_{kk} = c_{ki}$. Therefore, $R_e$ is transitive. (Also recall that the equalities $c_{ii} = c_{jj}$, $c_{ii} = c_{ij}$, $c_{jj} = c_{ji}$ imply $d_i$ and $d_j$ are identical.)

After showing that $R_e$ is an equivalence relation, it is then trivial to show that $R_e$ partitions the cluster seeds into equivalence classes [2, pp.87-88]. It is obvious that within an equivalence class there might be two or more (identical) cluster seeds. Only one of the seeds of an equivalence class is taken as a cluster seed, and the rest are considered false, since all are compatible (or equivalent) with the one chosen as the cluster seed.

The above algorithm might be applied as follows. First we have to determine the documents with almost the same (use a threshold value) seed power value. To do this we have to sort the documents according to their seed power value. 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 we compare the candidate seeds with each other. If two candidate seeds of the same group, $d_i$ and $d_j$, are the members of an equivalence class, then the entries $c_{ii}$, $c_{jj}$, $c_{ij}$, and $c_{ji}$ will be almost identical, i.e., the absolute value of $(c_{ii} - c_{jj}) < \varepsilon$, $(c_{ii} - c_{ij}) < \varepsilon$, and $(c_{jj} - c_{ji}) < \varepsilon$ where $\varepsilon$ is the threshold. In our experiments $\varepsilon$ was the real constant of 0.001 and we never had identical seeds, i.e., the first $n_c$ documents of the sorted list did not contain any identical seeds.

In our experiments [32] it has been observed that documents with medium number of terms tend to have higher seed power. That is to say, special documents that are defined by very few terms or general documents that are defined by too many terms are less likely to be selected as cluster seeds. This is what makes sense for a seed selection methodology since general or special documents would not be appropriate for a cluster seed. General documents do not provide inter-cluster 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. (3.10) and listed in decreasing order in the following:

$p_2 = 0.438 \times (1 - 0.438) \times 4 = 0.985$

$p_5 = 0.396 \times (1 - 0.396) \times 4 = 0.957$

$p_1 = 0.417 \times (1 - 0.417) \times 3 = 0.729$

$p_4 = 0.361 \times (1 - 0.361) \times 3 = 0.692$



$p_3 = 0.333 \times (1 - 0.333) \times 1 = 0.222$

Since $n_c = 2$, then $d_2$ and $d_5$ become candidate seed documents. Our false seed elimination algorithm determines that they are distinct (notice that the values $c_{22} = 0.292$ and $c_{55} = 0.528$ are significantly different). Hence $d_2$ and $d_5$ are selected as the cluster seeds. Notice that in this example there is no need to check $c_{25}$ and $c_{52}$; we can decide on the basis of only $c_{22}$ and $c_{55}$ since they are significantly different.

### 3.6 The $C^3M$ Algorithm

$C^3M$ is a partitioning type clustering algorithm which operates in a single-pass. A brief description of the algorithm is as follows [3, 6].

**$C^3M$:**

[a] *Determine the cluster seeds of the database.*

[b] i= *1;*

***repeat;*** */* construction of clusters */*

***if*** *$d_i$ is not a cluster seed*

***then***

***do;***

*Find the cluster seed (if any) which 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 remains unclustered documents group them into a rag-bag cluster (some non-seed documents may not have any covering seed document).*

Notice that in the above algorithm we try to concentrate non-seed documents around seed documents. Some non-seed documents may not find any seed document to join, such documents are inserted into a rag-bag cluster in step (c) of the algorithm. The upper bound for the size of rag-bag cluster is $(m - n_c)$. The better selection of seeds implies lower size for the rag-bag cluster. In our experiments we have never ended up with a rag-bag cluster; the case of INSPEC database for $D_{17}$ matrix (see Section 4.2) is the sole exception.

A multi-pass version of the $C^3M$ has been introduced and compared with the single-pass version. In the multi-pass version, cluster seeds are selected in the same way; however, in the assignment of documents to cluster seeds a similarity coefficient is used. This is because after the first pass cluster seeds are replaced by cluster centroids. This assignment is performed repetitively until cluster definitions reach stability.

Numerous experiments showed that the computationally efficient single-pass version of the algorithm generates clustering structures (partitions) compatible/similar with those of the multi-pass version. The compatibility of two partitions is measured by looking at the joint membership of pairs of documents in the two partitions. There are different approaches of measuring

compatibility [1, 30] and they yield a value between 0 (when one partition contains 1 cluster and the other contains m clusters) and 1 (when both partitions are identical). The compatibility of the generated partitions is valid in both binary [4, 6] and weighted [32] D matrices. In the multi-pass version, we have used conventional similarity coefficients (Dice and cosine) to assign documents to seeds/centroids. This was to show that the CC concept generates partitions that are compatible with those of computationally more expensive and more traditional methods that are based on conventional similarity measures.

**Implementation of the Algorithm**

The $C^3M$ algorithm can be implemented in the following ways. To implement Step (b) of the algorithm we can
I1: Implement the algorithm directly, or
I2: Use an inverted index for the terms that appear in cluster seeds..
The inverted index of a term includes the seed documents that contain the term and the weight of the term in that seed [41]. 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).

In I1, for a non-seed document $d_i$, $c_{ij}$ (where $d_j$ is a seed) values for each seed document are computed one after the other. In I2, $c_{ij}$ values are concurrently computed for those cluster seeds that have terms common with the non-seed document.

The complexity analysis of I1 and I2 is as follows:

I1: $O( (m - n_c) \times x_d \times n_c )$

I2: $O( (m - n_c) \times x_d \times t_{gs} + n_c \times x_d ) \text{ ---> } O( (m - n_c) \times x_d \times t_{gs} )$

In the above expressions $x_d$ is the average number of 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 non-seed documents. (The meaning of $t_{gs}$ will become much clear in the following paragraphs.)

Let us 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 approximately for each term of a non-seed document there is a list of length $t_{gs}$. Here we are ignoring the existence of terms with term generality one in non-seed 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, $(m-n) \times x_d \times t_{gs}$.

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

$$(m-n_c) \times (x_d \times s / n_2) \times (t_s / s), \text{ or}$$



$$(m-n_c) \times x_d \times t_s / n_2$$

In this formula s is the number of distinct terms used by 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 (i.e., no false seed documents were observed in the experiments), we can ignore the fact that some select terms would appear only in seed documents. Accordingly, the probability of occurrence of a non-seed document term in a seed document is roughly equal to $s/n_2$. Notice that non-seed document terms with term generality of minimum two can appear in seed documents and $n_2$ indicates the number of such terms in indexing vocabulary. (If indexing vocabulary contains terms with term generality greater than one then $n_2$ is equal to n.) This means that for each document to be clustered, a $s/n_2$ 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" (recall that only the terms with the minimum term generality of two are considered).

It is easy to show that the computational cost of I2 is less than I1, although it involves the construction of an inverted index for terms that appear in seed documents. To see this we have to show the validity of the following inequality:

$(m - n_c) \times x_d \times t_{gs} + n_c \times x_d < (m - n_c) \times x_d \times n_c$, which becomes

$t_{gs} \times (m - n_c) - n_c \times (m - 1 - n_c) < 0$, where $m >> 1$, hence we have

$(t_{gs} - n_c) \times (m - n_c) < 0$

where $\max(t_{gs}) = n_c$ and $\max(n_c) = m$; therefore, in a typical case the above inequality will be satisfied.

The computational cost of I2 will be a fraction of I1, which is $t_{gs}/n_c$. Our experiments showed that the typical cost of I2 is approximately five percent and one percent of I1, respectively, for the document databases TODS214 and INSPEC.

Consider the construction of clusters for the example D matrix. We have determined earlier that $D_o = \{d_1, d_3, d_4\}$ is the set of documents to be clustered and $D_s = \{d_2, d_5\}$ is the set of seed documents. To construct the clusters we need only to calculate $c_{ij}$s where $d_i \in D_o$ and $d_j \in D_s$ . In this calculation we consider only the terms common between non-seed and seed documents. For example, for the non-seed 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 $C_2 = \{d_3, d_4, d_5\}$.

### Characteristics of the $C^3M$ Algorithm

$C^3M$ satisfies the desirable characteristics of good clustering algorithms in the following ways:

(a) It has been experimentally shown [5, 6, 32] 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. For example, the average link method, which has shown good IR performance in some experiments [24] has instability, i.e., small changes in the similarities between documents may result in substantially different hierarchies [55, p. 28].

(b) The algorithm is independent of the order of documents. This is because coupling between two documents is not affected by the position of documents in the D matrix. Therefore, the $C^3M$ algorithm generates a well defined clustering pattern, i.e, it produces a unique classification. The hierarchical clustering methods of complete linkage and average link, for example, do not necessarily define a unique dendrogram. This is because during the construction of clusters similarity ties are broken arbitrarily [55, p. 27].

(c) The algorithm requires a very small memory space for the data structures needed in the implementation. $\alpha$s and $\beta$s require m and n memory locations, respectively. m memory locations are also needed to keep the diagonal entries of the C matrix and to calculate seed powers. It should be noted that we do not need to keep the entire C matrix in memory, except the diagonal elements, since any element of it can be obtained by computation. The memory space allocated to the diagonal entries can also be used to store document seed powers since they are calculated sequentially. In the case of weighted indexing we need to calculate the diagonal entries of the C' matrix, requiring n memory entries, to compute seed powers (see Eq. (3.11)). After determining cluster seeds all we need are $\alpha$s and $\beta$s, i.e., we can free the memory locations used for the other data structures. Assignment of documents is performed efficiently by computing CC values between non-seed documents and the seed documents. The memory locations needed for the inverted index of the select terms is proportional to $n_2 x t_{gs}$ which is very small. For example, for the description of the INSPEC database our $n_2$ and $t_{gs}$ values are 7435 and 3.10, respectively. 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 hence results in a time complexity of $O(m^2)$ [59].

(d) It has been experimentally observed that the 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.

The foregoing discussion shows that the expected complexity of $C^3M$ is $O(m \times x_d \times t_{gs})$ since the complexity of the other steps of the algorithm can be



ignored with respect to the complexity of Step (b) [3]. Obviously, this is better than most other clustering algorithms whose complexity range in $O(m^2)$ or $O(m^3)$ [43, 56, 59]. In Section 4.3 $C^3M$'s complexity will be experimentally verified. It should also be noted that in complexity comparisons the desirable features of a clustering algorithm should also come in play. That is, one would assess whether an order dependent, unstable, skewed, theoretically unpredictable yet a bit cheaper algorithm is all one needs.

### 3.7 Indexing-Clustering Relationships Obtainable from the CC Concept

The CC concept indicates some relationships between indexing and clustering. In this section we show analytical derivation of these relationships using binary indexing. In Section 4.2 these relationships are experimentally observed using both binary and weighted indexing.

For the derivation of the indexing-clustering relationships let us consider Eqs. (3.2) and (3.8) and find $n_c$:

$$n_c = \sum_{i=1}^{m} \delta_i = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij}^2 \times \alpha_i \times \beta_j \qquad (3.12)$$

In the case of binary indexing $d^2_{ij} = d_{ij}$. By substituting the values of $\alpha_i$ (Eq. (3.3)) and $\beta_j$ (Eq. (3.4)) in Eq. (3.12) we obtain the following:

$$n_c = \sum_{i=1}^{m} \sum_{j=1}^{n} \left[ d_{ij} \times \left[ \sum_{k=1}^{n} d_{ik} \right]^{-1} \times \left[ \sum_{k=1}^{m} d_{kj} \right]^{-1} \right] \qquad (3.13)$$

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 $t_j$ [34]. With these definitions Eq. (3.13) becomes

$$n_c = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} \times \left[ x_{d_i} \times t_{g_j} \right]^{-1} \qquad (3.14)$$

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

$$x_d = \sum_{i=1}^{m} x_{d_i} / m, \quad t_g = \sum_{j=1}^{n} t_{g_j} / n \qquad (3.15)$$

We can then roughly approximate $x_{d_i} \times t_{g_j}$ with $x_d \times t_g$ and rewrite (3.14) as follows:

$$n_c = \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} \times \left[x_d \times t_g\right]^{-1} = t \times \left[x_d \times t_g\right]^{-1} \qquad (3.16)$$

Eq. (3.16) indicates the relationships among the quantities of: number of clusters, $n_c$, total number of term assignments, t, average depth of indexing, $x_d$, and average term generality, $t_g$.

In Eq. (3.16) if we substitute t/n for $t_g$ and t/m for $x_d$ then $n_c$ could be written in the following way:

$$n_c = (m \times n) / t = m / t_g = n / x_d \qquad (3.17)$$

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:

$$d_c = m / n_c = 1 / \delta = m / (m / t_g) = t_g \qquad (3.18)$$

$$d'_c = n / n'_c = 1 / \delta' = n / (n / x_d) = x_d \qquad (3.19)$$

Equations (3.18) and (3.19) show that $t_g$ and $x_d$ are the basic determinants of document cluster and term cluster sizes, respectively. In other words, they determine the policy of indexing.

The value range of $n_c$, indicated by the indexing-clustering relationships, is consistent with the theoretical expectation, i.e., $1 \le n_c \le \min(m, n)$ (see Section 3.4). To show this consider Eq. (3.17), i.e., $n_c = (m \times n) / t$ along with the possible max(t) and min(t) values. Obviously, max(t) is equal to (m x n), which can be observed only if all documents of the database are identical and contain all terms. On the other hand, min(t) is equal to max(m, n) relationship holds if each term is assigned to only one document and the document is described by more than one term (i.e., n > m: see Figure 5.a), or each term is assigned to more than one document and all documents are described by only one term (i.e., m > n: see Figure 5.b). Therefore, the minimum value of $n_c$ will be obtained if t is maximum. Hence:

$$\min\ (n_c) = (m \times n) / \max(t) = 1$$

Similarly, the maximum value of $n_c$ will be observed if t is minimum, i.e.,

$$\max(n_c) = \left[(m \times n) / \min(t)\right] = \left[(m \times n) / \max(m, n)\right] = \min(m, n)$$

$$D_1 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad D_2 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

a) n > m: 3 > 2; t= n= 3 b) m > n: 3 > 2; t= m= 3

Figure 5. Example D matrices for observing of min(t)= max(m, n)

For example, if we consider the $D_1$ and $D_2$ matrices shown in Figure 5 then Eqs. (3.8) and (3.9) indicate that $n_c = n'_c = 2$. (In Figure 5.a the documents are unique, on the other hand in Figure 5.b the terms are unique.) For $D_1$ the document clusters are $\{d_1\}$ and $\{d_2\}$, and the term clusters are $\{t_1, t_2\}$ and $\{t_3\}$.



For $D_2$ the document and term clusters are $\{d_1, d_2\}$, $\{d_3\}$, and $\{t_1\}$, $\{t_2\}$ respectively. The indexing-clustering relationships also indicate the same value for $n_c$ since t= max(m, n)= max(3, 2)= 3, and $n_c$= (m x n)/ t: (3 x 2) / 3= 2= min(m, n).

If we apply Eq. (3.16) to the example D matrix of Section 3.1 we obtain

$$n_c = 15 / (3 \times 2) = 2$$

Similarly, by using the expression in Eq. (3.17)

$$n_c = (5 \times 6) / 15 = 5 / 2.5 = 2$$

In other words, under the CC, the number of clusters depicted by the indexing-clustering relationships is very close to the theoretically expected value, i.e., 1.95 (refer to Section 3.4). The same is also valid for Eqs. (3.18) and (3.19).

So far we have derived the indexing-clustering relationships (Eqs. (3.16) through (3.19)) indicated by the CC concept by using a binary D matrix. However, as will be shown in Section 4.2, these relationships are also valid for weighted indexing, with the exception of a little distortion introduced by term weights.

It goes without saying that the indexing-clustering relationships are very valuable for practical purposes. This is because

1. The results of the algorithm are predictable.
2. The indexing-clustering relationships can be used to estimate the storage requirement of the clustering algorithm.
3. The relationship $n_c = (m \times n) / t$ can be used to check the clustering tendency of a document collection. Testing of clustering tendency in IR measures the extent to which the use of a clustering method on a document database is likely to lead to increase in retrieval effectiveness with respect to FS [19, p. 361; 59].

## 4. EXPERIMENTAL DESIGN AND EVALUATION

In this section we present three sets of experiments:

(a) Validity experiments to test the validity of indexing-clustering relationships.

(b) Usability experiments to verify time efficiency of $C^3M$ for very large databases and to verify validity of the generated clustering structures.

(c) IR experiments to measure the performance of $C^3M$ by using different term weighting approaches in query and document matching.

Dealing with item (c) involves evaluation of effectiveness as well as efficiency. In an IR environment, efficiency involves many issues such as cost, time, number of disk accesses per retrieved document, amount of secondary storage required for the implementation, mean time required to process a query, etc. [45, 53, 55, 57], which fall within the realm of a separate performance study. However, the goal of this paper is limited to evaluate the validity and effectiveness of the concepts and/or methodologies that have been introduced via the CC concept.

### 4.1 Document Databases

In this study we have used two document databases: TODS214 and INSPEC. The TODS214 database contains the papers published by *Association for Computing*

*Machinery* in the journal of *Transactions on Database Systems* (ACM-TODS) during the period of March-1976 through September-1984. The database consists of 214 documents (papers). 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 [39].

The second database, INSPEC, is one of the largest standard test collections of the IR literature and it contains 12684 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 both TODS214 and INSPEC databases and the queries are provided in Sections 4.2 and 4.4.3.

### 4.2 Experimental Validation of Indexing-Clustering Relationships - Validity Experiments

For the experiments of this section we have used the binary and weighted versions of various D matrices of TODS214 and INSPEC. In the experiments, the entries of a weighted D matrix, $d_{ij}$ ($1 \le i \le m$, $1 \le j \le n$), indicate the number of occurrences of $t_j$ in $d_i$. The binary version of the same D matrix converts all non-zero $d_{ij}$ values to one.

Table 4 provides the information pertaining to the generation of D matrices for the TODS214 database. In this table, the frequency pair (min, max) indicates the frequency constraints that establish a stem as a term. For example, the first row of the table indicates that a stem must appear at least in one and at most in 180 documents to be used as a term. Actually, all stems are used as a term since the maximum observed stem frequency is 180. This makes the cardinality of the indexing vocabulary 2064 and it is indicated by n. t is the number of non-zero entries in the corresponding D matrix. In the rest of this paper, the D matrices of Table 4 will be identified by their frequency constraint used as subscripts, e.g., the D matrix corresponding to the second row of Table 4 will be named $D_{2\text{-}40}$.

Table 5 provides the same information as Table 4 for the INSPEC database. The first row of the table indicates the original D matrix of the INSPEC database. The INSPEC D matrices are named differently; the names are $D_{I1}, \ldots, D_{I7}$, where I stands for INSPEC.

Tables 6 and 7 summarize the experiments on indexing-clustering relationships for the TODS214 and the INSPEC databases, respectively. The second column of both tables gives the estimated number of clusters (see Eq. (3.17)). As it would be intuitively expected, the sparcity of the D matrix determines the number of clusters: the more sparse the D matrix is the more clusters it contains. For example, the original description of the INSPEC database shown by the matrix $D_{I1}$ of Table 5, has a size of 12684 by 14573 and contains 412,255 non-zero entries, and the estimated $n_c$ is 448. If we drop the stems appearing only in one document, then we obtain the matrix $D_{I2}$. $D_{I2}$ has a size of 12684 x 7435 and contains almost the same number of non-zero entries (405,117) as $D_{I1}$. This means the sparcity 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 332, i.e., the estimated $n_c$ of $D_{I2}$ is only 52% of the estimated $n_c$ of $D_{I1}$.



Table 4. Characteristics of the D Matrices for the TODS214 Database

| D Matrix | Frequency Min | Frequency Max | n | t |
|---|---|---|---|---|
| $D_{1\text{-}180}$ | 1 | 180 | 2064 | 11048 |
| $D_{2\text{-}40}$ | 2 | 40 | 1060 | 7446 |
| $D_{3\text{-}40}$ | 3 | 40 | 757 | 6840 |
| $D_{4\text{-}40}$ | 4 | 40 | 604 | 6381 |
| $D_{2\text{-}30}$ | 2 | 30 | 1045 | 6916 |
| $D_{3\text{-}30}$ | 3 | 30 | 742 | 6310 |
| $D_{4\text{-}30}$ | 4 | 30 | 589 | 5851 |
| $D_{2\text{-}20}$ | 2 | 20 | 988 | 5537 |
| $D_{3\text{-}20}$ | 3 | 20 | 685 | 4931 |
| $D_{4\text{-}30}$ | 4 | 20 | 532 | 4472 |

Table 5. Characteristics of the D Matrices for the INSPEC Database

| D Matrix | Frequency Min | Frequency Max | n | t |
|---|---|---|---|---|
| $D_{I1}$ | 1 | 5851 | 14573 | 412,255 |
| $D_{I2}$ | 2 | 5851 | 7435 | 405,117 |
| $D_{I3}$ | 2 | 3000 | 7431 | 387,811 |
| $D_{I4}$* | 2 | 1000 | 7382 | 308,886 |
| $D_{I5}$ | 3 | 5851 | 5677 | 401,601 |
| $D_{I6}$ | 3 | 3000 | 5673 | 384,295 |
| $D_{I7}$* | 3 | 1000 | 5624 | 305,370 |

(*) For these matrices m is equal to 12682.

In Tables 6 and 7, $n_{cw}$ and $n_{cb}$ indicate the number of clusters generated using the CC concept for the weighted and binary versions of the corresponding D matrices, respectively. The quantities $x_{dw}$ and $t_{gw}$ indicate the weighted depth of indexing and weighted term generality, respectively. $x_{dw}$ is obtained by finding the sum of all entries of a weighted D matrix and dividing the sum by m. $t_{gw}$ is obtained by dividing the same sum by n. That is, $x_{dw}$ is like $x_d$ but it is for a weighted D matrix and there is the same kind of correspondence between $t_{gw}$ and $t_g$. Similarly, $d_{cw}$ and $d_{cb}$ indicate the average size of a document cluster for the weighted and binary matrices, respectively. The average size of a term cluster is given by $d'_{cw}$ and $d'_{cb}$ for the respective term clusters. 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

Table 6. Results of the Indexing-clustering Relationships Experiments for the TODS214 Database

| Matrix | mxn t | $n_{cb}$ | $n_{cw}$ | $t_{gw}$ | $t_g$ | $d_{cb}$ | $d_{cw}$ | $x_{dw}$ | $x_d$ | $d'_{cb}$ | $d'_{cw}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $D_{1-180}$ | 40 | 39 | 36 | 8.99 | 5.35 | 5.49 | 5.94 | 86.75 | 51.63 | 52.92 | 57.33 |
| $D_{2-40}$ | 31 | 30 | 34 | 10.70 | 7.02 | 7.13 | 6.29 | 53.02 | 34.79 | 35.33 | 31.18 |
| $D_{3-40}$ | 24 | 23 | 27 | 13.90 | 9.04 | 9.30 | 7.93 | 49.15 | 31.96 | 32.78 | 28.04 |
| $D_{4-40}$ | 20 | 20 | 24 | 16.34 | 10.56 | 10.70 | 8.92 | 46.13 | 29.82 | 30.20 | 25.17 |
| $D_{2-30}$ | 32 | 32 | 36 | 10.01 | 6.62 | 6.69 | 5.94 | 48.90 | 32.32 | 32.66 | 29.03 |
| $D_{3-30}$ | 25 | 25 | 29 | 12.99 | 8.50 | 8.56 | 7.38 | 45.04 | 29.49 | 29.68 | 25.59 |
| $D_{4-30}$ | 22 | 21 | 26 | 15.20 | 9.93 | 10.19 | 8.23 | 42.01 | 27.34 | 28.05 | 22.65 |
| $D_{2-20}$ | 38 | 38 | 43 | 8.15 | 5.60 | 5.63 | 4.98 | 37.64 | 25.87 | 26.00 | 22.98 |
| $D_{3-20}$ | 30 | 30 | 35 | 10.55 | 7.20 | 7.13 | 6.11 | 33.78 | 23.04 | 22.83 | 19.57 |
| $D_{4-20}$ | 25 | 25 | 30 | 12.37 | 8.41 | 8.56 | 7.13 | 30.75 | 20.90 | 21.28 | 17.73 |

Table 7. Results of the Indexing-clustering Relationships Experiments for the INSPEC Database

| Matrix | mxn t | $n_{cb}$ | $n_{cw}$ | $t_{gw}$ | $t_g$ | $d_{cb}$ | $d_{cw}$ | $x_{dw}$ | $x_d$ | $d'_{cb}$ | $d'_{cw}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $D_{I1}$ | 448 | 439 | 475 | 50.34 | 28.29 | 28.89 | 26.70 | 57.84 | 32.50 | 33.20 | 30.68 |
| $D_{I2}$ | 232 | 227 | 275 | 97.08 | 54.49 | 55.88 | 46.12 | 56.91 | 31.94 | 32.75 | 27.04 |
| $D_{I3}$ | 243 | 238 | 294 | 90.33 | 52.19 | 53.29 | 43.14 | 52.92 | 30.58 | 31.22 | 25.28 |
| $D_{I4}$ | 303 | 294 | 364 | 71.36 | 41.84 | 43.14 | 34.84 | 41.62 | 24.36 | 25.11 | 20.28 |
| $D_{I5}$ | 179 | 175 | 218 | 126.07 | 70.74 | 72.48 | 58.18 | 56.43 | 31.66 | 32.44 | 26.04 |
| $D_{I6}$ | 187 | 183 | 233 | 117.24 | 67.74 | 69.31 | 54.44 | 52.44 | 30.30 | 31.02 | 24.35 |
| $D_{I7}$ | 234 | 227 | 289 | 92.77 | 54.30 | 55.88 | 43.88 | 41.14 | 24.08 | 24.78 | 19.46 |

the relationships and expressed by Eq.s (3.16) and (3.17) is very close to the actual $n_c$ value. $n_{cb}$ corresponds to binary D matrix. For the binary D matrices of the TODS database, the estimated $n_c$ and $n_{cb}$ values are identical in six of the nine experiments and differ only by 1 in the remaining cases. For the same database, the $n_{cw}$ values are on the average 15.4% higher than the estimated $n_c$ values.

The statistics for the INSPEC database are as follows. On the average, $n_{cb}$ and $n_{cw}$ values are, respectively, 2.37 percent less than and 19.36 percent higher than the estimated $n_c$ values. That is, the estimated $n_c$ and $n_{cb}$ are almost the same. Similarly, as indicated by Eq. (3.18), $t_g$ and the average size of clusters ($d_{cb}$, $d_{cw}$) assume very close values. For example, the $t_g$, $d_{cb}$, and $d_{cw}$ entries of the $D_{I1}$ matrix of INSPEC assume the following respective values: 28.29, 28.89, and 26.70. The validity of Eq. (3.19), i.e., the relationship between the average size of a term cluster ($d'_{cb}$, $d'_{cw}$) and $x_d$ can also be seen from Tables 6 and 7. For example, the $x_d$, $d'_{cb}$, $d'_{cw}$ entries of the $D_{I1}$ matrix of INSPEC assume the following respective values: 32.50, 33.20, and 30.68.



### 4.3 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 it must produce valid (meaningful) clusters. To satisfy these criteria we have defined two constraints to be met. 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 relate to other experiments.

#### 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 8. The execution times vary between 189 sec. to 74 sec., 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 ($D_{I1}$) database by various graph theoretical algorithms on an IBM 3083 BX environment is reported in [21, Table 1]. This IBM computer is a faster model than the IBM 4381 computer we have used in our experiments [29]. The reported execution times of the graph theoretical algorithms vary between 840 to 3416 seconds. These values are 4.4 to 18.1 times larger than the execution times of $C^3M$. The researchers El-Hamdouchi and Willett of the referenced study have tried using a version of the complete linkage method which is known to have good information retrieval performance [55, 57]. Due to the huge time and memory requirements encountered they were unable to use that particular version of the complete linkage algorithm [21]. Voorhees describes the implementation details of the same algorithm for the INSPEC database on a VAX 780 with a floating point accelerator and 6Mbytes of memory; however, no accurate clustering time performance statistics are available [56].

Based on the results just shown the execution time of the $C^3M$ algorithm is lower than those of the 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 be a constant across different D matrices. For all of the D matrices, $D_{I1}$ through $D_{I7}$, m is practically the same (either 12684 or 12682, see Table 5) therefore, we can ignore the factor m in this ratio. Table 8 shows $t_{gs}$, $x_d$, and the ratio $r = (x_d \times t_{gs})$ / (execution time). The fluctuations in r can be xplained

Table 8. Execution Time Behavior of the $C^3M$ Algorithm for the INSPEC Database

| Matrix type | $D_{I1}$ | $D_{I2}$ | $D_{I3}$ | $D_{I4}$ | $D_{I5}$ | $D_{I6}$ | $D_{I7}$ |
|---|---|---|---|---|---|---|---|
| Exec time (e.t.), sec | 189 | 126 | 114 | 85 | 105 | 95 | 74 |
| $t_{gs}$ | 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 |
| $r = (t_{gs} \times x_d)$ /(e.t.) | 0.53 | 0.44 | 0.48 | 0.53 | 0.55 | 0.59 | 0.63 |

by the inaccuracy in our assumptions. For example we are assuming that all select terms will be shared by non-seed documents. (Select terms are the terms used by seed documents, refer to Section 3.6.). This assumption can be regarded reasonable since seed documents are well separated due to the false seed elimination algorithm. We have two more assumptions: 1 - we are assuming that each document will contain $x_d$ distinct terms, and 2 - each select term is assumed to appear in $t_s/s$ number of seed documents. As it is indicated in Section 3.6, s and $t_s$ indicate the number of select terms and sum of seed document frequencies of select terms, respectively. We may agree that the first assumption is nearly accurate [20], however, since it is known that terms have hyperbolic frequency distributions [33] the second assumption may be less accurate. However, it has been experimentally shown that these assumptions are valid in an IR environment [20].

The value of r in Table 8 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 the most other clustering algorithms.

**The Second Usability Constraint: Validity of the Clustering Structure**

Let us consider validity of the generated clustering structure. The term "cluster validity" x refers mainly to the problem of deciding whether a clustering structure represents intrinsic character of clustered data set. In other words, the cluster validation problem is the evaluation of the results of a clustering process in an objective way. 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 the extensive list of references are given in [17, 30]. A brief overview of the problem from the viewpoint of IR is presented in [59].

The validity of a clustering structure can be basically stated in terms of external and internal criteria. An external criterion evaluates a clustering structure in terms of *a priori* information. In this case *a priori* information is usually the correct clustering structure for the clustered data. Internal criteria measure the correspondence between the clustering structure and data, using only the data themselves [30, p. 161]. Both external and internal criteria are based on the fact that a clustering structure is "valid" if it is significantly different from random case in the statistical sense.

Most of the known methods of cluster validity are inapplicable in the IR environment. First of all this is due to very large size of document databases. The size of document databases makes the amount of computations very high. Furthermore document databases introduce some conceptual problems. For example a typical external criterion requires the correct clustering structure for a data set. However, this is hard to realize for a large document database. This difficulty exists even for small databases due to large number of possibilities. (For m= 25 the number of possible clusters is greater than $4 \times 10^{18}$, see Section 2.) A solution to this problem could be to test the clustering algorithm to be used with an artificial data set with known clustering structure. This is inappropriate for an IR environment, since the generation of artificial data with known clustering structure requires the use of the Euclidian similarity measure [30]. As we have stated previously, the Euclidian measure is inappropriate in an IR environment.

This brief discussion of the cluster validity problem indicates that we need a cluster validity measure applicable in an IR environment. In partitioning type clustering environments of IR we can use the following approach to test validity.

Given is a query, let a target cluster be defined as a cluster which contains at least one relevant document for the query. For a given clustering structure it is easy to determine the number of target clusters of the query. Let $n_t$ indicate average number of clusters for a set of queries. If we preserve the same clustering structure ($n_c$ and the sizes of individual clusters) and assign documents randomly to the clusters, then we obtain random clustering for the same clustering structure. Intuitively, for a valid clustering structure, the average number of target clusters $n_t$ for all queries should be significantly different from the average random target clusters $n_{tr}$. That is to say, $n_t$ should be significantly less than $n_{tr}$. In the following experiments, we show that $n_t$ is significantly less than $n_{tr}$ for all the D matrices of the TODS214 database and $D_{I1}$ of the INSPEC database.

To obtain average number of target clusters for random clustering, $n_{tr}$, we will use Yao's theorem which is actually formulated for estimating block accesses [61]. It is assumed that the given are m documents grouped into $n_c$ number of clusters where $1 \leq n_c \leq m$ with each cluster containing exactly $m / n_c$ number of documents. If a query has k number of relevant documents ($k \leq m - m/n_c$) and if we select k random documents from the collection of size m,

then the expected number of clusters with at least one document selected is given by

$$n_c \times \left[ 1 - \prod_{i=1}^{k} \frac{my - i + 1}{m - i + 1} \right] \qquad \text{where } y = 1 - 1 / n_c$$

This formula assumes that all clusters have the same size. However, as would be expected, cluster sizes are not identical. The following corollary of Yao's theorem can be used for unequal sized clusters.

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

$$P_j = \left[ 1 - \prod_{i=1}^{k} \frac{m_j - i + 1}{m - i + 1} \right] \qquad \text{where } m_j = m - |C_j|$$

*Proof.* The proof follows steps similar to those in the proof of Yao's theorem [61], therefore we leave it to the reader. □

Accordingly, in random clustering with varying sizes of clusters and a query with k number of relevant documents we have:

$$\text{the number of random target clusters} = \sum_{i=1}^{n_c} P_j$$

In the case of random clustering it is therefore easy to find $n_{tr}$, i.e., average number of target clusters for all queries.

The case $n_t \ge n_{tr}$ suggests that tested clustering structure is invalid, since it is unsuccessful in placing the documents which are 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 for the validity of the clustering structure. However, as stated previously we have to show that $n_t$ is significantly less than $n_{tr}$. To show this we need to know the baseline distribution (probability density function), or reference population, of our test statistic: average number of target clusters for the query set. The baseline distribution of our test statistic is the distribution of the statistic under the null hypothesis, $H_o$.

*$H_o$: All permutations of the labels on m documents are equally likely.*

A Monte Carlo approach randomly selects a number of permutations r (i.e., randomly assigns documents to clusters) and calculates the average number of target clusters for this random case $n_t(r)$ and tries to estimate the distribution of $n_t$ under $H_o$. If the observed $n_t$ value is unusually small, i.e., if it is smaller than, say, 95% or 99% of the values in $\{n_t(r)\}$, then we can say that clusters are significantly different from random, i.e., the evidence suggests a nonrandom assignment of documents to clusters.

Notice that to estimate the baseline distribution we need to generate several random clustering structures if and only if $n_t < n_{tr}$. If this inequality is not



satisfied then it means that there is no need to generate the baseline distribution, since the clustering structure is not better than that of the average random case, i.e., we have strong evidence to assume that the clustering structure generated by the algorithm is invalid.

Table 9 gives the $n_t$ and $n_{tr}$ values for all D matrices of the TODS214 and INSPEC databases. The characteristics of the queries for both databases can be found in Section 4.4.3. 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 answer the question: is the $n_t$ value significantly different from (lower than) the corresponding $n_{tr}$ value? For this purpose we generated 10,000 random clustering structures for all the D matrices of the TODS214 database and obtained a histogram of the observed $n_t(r)$ values. The same is done for the $D_{11}$ matrix of the INSPEC database (due to the long computational requirements of the INSPEC database we generated 5,000 random cases).

Table 9. The Comparison of $C^3M$ and Random Clustering in Terms of Average Number of Target Clusters for All Queries

TODS214

| D matrix | $D_{1\text{-}180}$ | $D_{2\text{-}40}$ | $D_{3\text{-}40}$ | $D_{4\text{-}40}$ | $D_{2\text{-}30}$ | $D_{3\text{-}30}$ | $D_{4\text{-}30}$ | $D_{2\text{-}20}$ | $D_{3\text{-}20}$ | $D_{4\text{-}20}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $n_t$ | 3.466 | 3.379 | 3.379 | 3.293 | 3.603 | 3.397 | 3.586 | 3.879 | 3.707 | 3.931 |
| $n_{tr}$ | 4.744 | 4.757 | 4.643 | 4.605 | 4.808 | 4.706 | 4.683 | 4.901 | 4.813 | 4.755 |

INSPEC

| D matrix | $D_{11}$ | $D_{12}$ | $D_{13}$ | $D_{14}$ | $D_{15}$ | $D_{16}$ | $D_{17}$ |
|---|---|---|---|---|---|---|---|
| $n_t$ | 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 |

The 5,000 $n_t(r)$ values of INSPEC were grouped into 10 bins and Figure 7 shows the percentage counts of each bin, i.e., the figure shows the approximate baseline distribution 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 $C^3M$ are not an artifact of the algorithm, on the contrary they are valid and useful for IR. The question remains to be answered is the CBR effectiveness which is discussed next.

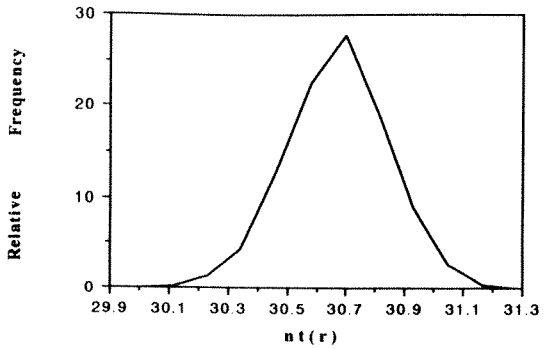


Figure 7. Histogram of the $n_t(r)$ values for the INSPEC database. (min= 30.051, max= 31.221, average= 30.659, standard deviation= 0.172, bin length= 0.117)

### 4.4 IR Experiments

#### 4.4.1 Evaluation Measures for Retrieval Effectiveness

In this paper we assess the effectiveness of $C^3M$ by comparing its CBR performance with that of FS and with CBR performance of some other clustering algorithms used in the current IR literature. FS returns documents in descending order of similarity with the user query. Then we can produce a graph showing precision values at standard recall levels (typically 0.1, 0.2, . . . , 1.0) and obtain the average of these precisions for all user queries [45, 53]. Lately there has been a common belief that the approach just described for effectiveness measurement is not appropriate for CBR [21, 24, 25, 55, 57], since it does not give a full ranking of a document database but just a portion of it.

The effectiveness measures used in this study are the average precision for all queries, total number of relevant documents retrieved for all queries, T, and the total number of queries with no relevant documents, Q [21, 25, 55]. Obviously, an effective system yields higher precision and T, and lower Q. As in other studies, the recall value is not used as an effectiveness measure. This is because, as the number of relevant documents retrieved increases both precision and recall increase and result in relatively identical effectiveness [55].

The effectiveness measures are obtained after retrieving ten and twenty documents. The reason for this is the assumption that a typical user will evaluate the relevance of the first ten to twenty documents returned by the system and after that he(she) is either satisfied or the query is reformulated.

In CBR, first the clusters are ranked according to their similarity with the user query. Then the $d_s$ documents of the first $n_s$ clusters are selected according to their similarity with the query. (The selected clusters and



documents must have non-zero similarity with the query.) Obviously, as we increase the value of $n_s$ the number of documents to be matched and ranked with respect to the query will increase. However, efficiency considerations dictate fewer $n_s$ and a smaller set of $d_s$ which contradict effectiveness considerations.

#### 4.4.2 Term Weighting or Query Matching Function Selection

The ranking of documents is affected by the matching function which calculates the similarity of user query with documents. The same is true for clusters and documents of the CBR. A recent study has evaluated the IR effectiveness for a total of two hundred and eighty seven distinct matching functions [46]. The study also has shown that the type of matching function drastically affects the IR performance. Two hundred and eighty seven matching functions have resulted from possible term weighting approaches used for both document and query terms. For term weighting there are basically three components: term frequency component (TFC), collection frequency component (CFC), and normalization component (NC). The weights of the terms of a document and a query (notated by $w_{d_j}$ and $w_{q_j}$ $(1 \le j \le m)$ 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

$$\text{similarity } (Q, D) = \sum_{i=1}^{t} w_{d_i} \times w_{q_i}$$

In our experiments, the matching function for queries and centroids is defined in a similar way. A brief description of TFC, CFC, and NC is provided in the following and the details can be found in [46].

Let us consider $d_{ij}$. The weight $w_{d_{ij}}$ of a term $t_j$ in $d_i$ will be (TFC x CFC x NC). The three possibilities for TFC are symbolized by b, t, n. b is binary weight, in this case ignore the term frequency and take TFC= 1; t is term frequency tf, which means TFC is equal to the number of occurrences of $t_j$ in $d_i$; n is augmented normalized term frequency and defined as (0.5 + 0.5 x tf / maxtf) where maxtf is max $(d_{i1}, d_{i2}, \ldots, d_{in})$, i.e., the maximum number of times any term appears in $d_i$.

The three possibilities for CFC are notated by x, f, 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 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 it is similar to f both in terms of definition and performance [46]. We did not use it in our experiments.

For normalization, i.e., the NC component, there are two possibilities notated by x and c. x means no change, i.e., take NC= 1; c means cosine normalization where each term weight TFC x CFC is divided by a factor representing Euclidian

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, 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 (Eq. (1.1) of Section 1). The combinations tfc, nfc, for documents and nfx, tfx, bfx for queries lead to better IR performance [46]. This gives us 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, tfc-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 10 shows these seven combinations, i.e., 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.

Table 10. 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 |

### 4.4.3 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 it would be expected, centroid length affects the effectiveness of the CBR [55, 57]. 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 $D_{1\text{-}180}$ and $D_{I1}$ matrices of the TODS214 and the INSPEC databases, respectively. The reason for this choice is that the $D_{I1}$ 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 $D_{I1}$, all the stems are chosen as terms and so is with the matrix $D_{1\text{-}180}$. This provides a common characteristics for both matrices.

For the TODS214 database we have used the maximum centroid lengths 50, 100, 150, 200 and for the INSPEC database the maximum lengths we have used are 250 and 500. The characteristics of the centroids generated for both databases are shown in Table 11. In Table 11, 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; %n indicates the percentage of D matrix terms that appear in at least one centroid (i.e., centroid terms). $t_{gc}$ stands for centroid terms. The last entry, %D, of the table 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 very slightly from the centroid length of 50 to 150 and after this point it almost remains the same. The observations just presented lead us to the following conclusion: the increase in centroid length first increases the IR effectiveness but after a threshold it does not provide any further improvement. This is also consistent with the findings of Voorhees [57, Table 4]. In this study, the performance figures for TODS214 and INSPEC will be 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% of the original D matrix. Considering the storage requirements of the various hierarchical clustering algorithms implemented for the same database [55, pp. 110-112; 63] this percentage is quite reasonable.

Table 11. Characteristics of the Centroids

TODS214 ($D_{1\text{-}180}$)

| Cent. Len. (max $x_c$) | Avg. Cen. Len $x_c$ | $t_{gc}$ | $\%X_c$ | %n | %D |
|---|---|---|---|---|---|
| 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 ($D_{I1}$)

| Cent. Len. (max $x_c$) | Avg. Cen. Len $x_c$ | $t_{gc}$ | $\%X_c$ | %n | %D |
|---|---|---|---|---|---|
| 250 | 237.09 | 12.96 | 51 | 60 | 27 |
| 500 | 399.06 | 14.74 | 86 | 88 | 46 |

The query characteristics are presented in Table 12. In both of the databases, query vectors are created in the same manner as the D matrices are created. The query terms are the query word stems which also appear in the related D matrix. Thirty nine of the TODS214 queries are constructed from five textbooks on databases. If a chapter cites more than one TODS214 article then the titles and subtitles of that chapter are used as a query text. The documents corresponding to the cited articles are assumed to be relevant to the query. The query set is almost evenly distributed among the five texts. The other set of nineteen queries are explained in [9, 39]. The queries of the INSPEC database are collected at Cornell and Syracuse universities. 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 incorporated with the query matching functions whenever needed.

Table 12. Characteristics of the Queries

| Database | No. of Queries | Avg. Terms per Query | Avg. Rel. Docs. per Query | Total Relevant Documents | No of Distinct Docs. Retrieved | No. of Dist. Terms for Query Def. |
|---|---|---|---|---|---|---|
| TODS214 | 58 | 13.36 | 5.26 | 305 | 126 | 291 |
| INSPEC | 77 | 15.82 | 33.03 | 2543 | 1940 | 577 |

#### 4.4.4 The Effectiveness of CBR

**INSPEC Database**

In CBR the first step is to decide about 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 go upto a certain $n_s$, and after this (saturation) point the retrieval effectiveness remains the same (in some cases it could go worse). In our experiments, for the term weighting strategies of TW1 through TW7, and the INSPEC database 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 $n_c = 475$). For all the queries and for all the matching functions TW1 through TW7 the average percentage of the matched documents is 12.5 percent which is slightly higher than 10.5, i.e., the percentage of the selected clusters. These comparable values also indicate that documents are evenly distributed among clusters.

The performance figures for each of TW2, TW3, TW4 are similar, TW2 being the best and TW4 being the worst. The same is true for each of TW5, TW6, TW7, 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 ten documents for INSPEC are shown in Figure 8 for TW1, TW2, and TW5 with different values of $n_s$. The horizontal lines indicate the precision values for the corresponding full search (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., increase in T and decrease in Q). The plot also indicates that the best term weighting strategy (i.e., matching function) is TW2. The 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 $n_s = 30$. In CBR, if we examine ten to twenty percent of the database then the search process can be considered efficient [42]. The case with $n_s = 50$ returns 12.5 percent of the documents to examine, which is within the above efficiency range and it is also the saturation point for the retrieval effectiveness. Because of this and to decrease the volume of the presented data the CBR effectiveness will be analyzed at $n_s = 50$ for the INSPEC database. Figure 8 shows the general picture for the other $n_s$ values.



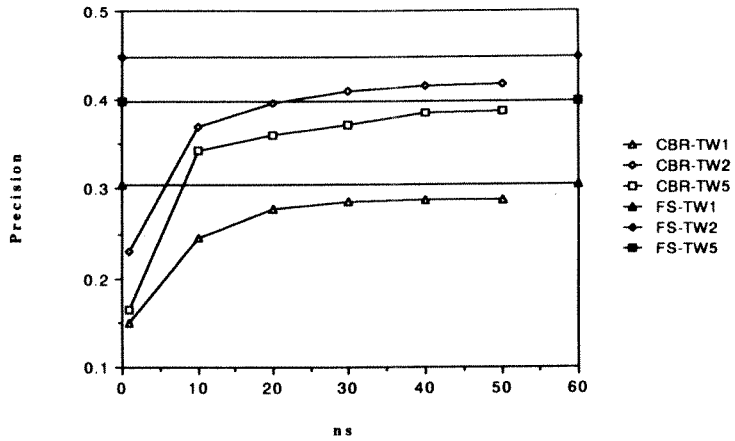


Figure 8. Precision with ten documents ($d_s = 10$) for different values of $n_s$ (INSPEC).

Table 13 shows the effectiveness figures for the INSPEC database. At this point, it should be stated that the effectiveness measures of precision and T are closely related to each other, and this is easy to show. Let $r_i$ indicate the number of relevant documents for query-i after examining $d_s$ number of documents in the case of FS. Assuming that for each query the system is able to find $d_s$ documents with non-zero similarity, then for nq number of queries the average precision for FS can be expressed as

$$[1 / (nq \times d_s)] \times (r_1 + r_2 + \ldots + r_{nq}) = [1 / (nq \times d_s)] \times T$$

where T is equal to $(r_1 + r_2 + \ldots + r_{nq})$. In the CBR case we will have exactly the same expression. Accordingly, we will have the following identity:

(precision for FS) / (precision for CBR) = (T for FS) / (T for CBR)

The first and second rows of each item of Table 13 is for $d_s$ values of 10 and 20, respectively. For INSPEC the above relationship between precision and T is always valid, therefore, it suffices to consider only one of them (i.e., either precision or T). We will consider precision.

Table 13. Effectivenesses of FS and CBR for the INSPEC Database

| Effec. | TW1 | | TW2 | | TW3 | | TW4 | | TW5 | | TW6 | | TW7 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Measr. | FS | CBR | FS | CBR | FS | CBR | FS | CBR | FS | CBR | FS | CBR | FS | CBR |
| Prec. | .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 |

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 the cases the average percentage decrease of the precision of CBR with respect to FS is 3.9. The CBR with the matching function TW2 gives the precision values of .418 and .336 for the $d_s$ values of 10 and 20, respectively. 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. For $d_s$= 20 the CBR precision (.336) of TW2 is greater than all the others except TW4 (whose precision value is .342).

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

The experimental results indicate that matching function is very influential on retrieval effectiveness. TW1, the cosine similarity measure, is not a good choice since it considers only the term frequencies. TW2 looks promising. The CBR with TW2 is better than those of FS of the other cases: the precision increase with respect to FS is between 37.1 percent to 1 percent with an average of 11.6 percent for $d_s$= 10.

Let us compare our results with the other algorithms tested on the INSPEC database. Voorhees reports her effectiveness experiments in [55, 57]. Since the results of [57] are better we will consider them first. This work reports the results of the complete linkage algorithm with three different centroid lengths. The Q values of this reference are similar to our case. The only effectiveness measure that remains to be compared is precision. The precision values for FS and CBR for $d_s$= 10 and 20 with different centroid lengths are given in Table 14 (CBR1, CBR2, and CBR3 indicate the results for the maximum centroid lengths of 75, 100, and 250, respectively.). Even though there is no exact definition, the matching function of the referenced study is similar to our TW6. For FS our matching function gives the precisions of .386 and .319. The best result in that study is with CBR2 (.397) which is 3.4 percent better than that of FS for $d_s$= 10. For CBR3 with $d_s$= 20 the difference is 9.8 percent worse than that of FS. The worse 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, i.e., .418 (for $d_s$= 10) and .336 (for $d_s$= 20) which are 5.3 percent (100 x .418/.397) and 7.0



percent (100 x .336/.314) better than the best results of Voorhees for $d_s$= 10 and 20, respectively. Obviously, if Voorhees had changed the matching function she could have obtained better results, with no guarantee, however. This discussion indicates that our results are compatible with the results of [57] on the INSPEC database.

We should also point out that all of our CBR results are better than those of performed by using the single link and average link hierarchical clustering methods reported in [55, p.88]. However, we should state that the matching functions of the compared experiments are not exactly the same.

Table 14. Precision of FS and CBR for the INSPEC Database (complete linkage) (taken from Voorhees [57])

| FS | CBR1 | CBR2 | CBR3 |
|---|---|---|---|
| .384 | .387 | .397 | .373 |
| .315 | .308 | .314 | .284 |

Next, let us compare our experiments with the results of El-Hamdouchi and Willett [21] 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 the paper this is called group average), and the Ward method. Their complete linkage algorithm is different from the one reported in [55, 57] 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 [21, Table 3]. The effectiveness measures that are common with our study are T and Q. The precision values are not reported, however, as we have shown, T and precision for a large database such as INSPEC are directly proportional. They used the cosine similarity coefficient 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. In terms of the document and query term weighting components this function is equivalent to the case of bxc.bfx. As explained in Section 4.4.2, b indicates that all vectors (i.e., query, document, centroid) will be treated as binary, c indicates cosine normalization, and f indicates inverse document frequency.

The "best" T and Q results for single link (the reference [21] uses SL as the acronym), complete linkage (CL), average link (GA), and the Ward method (WM) versus our $C^3M$ results are shown in Table 15 based on the number of selected clusters, $n_s$, of 10 and 50. Before comparing the results we should point out that cosine similarity is not a very good matching function (also refer to the results of TW1 of Figure 8). The results of $C^3M$ using binary terms and the cosine coefficient are worse than those of the cases of TW1 through TW7 (refer to Table 13). All the values of Table 15 are obtained after examining ten documents ($d_s$= 10). The results of the table indicate that $C^3M$ always outperforms the hierarchical algorithms. (As a sidelight we should point out that in the comparisons we have used the "best" results of hierarchical clustering against our cases for only $n_s$= 10 and 50, a very modest comparison!)

Table 15. Best T and Q Values for Various Hierarchical Clustering Algorithms (taken from [21]) and $C^3M$ with $n_s$= 10 and 50

| Hierarchical Clustering Algorithms | | | | | $C^3M$ | |
|---|---|---|---|---|---|---|
| Effective Meas. | 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 |

The percentage improvement in effectiveness using $C^3M$ ($n_s$= 10) with respect to the hierarchical algorithms is shown in Table 16. 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. Table 16 shows the effectiveness improvements in terms of T and Q. The percentage improvement in T ranges from 15.1 to 63.5 with an average of 47.5. The decrease in Q (the number of documents with no relevant documents retrieved) ranges from 44.4 percent to 60.5 percent with an average of 54.7.

Table 16. Percentage Improvement in Effectiveness using $C^3M$ with Respect to Hierarchical Algorithms (for $n_s$= 10)

| Effect. Meas. | Single Link | Complete Linkage | Average Link | Ward Method |
|---|---|---|---|---|
| T | 62.2 | 63.5 | 15.1 | 49.3 |
| Q | -59.5 | -60.5 | -44.4 | -54.5 |

**TODS214 Database**

The queries of TODS214 are binary. Accordingly, the matching functions TW2, TW3, and TW4 become equivalent, and the same is true for TW5, TW6, and TW7. This is because, by definition, the query term weighting components nfx and tfx are equivalent to bfx for binary query terms (refer to Section 4.4.2). Notice that only the definition of TW4 and TW7 are consistent with the binary nature of the queries. TW2 and TW3 are transformed to TW4. Similarly, TW5 and TW6 are transformed to TW7.

To compare FS and CBR for both cases we must have $d_s$= 10 and $d_s$= 20 (as defined earlier $d_s$ is the number of documents to be returned to the user). For the FS case when $d_s$= 20 the number of documents with non-zero similarity is, on the average, slightly less than twenty (to be exact, it is 19.91). Beginning at $n_s$= 4 the matching functions return close to twenty documents (to be exact 18.78, 18.84, 18.24 average number of documents, respectively, for TW1, TW2, and TW5 when $n_s$= 4). In short, the CBR and FS results are comparable starting at $n_s$= 4.

Table 17 shows the effectiveness figures for the TODS214 database. The first and second rows of each item indicate the cases for $d_s$= 10 and $d_s$= 20,



respectively. As can be seen from the table, $n_s$= 6 is the saturation point, i.e., after this point expansion of new clusters does not increase effectiveness. 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).

Table 17. Effectiveness of FS and CBR for the TODS214 Database

| Effec. | TW1 | | | | TW4 (TW2, TW3) | | | | TW7 (TW65, TW6) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Measr. | 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 |
| Prec. | .166 | .160 | .155 | .159 | .179 | .186 | .181 | .179 | .179 | .193 | .188 | .186 |
| | .110 | .116 | .116 | .114 | .128 | .127 | .124 | .124 | .124 | .129 | .128 | .126 |
| T | 96 | 93 | 90 | 92 | 104 | 108 | 105 | 104 | 104 | 112 | 109 | 108 |
| | 127 | 130 | 132 | 131 | 148 | 142 | 141 | 142 | 143 | 141 | 147 | 144 |
| 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 |

Table 18. Percentage Difference of CBR Precision over FS After Examining $n_s$ Number of Clusters

| TW1 | | | TW4 (TW2, TW3) | | | TW7 (TW5, TW6) | | |
|---|---|---|---|---|---|---|---|---|
| $n_s$=4 | $n_s$=6 | $n_s$=8 | $n_s$=4 | $n_s$=6 | $n_s$=8 | $n_s$=4 | $n_s$=6 | $n_s$=8 |
| -3.6 | -6.6 | -4.2 | +3.9 | +1.1 | 0.0 | +7.8 | +5.0 | +3.9 |
| +5.5 | +5.5 | +3.6 | -0.8 | -3.1 | -3.1 | +4.0 | +3.2 | +1.6 |

Table 18 shows the percentage difference of CBR precisions from those of FS after selecting four, six, and eight clusters. For TODS214 the best matching function is TW7. In the case of INSPEC the best matching function was 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.

The CBR with the matching functions TW4 and TW7 is better than those of FS with TW1. For example, the TW1 precision (0.166) for $d_s$= 10 is less than all of the CBR precision values with TW4 and TW7. This again shows that matching function (or equivalently term weighting) plays an important role in performance. The Q values of CBR are slightly worse than those of FS; however, Table 18 shows that the CBR with TW7 provides a 7.8 percent effectiveness increase with respect to FS.

## 5. CONCLUSION

In this study a new clustering methodology called $C^3M$ has been introduced. $C^3M$ relies on its heuristic called the cover coefficient (CC) concept which indicates relationships among documents (or terms) based on a two-stage

probability experiment. In the paper, a hypothesis is introduced to estimate the number of clusters in a document database. The hypothesis states that 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. By various theorems and corollaries it is shown that the CC concept satisfies the hypothesis and can be used to find the number of clusters indicated by the hypothesis. The CC concept is also used for selection of cluster seeds and to relate indexing and clustering analytically. In the experiments conducted, the relationships indicated by the CC concept are validated.

In the paper, it is shown that the complexity of $C^3M$ is better than most other clustering algorithms whose complexity range in $O(m^2)$ or $O(m^3)$. 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 the 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 linkage method which is known to have good performance. It is known that the implementation of complete linkage method is very demanding in terms of computation time and memory and can only be used by some approximations, if any, in very large database environments [56]. 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 full search 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 to future research. (1) Regarding the indexing-clustering relationships shown in the paper there remain the question: what is the effect of variations of indexing on CBR performance [19]? (2) In an IR system it may be worth to combine FS and CBR since they may return different sets of documents to the user [25]. Although this would increase the cost, this combination may be worth to consider if one aims to increase effectiveness. (3) The efficiency comparison of IIS (inverted index search) and CBR may be worth to pursue [55, 57]. 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 to incorporate the CC concept in 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 [8] for the TODS214 database.

## ACKNOWLEDGMENTS

The authors are indebted to Dr. Jon M. Patton of Miami University for streamlining our proofs of the theorems contained in the paper. The authors are also indebted to Drs. Gerard Salton of Cornell University for helping us to acquire the INSPEC database and Peter Willett of 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.** Bertziss, A. T. *Data Structures Theory and Practice*. Academic Press, New York, 1975.

**3.** Can, F., Ozkarahan, E. A. "A Clustering Scheme." In *Proceedings of the 6th Annual International ACM-SIGIR Conference* (1983), ACM, New York, 115-121.

**4.** Can F., Ozkarahan, E. A. "Two Partitioning Type Clustering Algorithms." *Journal of the American Society for Information Science*. 35, 5 (September 1984), 268-276.

**5.** Can, F., Ozkarahan, E. A., "Similarity and Stability Analysis of the Two Partitioning Type Clustering Algorithms." *Journal of the American Society for Information Science*. 36, 1 (January 1985), 3-14.

**6.** 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 University, Ankara, 1985.

**7.** Can, F., Ozkarahan, E. A. "Concepts of the Cover Coefficient Based Clustering Methodology." In *Proceedings of the 8th Annual International ACM-SIGIR Conference* (June 1985), ACM, New York, 204-211.

**8.** Can, F., Ozkarahan, E. A. "Dynamic Cluster Maintenance." *Information Processing and Management*. 25, 3 (1989), 275-291.

**9.** Clarke, C. H. *Performance Assessments to Test the Effectiveness of the Cover Coefficient Based Clustering Information Retrieval System*. MSc Thesis, Dept. of Computer Science, Arizona State University, Tempe, AZ, 1987.

**10.** Crawford, R. G. "The Relational Model in Information Retrieval." *Journal of the American Society for Information Science*. 32, 1 (January 1981), 51-64.

**11.** Croft, W. B. "Clustering Large Files for Documents Using the Single-Link Methods." *Journal of the American Society for Information* Science. 28 (1977), 341-344.

59. Croft, W. B. "A Model of Cluster Searching Based on Classification." *Information Systems*. 5 (1980), 189-195.

**13.** Crouch, D. B. "A File Organization and Maintenance Procedure for Dynamic Document Collections." *Information Processing and Management*. 11 (1975), 11-21.

**14.** Dattola, R. T. "Experiments with a Fast Algorithm for Automatic Classification." In G. Salton, Ed. *The Smart Retrieval System-Experiments in Automatic Document Processing*. Prentice Hall, Englewood Cliffs NJ, 1971 (Chap. 12).

**15.** Deogun, J. S., Raghavan, V. V. "User Oriented Document Clustering: A Framework for Learning in Information Retrieval." In *Proceedings of the 9th Annual International ACM-SIGIR Conference* (September 1986), ACM, New York, 157-163.

**16.** Deogun, J. S., Raghavan, V. V., Tsou, T. K. W. "Organization of Clustered Files for Consecutive Retrieval." *ACM Transactions on Database Systems*. 9, 4 (December 1984), 646-671.

**17.** Dubes, R., Jain, A. K. "Clustering Methodologies in Explanatory Data Analysis." In M. C. Yovits, Ed. *Advances in Computers*. Academic Press, New York, 1980, 113-128.
**18.** Dubes, R. C."How Many Clusters are Best? - An Experiment." *Pattern Recognition*. 20, 6 (1987), 645-663.
**19.** El-Hamdouchi, Willett, P. "Techniques for the Measurement of Clustering Tendency in Document Retrieval Systems." *Journal of Information Science*. 13 (1987), 361-365.
**20.** El-Hamdouchi, A., Willett, P. "An Improved Algorithm for the Calculation of Exact Term Discrimination Values." *Information Processing and Management*. 24, 1 (1988), 17-22.
**21.** El-Hamdouchi, A., Willett, P. "Comparison of Hierarchical Agglomerative Clustering Methods for Document Retrieval." *The Computer Journal*. 32, 3 (June 1989), 220-227.
**22.** Everitt, B. S. "Unresolved Problems in Cluster Analysis. *Biometrics*. 35, (1979), 169-181.
**23.** Everitt, B. S. *Cluster Analysis*. Halsted Press Div. of John Wiley and Sons, New York, 1980.
**24.** Griffiths, A., Robinson, L. A., Willett, P. "Hierarchical Agglomerative Clustering Methods for Automatic Document Classification." *Journal of Documentation*. 40, 3 (Sept., 1984), 175-205.
**25.** Griffiths, A., Luckhurst, C., Willett, P. "Using Interdocument Similarity Information in Document Retrieval Systems." *Journal of the American Society for Information Science*. 37, 1 (1986), 3-11.
**26.** Harding, A. F., Willett, P. "Indexing Exhaustivity and the Computation of Similarity Matrices." *Journal of the American Society for Information Science*. 30 (1979), 224-228.
**27.** Hartigan, J. A. *Clustering Algorithms*. John Wiley and Sons, New York, 1975.
**28.** Hodges, J. L., Lehmann, E. L. *Basic Concepts of Probability and Statistics*. Holden-Day Inc, San Fransisco, 1964.
**29.** IBM. "IBM 3083 Processor Complex." IBM Document No. G221-2417-0, March 1982.
**30.** Jain A. K., Dubes, R. C. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, 1988.
**31.** Kusiak, A., Chow, W. S. "An Efficient Cluster Identification Algorithm." *IEEE Trans. on Systems, Man, and Cybernetics*. SMC-17, 4 (July/August 1987), 696-699.
**32.** Kutluay, M. S. *A Validity Analysis of the Cover Coefficient Concept on Cluster Analysis*. MSc Thesis, Dept. of Electrical and Electronics Engineering, Middle East Technical University, Ankara, 1986.
**33.** Lynch, M. F. "Variety Generation - A Reinterpretation of Shannon's Mathematical Theory of Communication, and Its Implications for Information Science." *Journal of the American Society for Information Science*. 28, 19 (1977), 19-25.
**34.** Maron, M. E. "Depth of Indexing." *Journal of the American Society for Information Science*. 19 (1978), 224-228.
**35.** Milligan, Cooper An Examination of Procedures for Determining the Number of Clusters in a Data Set." *Psychometrika*. 50, 2 (June 1985), 159-179.
**36.** Omiecinski, E. R. *Algorithms for Record Clustering and File Reorganization*. Ph. D. Dissertation. Field of Computer Science, Northwestern University, Evanston, Ill, 1984.



**37.** Ozkarahan, E. A., Can, F. "An Integrated Fact/Document Information System for Office Automation." *Information Technology: Research and Development*. 3, 3 (1984), 142-156.
**38.** Ozkarahan, E. *Database Machines and Database Management*. Prentice Hall, Englewood Cliffs NJ, 1986.
**39.** Ozkarahan, E. A., Can, F. "An Automatic and Tunable Indexing System." In *Proceedings of the 9th Annual International ACM-SIGIR Conference* (September 1986), ACM, New York, 234-243.
**40.** Raghavan, V. V., Ip, M. Y. L. "Techniques for Measuring the Stability of Clustering: A Comparative Study." In *Proceedings of the 5th Annual International ACM-SIGIR Conference* (1982), Springer Verlag, Berlin.
**41.** Rasmussen, E. M., Willett, P. "Non-Hierarchic Document Clustering Using the ICL Distributed Array Processor." In *Proceedings of the 10th Annual International ACM-SIGIR Conference* (June 1987), ACM, New York, 132-139.
**42.** Salton, G. "Clustering Search Strategies and the Optimization of Retrieval Effectiveness." In G. Salton, Ed. The Smart Retrieval System-Experiments in Automatic Document Processing. Prentice Hall, Englewood Cliffs, NJ, 1971 (Chap. 10).
**43.** Salton, G. *Dynamic Information and Library Processing*. Prentice Hall, Englewood Cliffs NJ, 1975.
**44.** Salton, G., Wong, A. "Generation and Search of Clustered Files." *ACM Transactions on Database Systems*. 3, 4 (Dec. 1978), 321-346.
**45.** Salton, G., McGill, M. J. *Introduction to Modern Information Retrieval*. McGraw Hill, New York, 1983.
**46.** Salton, G., Buckley, C. "Term-Weighting Approaches in Automatic Text Retrieval." *Information Processing and Management*. 24, 5 (1988), 513-523.
**47.** Salton, G. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison Wesley, Reading, Massachusetts, 1989.
**48.** Scheck, H. J. "Methods for the Administration of Textual Data in Database Systems." In *Proceedings of the Joint BCS and ACM Symp.*, 1980, 218-235.
**49.** Sneath, P. H. A., Sokal, R. R. *Numerical Taxonomy*. W. H. Freeman, San Fransisco, 1973.
**50.** Sparck, Jones, K. "Collection Properties Influencing Automatic Term Classification Performance." *Information Processing and Management*. 9 (1973), 499-513.
**51.** Stonebraker, M., et. al. "Document Processing in Relational Database System." *ACM Transactions on Office Information Systems*. 1 (1983), 143-158.
**52.** Ural, M. H. *Performance Evaluation of the Cover Coefficient Based Clustering and Cluster Maintenance Methodology in Information Retrieval*. MSc Thesis, Dept. of Electrical and Electronics Engineering, Middle East Technical University, Ankara, 1986.
**53.** Van Rijsbergen, C. J. *Information Retrieval, 2nd ed.* Butterworths, London, 1979.
**54.** Voorhees, E. M. "The Cluster Hypothesis Revisited." In *Proceedings of the 8th Annual International ACM-SIGIR Conference* (June 1985), ACM, New York, 188-196.
**55.** Voorhees, E. M. *The Effectiveness and Efficiency of Agglomerative Hierarchical Clustering in Document Retrieval*. PhD Thesis, Cornell University, Ithaca, NY, 1986.
**56.** Voorhees, E. M. "Implementing Agglomerative Hierarchical Clustering Algorithms for Use in Document Retrieval." *Information Processing and Management*. 22, 6 (1986), 465-476.

**57.** Voorhees, E. M. "The Efficiency of Inverted Index and Cluster Searches." In *Proceedings of the 9th Annual International ACM-SIGIR Conference* (Sept., 1986), ACM, New York, 164-174.

**58.** Willett, P. "Document Clustering Using an Inverted File Approach." *Journal of Information Science*. 2 (1980), 223-231.

**59.** Willett, P. "Recent Trends in Hierarchical Document Clustering: A Critical Review." *Information Processing and Management*. 24, 5 (1989), 577-597.

**60.** Wong, S. K. M., Ziarko, W., Raghavan, V. V., Wong, P. C. N. "On Extending the Vector Space Model for Boolean Query Processing." In *Proceedings of the 9th Annual International ACM-SIGIR Conference* (September 1986), ACM, New York, 175-185.

**61.** Yao, S. B. "Approximating Block Accesses in Database Organizations." *Communications of the ACM*. 20, 4 (April 1977), 260-261.

**62.** Yu, C. T., Chen, C. H. "Adaptive Document Clustering." In *Proceedings of the 8th Annual International ACM-SIGIR Conference*. (June 1985), ACM, New York, 197-203.

**63.** Yu, C. T., Suen, C., Lam, K., Siu, M. K. "Adaptive Record Clustering." *ACM Transactions on Database Systems*. 10 (1985), 180-204.