Computer Science and Systems Analysis

Computer Science and Systems Analysis

Technical Reports

Miami University *Year* 1987

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

Fazli Can* Esen Ozkarahan†

*Miami University, commons-admin@lib.muohio.edu

†Miami University, commons-admin@lib.muohio.edu

MIAMI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & SYSTEMS ANALYSIS

TECHNICAL REPORT: MU-SEAS-CSA-1987-002

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

Fazli Can and Esen A. Ozkarahan

School of Engineering & Applied Science | Oxford, Ohio 45056 | 513-529-5928

CONCEPTS AND EFFECTIVENESS OF THE COVER COEFFICIENT BASED CLUSTERING METHODOLOGY FOR TEXT DATABASES

by

Fazli Can
Systems Analysis Department
Miami University
Oxford, Ohio 45056

Esen A. Ozkarahan
Computer Science
Arizona State U.
Tempe, Az 85287

Working Paper #87-002

12/87

CONCEPTS AND EFFECTIVENESS OF THE COVER COEFFICIENT BASED CLUSTERING METHODOLOGY FOR TEXT DATABASES

Fazli CAN*
Esen A. OZKARAHAN+

12/14/87

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. The CC concept is used also to identify the cluster seeds, to form clusters with the seeds, and to calculate Term Discrimination and Document Significance values (TDV, DSV). TDVs and DSVs are used to optimize document descriptions. The CC concept also relates indexing and clustering analytically. Experimental results indicate that the clustering performance in terms of the percentage of useful information accessed (precision) is forty percent higher, with accompanying reduction in search space, than that of random assignment of documents to clusters. The experiments have validated the indexing-clustering relationships and shown improvements in retrieval precision when TDV and DSV optimizations are used.

Categories and Subject Descriptors: H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing - *indexing methods*; 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, term discrimination value, clustering-indexing relationships

---

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

+ Dept. of Computer Science, Arizona State University, Tempe, AZ 85287

1. INTRODUCTION

An Information Retrieval System (IRS) tries to find documents that are relevant to user queries. An IRS environment has similarities as well as differences as compared to a Database Management System (DBMS) environment. This is mostly due to the unformated nature of data (e.g., journal and newspaper articles, reports, books, all of which are referred to as "documents" or "text") managed in an IRS. In our modern times there have been efforts for the integration of DBMS and IRS [12, 29, 30, 38].

Because document databases are huge in size IRSs normally perform retrievals on document representatives. Documents can be represented in various forms an example of which is the vector space model. In the vector space model [36] documents are represented by a document by term matrix, which is referred to as the D matrix. In this matrix each row is a vector that describes a document by means of its elements which are index terms, or terms for brevity. The D matrix can be generated manually or by automatic indexing [30, 36, 46]. The individual entries of D, d_{ij} ($1 \leq i \leq m$, $1 \leq j \leq n$), indicate the importance of term-j (t_j) in document-i (d_i), where t_j is a member of the indexing vocabulary T, $T= \{t_1, t_2, ..., t_n\}$. If indexing is binary d_{ij} will be either 0 or 1 indicating 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. In other words, we are modelling a document by a vector in an n-dimensional space defined by n terms. These descriptions are clustered to narrow down search for retrieval. Documents can also be represented by more traditional ways such as inverted files constructed for terms. Alternatively, we can map, by hashing, documents into what is called signature files [30].

These document representatives are stored in secondary storage, using a structure that facilitates query processing [30, 36, 46, 49]. Query processing retains documents that are relevant to the user request. Normally, relevant documents are determined by use of a similarity function since, unlike DBMS, exact match techniques are not suitable for IRS. In the vector space model [36] the relatedness or similarity of a query to documents is determined by a matching function such as the cosine function given in Eq. (1.1).

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

In this expression X and Y are vectors of length n, Cosine (X, Y) is the cosine of the angle formed by the vectors X and Y in the n-dimensional space and therefore gives a value between zero and one. One vector represents the query and the other the document being compared. Documents that are found relevant (i.e., most similar with cosine value closer to one) to the user query according to the matching function are presented to the user. For example, if Eq. (1.1.) is used as the matching function, documents with a similarity (i.e., cosine coefficient) value greater than a threshold will be retrieved by the system. The actual relevance of the document is then decided by the user. This decision depends on various factors such as user background, type of document, depth and style of document, author, etc.

In Information Retrieval (IR) we can theoretically search the documents by brute force, that is, by full search (FS) which involves comparison of a query vector with the vectors of individual documents representing the database. Obviously, this is inefficient unless the database is too small. One way to increase efficiency of the FS is the clustering of documents via their representatives. In the context of IR, a "cluster" indicates a homogeneous group of documents that are more strongly associated with each other than those in different groups. The process of forming the groups is referred to as clustering. It has been observed that "closely associated documents tend to be relevant to the same request" [46] and this justifies clustering of documents in a database. In a 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 the selected clusters. CBR leads to efficient search spaces in comparison with FS [30]. To further increase the efficiency of CBR, the documents of the same cluster can be put into close proximity within a disk medium to minimize I/O delays[17, 33]. In addition to being efficient a clustering system must be effective in the sense of meeting user needs. In this article we will introduce a new clustering methodology based on the Cover Coefficient (CC) concept. In this regard we will first present CC concept along with the various concepts and methodologies based on it. It will be seen that, in contrast to most other clustering methods, 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, and the indexing-clustering relationships indicated by it are introduced in Section 3. The CC-based Clustering Methodology (C^3M) is one of them and will be introduced here. In the fourth and last section, we will cover our experimental design and evaluation. The experiments were designed to validate the indexing-clustering relationships indicated by the CC concept, to evaluate performances of C^3M and the CC-based D matrix optimization methodology.

2. OVERVIEW OF CLUSTERING ALGORITHMS

Clustering or cluster analysis has a wide spread use in various disciplines [1]. The vast amount of literature on cluster analysis [1, 18, 20, 23, 39] supports this fact. To see the diversity of the subject the reader may refer to [18] which cites more than three hundred publications (over two hundred and fifty articles from seventyseven journals, forty books, and eighteen reports).

At first sight one may think that an expert of a field would be able to judge clustering of the observations at hand by enumarating all possibilities and simply choosing the ones which look the best. However, the problem is not that simple. The number of ways clustering m observations into n_c nonempty subsets is a Stirling number of second type and 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 x 10^{13}. If we do not know the number of clusters n_c the number of possibilities becomes the sum of Stirling numbers [1]. For m= 25

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

This is a very large number which indicates that all cluster possibilities cannot be described with an expression having a length bounded by a polynomial function of the input length. It is shown that the optimum solution of the clustering problem is NP-complete [28, pp. 161-167; 45] which explains the existence of various heuristics instead.

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

In judging suitability of clustering algorithms 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 clusters as uniform as possible?

(e) Is maintenance of clusters practical and efficient? In other words, is the clustering algorithm able to handle document growth efficiently [9]?

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

The answers for 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 pattern or structure of clusters, as in the following:

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

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

(c) Hierarchical type: Clusters of the lowest level are clusters of documents. The higher level clusters are clusters of clusters.

Another classification of clustering algorithms is by their implementation as in the following:

(a) Single-pass algorithms: In this approach documents are handled only once. The first document will form the first cluster. Then the consecutive documents will be compared with the already formed clusters or more correctly with their centroids. When a document is found close enough it is assigned to the corresponding cluster(s) and then the centroid of the cluster is modified accordingly [35].

(b) Iterative algorithms: A typical approach starts with the assignment of documents into existing initial clusters or seeds. (There should be a way of creating the seeds.) The centroid vectors are then modified. 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: The concepts like "single link", "average link", and "maximally complete graph" are used. Here, a link means a similarity value, greater than or equal to a threshold, between any two documents. In the single link method, the members of a cluster are connected to each other by at least one link. In the average link, and maximally complete graph (clique) the number of links of a member to the other members in the same cluster is expected to be greater than or equal to a minimum number, and all the members should be connected to each other, respectively. By changing the similarity threshold from higher to lower values one may construct a hierarchical scheme called dendrogram [46].

2.1 Partitioning Type Clustering Algorithms

C^3M clustering algorithm presented in this article is of partitioning type. We will, therefore, analyze partitioning type clustering algorithms in detail. As can be understood from the name a partitioning type clustering algorithm generates a partition P, where

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

The set P can be formed in different ways. For example, a "suitable" threshold value used with the single link, average link, or maximally complete graph concept can lead to an acceptable partitioning. However, it is very hard, if not impossible, to estimate the "suitable" similarity threshold value that will lead to an acceptable partition.

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 the 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 "the 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 the member of the same cluster, i.e.,

$$d_i \neq d_i \text{ and } d_i \; R_M \; d_i \text{ ---> } d_i \in (D - D_s) \oplus d_i \in (D - D_s)$$

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 that of the earlier studies, a non-exhaustive list of known seed selection methods are listed below [1, 15, 30, 36, 47], 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.

(c) Generate the initial cluster seeds by a random process.

(d) Divide the database into partitions, then form the partition centroids as the seed points.

(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.

(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 initial seed. In the above process, terms containing too many or too few documents may be eliminated.

2.2 Unresolved Problems of Clustering Algorithms

The clustering research in IR has produced various methodologies [13-16, 29, 33, 35, 44-47]. However, as discussed in [19] 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 in $O(m^2)$ for m documents in the database.

(b) Most graph theoretical based algorithms depend on determination of a similarity threshold to generate cluster links which 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 [32].

(c) Any attempts of reducing $O(m^2)$ time complexity, such as using inverted lists, introduce additional complexities if clustering and indexing are to be done properly (e.g., exhaustive indexing) [22].

(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 dependence of clustering on the order of documents (order dependence) or nonuniform distribution of documents in resulting clusters.

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

(f) Clusters can be generated independent of the document description matrix D. For example, the relevant documents of a query can be made a cluster. However, such an approach may lead to clusters containing documents with unsimilar document description vectors. This may be detrimental for processing of queries in a general environment [16, 44].

3. CONCEPTS OF C^3M

Cover Coefficient, CC, is the base concept of C^3M clustering. The CC concept can be used for the following purposes.

(a) to identify relationships among documents (and terms) of a database by C (in the case of terms, C') matrix;

(b) to determine number of clusters within a document database;

(c) to select the seed documents using a new concept called cluster seed power;

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

(e) to predict the relationships between clustering and indexing;

(f) to calculate the importance of terms and documents within a database (these are referred to as term discrimination value and document significance value, respectively);

(g) to increase the effectiveness of IR optimize the D matrix by use of the term discrimination and document significance values.

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]. In this article a probabilistic interpretation for this concept will be introduced.

Definition: Given is a D matrix representing the database $\{d_1, d_2, \ldots, d_m\}$ using 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} $(1 \leq i, j \leq m)$, indicate the probability of selecting any term of d_i from d_j.

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

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

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

We will introduce the CC concept using binary indexing. Subsequently, however, its behaviour will be analyzed also with respect to 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 (number of terms common to both d_i and d_j) / (total number of document frequency of terms of 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 double-stage experiment [24, p. 94] and each row of the C matrix summarizes the results of a double-stage experiment. Let s_{ik} indicate the event of selecting t_k from d_i in the first stage, and s'_{jk} indicate the event of selecting d_j from t_k in the second stage. (The problem can be equivalent to the following: Suppose we have many urns -terms of d_i-, each containing many documents -notice that individual terms of d_i appear in many different documents-. An urn is chosen at random, and from it a document is drawn at random. What is the probability of getting d_j.) 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 $P(s_{ik}) \times P(s'_{jk})$ [24]. 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 (including d_i itself) we can represent the D matrix with respect to the double-stage probability model, as shown in Table 1.

Table 1. Double-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 shows the hierarhical interpretation of this model. In Figure 1, we start from d_i, and end up at one of the documents of the database. In reaching from d_i to d_j ($1 \le j \le m$) there are n possible ways ($s_{i1}, s_{i2}, \ldots, s_{in}$). Choose one of them, e.g. s_{ik}, then, 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 $s_{ik} \times s'_{jk}$.

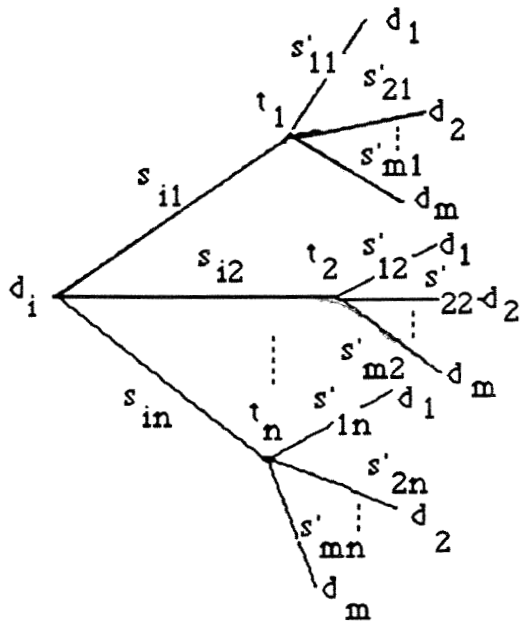

Figure 1. Hierarchical representation of the double-stage probability model for d_i of the D matrix

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.

$$D=\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

To illustrate the concept let us follow the calculation of c_{12}. According to the D matrix, d_1 contains three terms $\{t_1, t_2, t_5\}$ and there are a total of eight occurrences of these terms and three of this total appear 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 is in d_2, the probability of getting a term of d_1 form d_2 is 3/8= 0.375 [24, p.92]. As we have stated previously, however, this is not true. This is because one has no right to treat eight terms as equally likely. According to the double-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. In 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 in the first stage, then the probability of selecting d_2 in 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. In 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 ($\{t_3, t_4, t_6\}$ do not appear in d_1). Pictorial representation of this experiment is provided in Figure 2 (zero s_{1j} or s'_{ij} values for $1 \leq i \leq 5$, and $1 \leq j \leq 6$) are not shown in the figure). According to Figure 2,

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

The C matrix is formed from the matrices named S and S', i.e., $C= S \times S'^{T}$ (S'^{T} indicates the transpose of matrix S'), where s_{ik} and s'_{jk} were defined previously. s_{ik} and s'_{jk} indicate the probability of selecting t_k from d_i, and probability of selecting d_j "from" t_k, respectively. (In s'_{jk} case we consider the term definition vector, i.e., the jth column of the D matrix.) Accordingly, the entries of the C matrix are defined as follows, by using the definition of S and S' matrices,

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

where $s'^{T}_{kj} = s'_{jk}$.

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

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

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

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

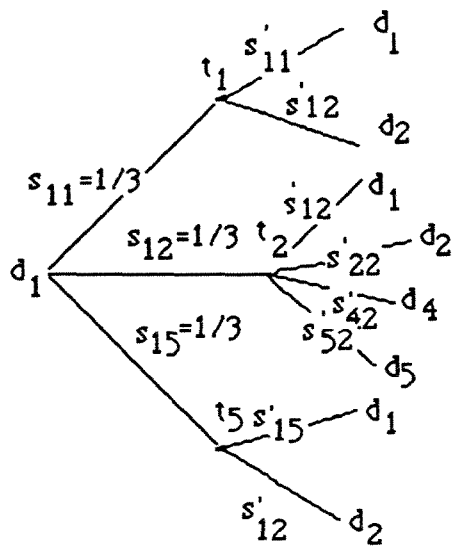

Figure 2. Hierarchical representation of the double-stage probability model for d_1 of the example D matrix

3.2 Properties of the C Matrix

The following properties hold for the C matrix:

(a) For $i \neq j \quad 0 \le c_{ij} \le c_{ii} \ge 0$ (i.e., the values of the off-diagonal entries vary between 0 and c_{ii}, the value of c_{ii} is always being 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 d_i is unique, i.e., 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 $c_{ij} \neq c_{ji}$.

The proofs of these properties are given in [3, 6]. Here we present the interpretation of these properties.

Property (a): $c_{ij} \geq 0$ means that it may ($c_{ij} > 0$) or may not ($c_{ij} = 0$) be possible to select the terms of d_i from d_j. $c_{ij} \leq c_{ii}$ is obvious, when one tries to select the terms of d_i from the database, d_i itself will have higher chance than any other document including d_j. 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 double-stage experiment, and the sum of all the probabilities will be the universal space with the sum being one.

Property (c): means that if some of the terms of d_i appear in the other documents, d_j, then those 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 (since 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.

Another property of the C matrix is the following. If a D matrix is mapped into a C matrix, then μxD, where "μ" is a positive real number, will be mapped into the same C matrix. This comes with the definition of the c_{ij} entries, Eq. (3.2). Therefore, if we shift the document vectors with the same amount in the vactor space, this is not going to affect the C matrix.

To have better intuition of 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 as shown in Figure 3. The relationships between d_i and d_j in terms of the C matrix entries (i.e., in terms of c_{ii}, c_{ij}, c_{jj}, and c_{ji}) are described in the following. (a) d_i and d_j are identical: i.e.,

$$c_{ik} = c_{jk}, \; c_{ki} = c_{kj} \text{ for } 1 \leq k \leq m$$

(b) d_i and d_j have some common terms and documents do not contain each other: i.e.,

$$c_{ii} > c_{ij}, \; c_{jj} > c_{ji}, \; c_{ii} \neq c_{jj}, \; c_{ij} \neq c_{ji}$$

(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.,

$$c_{ii} > c_{ij}, \; c_{jj} > c_{ji}, \; c_{ij} = c_{ji} = 0$$

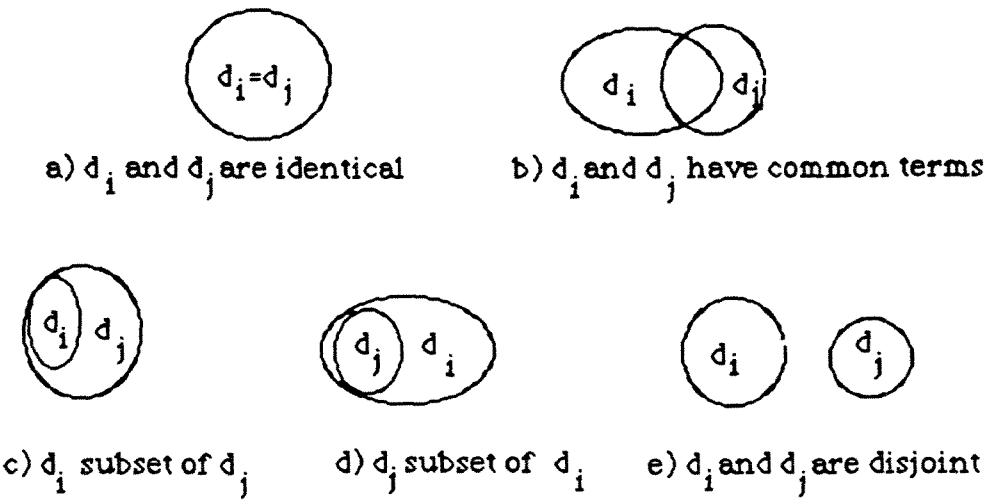

Figure 3-The possible relationships between two document vectors d_i and d_j

From the properties of the C matrix and the CC relationships between two document vectors (refer to Figure 2), c_{ij} is given the following meaning:

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

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

(a) Identical documents: uniqueness and extent with which documents cover each other are identical. Furthermore, the extent with which these documents covered by the other documents is also identical ($c_{ik} = c_{jk}$, where $1 \leq k \leq m$). Similarly, the extent with which these documents cover the other documents ($c_{ki} = c_{kj}$, where $1 \leq k \leq m$) is identical.

(b) Overlapping documents: each document will cover itself more than the 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) Identical with 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 not be equal to 1.

In a D matrix, if d_i $(1 \leq i \leq m)$ is relatively more unique (i.e., if d_i contains terms which appear in less number of documents), then c_{ii} will take higher values. Because of this, c_{ii} is called uniquness or decoupling coefficient, δ_i, of d_i. If none of the terms of d_i is contained in any other document, then $\delta_i = 1$ (i.e., d_i is completely unique, or "decoupled" from the other documents in the database). In terms of the double-stage experimentation, it is not possible to select a term of d_i from the other documents of the database. Contrary would mean nonzero c_{ij} which in turn implies nondisjointness of 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, ψ_i , of d_i. From the properties of the C matrix $\psi_i = 1 - \delta_i$. The ranges for δ_i and ψ_i are $0 < \delta_i \leq 1, 0 \leq \psi_i < 1$ for $1 \leq i \leq m$.

By using the individual δ_i and ψ_i values, the overall or average decoupling and coupling coefficients δ and ψ, respectively, of 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 can then be obtained with 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 cefficient 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 seen in 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 n x n for the index terms. C' has the same properties of C. This time, however, each row of the C' matrix summarizes the results of a double-stage experiment. C' is defined as the product of S'^{T} and S matrices the elements of which are obtained as

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

where $s'^{T}_{ik} = s'_{ki}$.

Notice that in the case of C', the stages of the double-stage experiment are interchanged with respect to the order for the C matrix. By using the definitions of the S and S' matrices

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

The concepts of decoupling and coupling coefficients of t_j, δ'_j and ψ'_j are the counterparts of the same concepts defined for documents. Hence $\delta'_j = c'_{jj}$, and $\psi'_j = 1 - \delta'_j$. The concepts of overall or average coupling and decoupling are also valid for terms and represented by δ' and ψ', 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 decouplind and coupling coefficients of t_1 are $\delta'_1 = c'_{11} = 0.292$ and $\psi'_1 = 1 - \delta'_1 = 0.708$. The overall decoupling, δ', and coupling, ψ', 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., $d_{i1} = d_{i5}$ ($1 \leq i \leq 5$). Accordingly, from the properties of the C' matrix $c'_{1j} = c'_{5j}$ and $c'_{j1} = c'_{j5}$ for $1 \leq j \leq 5$ 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).

3.4 Number of Clusters Hypothesis

The determination of the number of clusters for a database has been an unresolved problem of clustering theory[19]. With the introduction of the CC concept it has been possible to assert the following hypothesis.

Hypothesis. The number of clusters within a database, n_c, should be high if the individual documents are dissimilar and low otherwise. Furthermore, n_c can be obtained as

$$n_c = \sum_{i=1}^{m} \delta_i = \delta \times m \tag{3.8}$$

Similarly, the number of clusters implied by the C' matrix, n'_c, would be

$$n'_c = \sum_{j=1}^{n} \delta'_j = \delta' \times n \tag{3.9}$$

It is known that classifying documents imply classifying terms and vice versa [40]. This is intuitively obvious that as we classify documents, the clustering process will implicitly group the terms which are most frequently used by the members of a cluster. The reverse is also true. That is, as we classify terms this will also imply a classification among documents. The idea of classifying terms using term clusters to form document clusters has been used in the literature by various researchers [14]. This leads to the fact that n_c and n'_c should be identical and this identity has been proven [3, 6] to hold.

The equations (3.8) and (3.9) are consistent with the statement of the number of clusters hypothesis. This is because a database with similar documents will have a low δ (decoupling) whereas a database with dissimilar documents will have a high δ. Let us show the conceptual validity of Eq.s 3.8 and 3.9 with the following propositions and corollaries.

Proposition-1: $n_c = 1$ if all documents of D are identical.

Proof: From the properties of the D matrix $d_{ik} > 0$ for $1 \leq i \leq m$, $1 \leq k \leq n$. 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 $1 \leq i, j \leq m$ and $1 \leq k \leq n$. For simplicity let us use d_k for d_{ik} and d_{jk}, $1 \leq k \leq n$. Then from Eq. (3.2)

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

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 $1 \leq i \leq m$. This implies that n_c which is the summation of all δ_i is equal to m x 1/m=1. □

Proposition-2: For a binary D matrix the minimum value of c_{ii} (δ_i) $1 \leq i \leq m$ is 1 / m.

Proof: If $c_{ii} < 1/m$ then this means that $c_{ij} < 1/m$ for $i \neq j$ (since $c_{ii} \geq c_{ij}$), 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 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 $c_{jj} > c_{ji}$. Assume that $c_{ii} > c_{ij}$. Since the minimum value of c_{ii} is1 / m (notice that when $c_{ii} = 1 / m$, then $c_{ij} = 1 / m$ $i \neq j$, or else row sum will be greater than 1, which contradicts the property (b) of the C matrix) then in order to have the inequality to hold $c_{ii} > 1 / m$. If we have at least one c_{ii} value greater than 1/m it implies that n_c is greater 1. Since the lowest value which can be assumed by c_{ii} is 1/m. □

Corollary-2: The value range of n_c is $1 \leq n_c \leq \min(m, n)$.

Proof: 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 with 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 $n_c \leq \min(m, n)$ is logical, since $n_c \geq \min(m, n)$ would violate the partitioning property of C^3M that will be introduced later. For instance if $n_c > m$ then some documents must be the member of more than one cluster, which contradicts the definition of partition, i.e., nonoverlapping clusters.

After determining n_c, it is easy to estimate the average number of documents d_c within a cluster as $d_c = m / n_c = 1 / \delta$. The concept of decoupling coefficient implies that increase in δ or in individual δ_i ($1 \leq i \leq m$) will increase the number of clusters ($n_c = \delta \times m$) and hence decrease the average size of clusters. The opposite is also true.The relationship between d_c and δ is shown in the logarithmic scale in Figure 4. The value range of d_c is $m/\min(m, n) \leq d_c \leq m$.

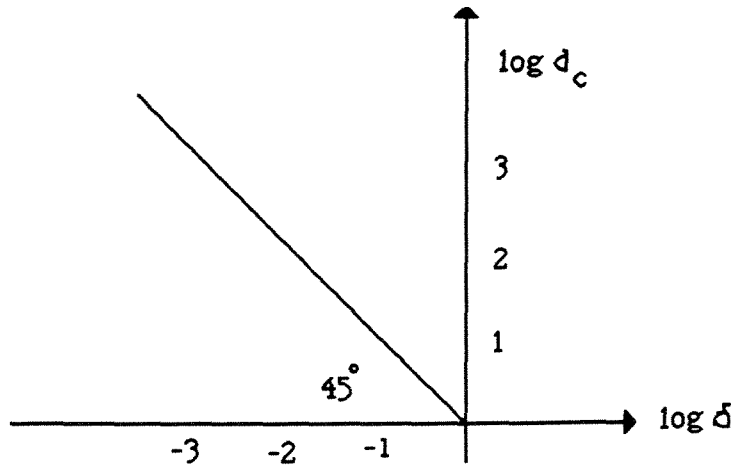

Figure 4. The relationship between d_c and δ in the logarithmic scale, $\log d_c = |\log \delta|$ $(0 < \delta \leq 1)$

Now, let us compute the number of clusters that will result from the example D matrix:

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

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

If we use the C' matrix n'_c would come out as 1.946 which agrees with n_c (the difference between n_c and n'_c 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 $\min(d_{jk}) = d_{ik}$ and if $d_{jk} > d_{ik}$ for at least one k $(1 \leq k \leq n)$ then $c_{ij} > c_{ii}$. Hence for a weighted D matrix $\delta_i < 1/m$ can be observed. However, $1 \leq n_c \leq \min(m, n)$ is still valid since the proofs of proposition-1 and corollary-2 are also valid for the weighted case. The identity $\min(n_c) = 1$ for the weighted case is obvious.

3.5 Cluster Seed Power

The C^3M is seed oriented, i.e., n_c number of documents are selected as cluster seeds and non-seed documents are concentrated 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

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

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

The Eqs. (3.10) and (3.11) pertain to a binary and weighted D matrix, respectively. In these equations δ_i provides the separation of clusters (inter cluster dispersion), ψ_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 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 over estimation of the weights of some terms in d_i . By such an approach a term with a high weight, but with a skewed frequency (i.e., a very frequent or very rare term) will not contribute much to the summation, i.e., the seed power of d_i.

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

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

[a] *Calculate the 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 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, since d_i R_e d_j and d_j R_e d_i imply that $c_{ii} = c_{jj}$, $c_{ii} = c_{ij}$, $c_{jj} = c_{ji}$, hold for both directions by changing the order of the operands at both sides of the equalities, therefore, R_e is symmetric.

(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 $c_{jj} = c_{kk}$, $c_{jj} = c_{jk}$, $c_{kk} = c_{kj}$. These equalities also imply that $c_{ii} = c_{kk}$, $c_{ii} = c_{ik}$, $c_{kk} = c_{ki}$. Therefore, R_e is transitive.

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. To eliminate the false cluster seeds, it is necessary to compare each cluster seed with the next possible cluster seed. This assumes that the cluster seed powers are sorted in descending order, the first seed is compared with the second seed and so on. It will be enough to compare the cluster seed under consideration with the next (lower) cluster seed, since they will be in consecutive order due to their close similarity. If the seeds (documents) 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., for example absolute $(c_{ii} - c_{jj}) < \varepsilon$, where ε is a small positive number chosen as a threshold.

In various experiments [26] it has been observed that documents with medium number of terms are selected as cluster seeds (i.e., special general or documents that are defined by too little or too many terms are not selected as cluster seeds). This is what is expected from a seed selection methodology since general or special documents are not 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 by using 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 computes in 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 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.*

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, for the assignment of documents to cluster seeds a similarity coefficient is used. After all documents are assigned to the seeds, the cluster seeds are replaced by cluster centroids. This repetitive assignment is performed until cluster definitions reach stability. Numerous experiments showed that the computationally efficient single-pass version generates clusters compatible with those of the multi-pass version. The compatibility of the generated clusters is valid in both binary [4, 6] and weighted [26] D matrices.

Now, consider the construction of clusters for the example D matrix. In the database $D_o = \{d_1, d_3, d_4\}$ is the set of documents to be clustered and $D_s = \{d_2, d_5\}$ is the set of seed clusters. To construct the clusters we need only to calculate c_{ij}s where $d_i \in D_o$ and $d_j \in D_s$. For example, for 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 $C_2 = \{d_3, d_4, d_5\}$.

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

(a) It has been experimentally shown [5, 6, 26] 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 will lead to small changes in the C matrix.

(b) The algorithm is independent of the order of documents. This is because the coupling between two documents is not affected by the place of the respective documents in the D matrix. Accordingly, the algorithm generates a well defined clustering pattern, i.e, it produces unique classification.

(c) Implementation of the algorithm requires very small memory for the data structures. αs and βs require m and n memory locations, respectively. m memory locations are also needed for the diagonal entries of the C matrix and to calculate seed powers. In the case of weighted indexing we need to calculate

the diagonal entries of the C' matrix, requiring n memory entries (they are required for seed power calculation). After determining cluster seeds all we need are αs and βs, i.e., we can free the memory locations used for the other data structures. Assignment of documents is done one by one and in this we consider only the non-seed documents. Therefore, we calculate only n_c x (m - n_c) many entries of the C matrix. On the other hand, a graph theoretical approach to clustering would require calculation of similarity coefficients and this requires $(m^2-m)/2$ memory entries plus the cost of cluster generation.

(d) The algorithm distributes documents uniformly among clusters, in other words it does not create a few "fat" clusters and a lot of singletons (i.e., clusters containing only one document), a classical problem encountered in clustering.

The average complexity of C^3M is $O(m^2/\log m)$ and it is shown that [3, 6] the worst case behavior of the algorithm would not be worse than the average case. This complexity compares favorably with the complexity of the other clustering algorithms [33].

3.7 Indexing-Clustering Relationships Obtainable from the CC Concept

The CC concept indicates some relationships between indexing and clustering. In this section we will show the analytical derivation of these relationships using binary indexing. In Section 4.2 it will be experimentally shown that these relationships are observable using either binary or weighted indexing.

For the derivation of the relationships consider Eqs. (3.2) and (3.8) for n_c :

$$n_c = \sum_{i=1}^{m} \delta_i = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij}^2 \cdot \alpha_i \cdot \beta_j \tag{3.12}$$

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

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

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_{di} for document d_i and term generality t_{gj} for term t_j [27]. With these definitions Eq. (3.13) becomes

$$n_c = \sum_{i=1}^{m} \sum_{i=1}^{n} d_{ij} \cdot \left[x_{di} \cdot t_{gj} \right]^{-1} \tag{3.14}$$

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

$$x_d = \sum_{i=1}^{m} x_{di} \; / \; m, \quad t_g = \sum_{i=1}^{n} t_{gj} \; / \; n \tag{3.15}$$

We can approximate $x_{di} \cdot t_{gj}$ with $x_d \cdot t_g$ and rewrite (3.14) as follows:

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

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

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

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

Using d_c and d'_c to indicate the average size of a document 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 and term cluster size, 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., $1 \leq n_c \leq \min(m, n)$ (see corollary-2 of Section 3.4). To show this consider Eq. (3.17), i.e., $n_c = t / (m \cdot n)$ along with the possible max(t) and min(t) values, respectively. Obviously max(t) = m . n which can be observed only if all documents of the database are identical. On the other hand min(t)= max(m, n) holds if each term is assigned to only one document and the document is described by more than one term (i.e., n > m: consider 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: consider Figure 5.b). Accordingly, the minimum value of n_c will be observed if we have the maximum value of t. Hence:

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

Similarly, the maximum value of n_c will be observed if we have the minimum value for t:

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

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

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 matrices D_1 and D_2 shown in Figure 5, Eqs. (3.8) and (3.9) indicate that $n_c = n'_c = 2$. (In Figure 5.a, documents are unique, on the other hand in Figure 5.b the terms are unique.) For D_1, document clusters are $\{d_1\}$ and $\{d_2\}$; 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 the following will be obtained

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

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

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

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

In this section we have derived the indexing-clustering relationships (Eqs. (3.16) through (3.19)) indicated by the CC concept. To do this we have used 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 the noise effect of the weights.

It goes without saying that the indexing-clustering relationships are very valuable for practical purposes. They can be used to control the number (or equivalently the size) of individual clusters so that the cluster sizes can be set according to the user requirements. In general, a user can be either recall or precision sensitive and thus the indexing policy can be tuned in such a way that higher user satisfaction can be obtained. (Recall and precision are, respectively, the proportion of relevant documents that are retrieved and proportion of retrieved documents that are relevant.) Furthermore, we can vary n_c

1) to accommodate physical storage constraints;
2) to control the computational requirements of C^3M.

3.8 Use of CC Concept for D Matrix Optimization

3.8.1 Term Discrimination Value Calculation

In IR, an indexing concept called Term Discrimination Value (TDV) is used to optimize representation of documents by index terms (i.e., D matrix) to increase the retrieval performance of the retrieval system [33, 34, 36, 37]. A TDV indicates the effect of an index term on the distinquishability (or separation) of documents from each other. TDVs of individual terms can be calculated by looking at the average similarity among documents, Q, which is also referred to as document space density [36] and given by

$$Q = 2 \times \left[m \times (m-1) \right]^{-1} \times \sum_{i=1}^{m-1} \sum_{i=i+1}^{m} s(d_i, d_j) \qquad 0 \le Q \le 1, \text{ for } 0 \le s \le 1$$

where $s(d_i, d_j)$ and Q_h indicate similarity between document pairs d_i and d_j (calculated by an expression such as in Eq. (1.1)) and the document space density after the deletion of term t_h from the indexing vocabulary, respectively. If the assignment of t_h makes the documents more separated from each other, then this assignment will decrease the document space

density, hence $Q < Q_h$. In the vice versa, i.e., if the assignment of t_h makes the documents more closer to each other, then its assignment will increase the document space density and hence $Q > Q_h$. Accordingly, TDV of t_h is calculated as follows [36]:

$$TDV_h = Q_h - Q \tag{3.20}$$

$(Q_h - Q)$, i.e., TDV_h, will be greater than zero and less than zero if the assignment of t_h makes documents more separated from each other, and closer to each other, respectively. A term t_h with no significance will have TDV ≈ 0 and hence it is referred to as an indifferent discriminator. Higher separation of documents helps to distinguish the relevant and irrelevant documents from each other in retrievals based on user queries[36]. Therefore, for an effective IR we must have an indexing system whose terms have high, positive TDVs. Such terms are referred to as good discriminators.

We can exploit the CC concept also in the computation of TDVs [8]. If we consider the notions of document (term) coupling and decoupling we can easily realize that the concepts of document space density (Q) and average decoupling of documents (δ or n_c) are inverse to each other. (Even though coupling is the direct counterpart, use of decoupling is computationally more convenient.) Table 2 shows the interpretation of the related quantities with respect to TDV_h (where δ and δ_h are the average decoupling of documents before and after the deletion of t_h).

Table 2. Effects of type of index term(t_h) on the values of Q, δ and (n_c).

	Quantity		
Type of t_h	Q vs Q_h	δ vs δ_h	n_c vs n_{ch}
Good discriminator ($TDV_h > 0$)	$Q < Q_h$	$\delta > \delta_h$	$n_c > n_{ch}$
Indif. discriminator ($TDV_h \approx 0$)	$Q \approx Q_h$	$\delta \approx \delta_h$	$n_c \approx n_{ch}$
Poor discriminator ($TDV_h < 0$)	$Q > Q_h$	$\delta < \delta_h$	$n_c < n_{ch}$

TDV_h of t_h $(1 \le h \le n)$ is defined as the difference:

$$TDV_h = n_c - n_{ch} \tag{3.21}$$

Eq. (3.21) uses the same approach as Eq. (3.20). However, a deleted term will have a reverse effect on n_c with respect to Q as can be observed from the respective expressions and Table 2. By Eq. (3.21) good, poor, and indifferent discriminators will, respectively, have a TDV of positive, negative, or approximately zero values.

To implement Eq. (3.21) we need n_c and n_{ch}. These are provided by the decoupling coefficients as follows:

$$n_c = \sum_{i=1}^{m} \delta_i = \sum_{i=1}^{m} \alpha_i \times (d_{i1}^2 \times \beta_1 + d_{i2}^2 \times \beta_2 + \ldots + d_{in}^2 \times \beta_n)$$

$$n_{ch} = \sum_{i=1}^{m} \delta_i^h = \sum_{i=1}^{m} \alpha_i^h \times \left[\begin{array}{l} d_{i1}^2 \times \beta_1 + d_{i2}^2 \times \beta_2 + \ldots + d_{i,h-1}^2 \times \beta_{h-1} + d_{i,h+1}^2 \times \beta_{h+1} + \\ + \ldots + d_{in}^2 \times \beta_n \end{array} \right]$$

In the formula of n_{ch}, the superscripts of δ_i^h and α_i^h notate the absence of t_h in d_i ($1 \le i \le m$). Also, α_i^h can be easily defined as

$$\alpha_i^h = \left[\sum_{j=1}^{n} d_{ij} \right]^{-1} = \left[\alpha_i^{-1} - d_{ih} \right]^{-1} \quad j \neq h \tag{3.22}$$

As can be seen, the two expressions for n_c and n_{ch} differ only in the term belonging to t_h. n_{ch} can also be expressed as:

$$n_{ch} = \sum_{i=1}^{m} \alpha_i^h \times \left[\delta_i / \alpha_i - d_{ih}^2 \times \beta_h \right] \tag{3.23}$$

In Eq. (3.23) , the term $-d_{ih}^2 \cdot \beta_h$ eliminates the contribution of t_h on n_c via its individual term generality ($\beta_h = 1/ t_{gh}$). δ_i/α_i eliminates the contribution of t_h to n_{ch} due to its effect on depth of indexing ($\alpha_i = 1/x_{di}$); α_i^h reintroduces the effect of the modified depth of indexing. Notice that not all of the documents contain the deleted term t_h. By using this fact and Eqs. (3.22) and (3.23),

$$TDV_h = \sum_{i=1}^{f_h} \left[\delta_i - \alpha_i^h \times \left[\delta_i / \alpha_i - d_{ih}^2 \times \beta_h \right] \right] \tag{3.24}$$

where $f_h = |D_h|$ and $D_h = \{d_i | \ d_i \in D$ and $d_{ih} \neq 0\}$, i.e., f_h is the document frequency of t_h, i.e., t_{gh}. Obviously, TDV_h is nothing but the change in the number of clusters after the deletion of t_h.

For a binary D matrix, the expression of Eq. (3.24) will take the following simplified form:

$$TDV_h = \sum_{i=1}^{f_h} \alpha_i^h \times \left[\beta_h - \delta_i \right] \tag{3.25}$$

The above simplification comes from the fact that for a binary D matrix

$$d_{ih}^2 = d_{ih} \quad \text{and} \quad \alpha_i^{-1} - \alpha_i^h = 1 \quad \text{if} \quad d_{ih} = 1$$

It should be noticed that the CC approach for TDV calculation yields exact values. On the other hand, approximation techniques are based on centroids rather than the individual document vectors.

The consistency of CC approach for TDV has been tested and compared with other methods [11, 48]. These tests revealed very satisfactory results [42]. Furthermore, the computational cost of the CC based approach is favorably comparable [8] with the other approaches available in the literature [11, 48].

3.8.2 Optimization of Document Representatives

Conventionally TDVs are used for the optimization of document descriptions. Representing a TDV by the ratio of densities Q_j/Q, instead of their difference, as in [34, 36] each index entry in a document description is readjusted as

$$d'_{ij} = TDV_j \times d_{ij} \quad (1 \le i \le m, \ 1 \le j \le n) \tag{3.26}$$

In Eq. (3.26) good, bad, and indifferent terms will have TDVs of greater than one, less than one, and approximately one, respectively. It has been observed that optimizing document representations in this manner helps improve IR performance [36].

We are proposing another concept which we call Document Significance Value (DSV) to be used together with TDV in the optimization of document representation, i.e., the D matrix. As in TDV, deletion of a document may change n_c. Obviously, documents exist *a priori* and cannot be deleted from a database. However, conceptually, we may think of deletion of a document to compute its significance value DSV.

Similar to that of TDV, DSV of d_h is defined as $(n_c - n_{ch})$ where

n_c : number of clusters in D,

n_{ch} : number of clusters in $\{D - d_h\}$.

Computing DSV in a way similar to TDV, but this time using n'_c Eqs. (3.7) and (3.9) instead of n_c, we an obtain the following for DSV of d_h as:

$$DSV_h = \sum_{j=1}^{f_h} \left[\delta'_j - \beta_j^h \times \left[\delta'_j / \beta_j - d_{hj}^2 \times \alpha_h \right] \right] \tag{3.27}$$

where $f_h = |T_h|$ and $T_h = \{t_j \mid t_j \in T \text{ and } d_{hj} \neq 0\}$, i.e., f_h is the number of terms used for the description of d_h, and $\beta_j^h = (\beta_j^{-1} - d_{lj})^{-1}$ (i.e., β_j^h is the reciprocal of the column-j sum of the matrix that excludes d_{hj}).

As in TDV Eq. (3.25), Eq. (3.27) can be rewritten for a binary D matrix as:

$$DSV_h = \sum_{j=1}^{f_h} \beta_j^h \times \left[\alpha_h - \delta'_j \right] \tag{3.28}$$

Then the proposed approach for D matrix optimization uses the adjustment of

$$d'_{ij} = DSV_i \times TDV_j \times d_{ij} \tag{3.29}$$

In the above expression DSV_i and TDV_j are taken as n_c/n_{ci} and n_c/n_{cj}, respectively. In this way DSV_i and TDV_j will assume values of <1, >1, = 1. This approach for the calculation of TDV and DSV does not prohibit the use of equations (3.24), (3.25) and (3.27), (3.28), respectively. Eq.s (3.24) through (3.28) can be used to obtain the change in the number of clusters. The new number of clusters can be calculated by using the old n_c and the change in n_c, which is indicated by Eq.s (3.24), (3.25), (3.27), and (3.28).

The product

$$DSV_i \times TDV_j = (n_c / n_{ci}) \times (n_c / n_{cj}) = n_c^2 / (n_{ci} \times n_{cj}) \tag{3.30}$$

is referred to as the "weight modification factor" for the weight of t_j in d_i and is abbreviated as Δ_{ij}. Accordingly, the D matrix will be redefined as

$$d_{ij} = \Delta_{ij} \times d_{ij} \qquad \text{for } 1 \le i \le m,\ 1 \le j \le n \tag{3.31}$$

The weight modification factor, Δ_{ij}, modifies the weight of n_j in d_i by observing 1) the importance of the term and 2) the significance of the document d_i with respect to the other documents of the database.

Let us illustrate TDV and DSV using the example D matrix.

$$TDV_1 = \alpha_1^1 \times (\beta_1 - \delta_1) + \alpha_2^1 \times (\beta_1 - \delta_2)$$

$$= 1/2 \times (1/2 - 0.417) + 1/3 \times (1/2 - 0.438)$$

$TDV_1 = 0.062$ $\qquad$ $TDV_4 = 0.055$

$TDV_2 = -0.250$ $\qquad$ $TDV_5 = 0.062$

$TDV_3 = 0.104$ $\qquad$ $TDV_6 = -0.035$

$$DSV_1 = \beta_1^1 \times (\alpha_1 - \delta_1') + \beta_2^1 \times (\alpha_1 - \delta_2') + \beta_5^1 \times (\alpha_1 - \delta_5')$$

$$= 1 \times (1/3 - 0.292) + 1/3 \times (1/3 - 0.292) + 1 \times (1/3 - 0.292)$$

$DSV_1 = 0.092$ $\qquad$ $DSV_4 = -0.042$

$DSV_2 = -0.098$ $\qquad$ $TDV_5 = -0.195$

$DSV_3 = 0.236$

For the optimization of the D matrix consider entry d_{53}. The conventional optimization approach (i.e., Eq. (3.26)) would increase d_{53} since $TDV_3 > 0$. However, our proposed optimization (i.e., Eqs. (3.30), (3.31)) observes both the values of DSV_5 and TDV_3. According to Eq. (3.28), d_5 is considered insignificant ($DSV_5 < 0$). Hence Δ_{53} will assume a value greater than one only if the insignificance of d_5 is compansated by TDV_3. Δ_{53} is calculated as follows:

$TDV_j = n_c - n_{cj} \Rightarrow n_{c3} = n_c - TDV_3 = 1.945 - 0.104 = 1.841 \quad (j = 3)$

$DSV_i = n_c - n_{ci} \Rightarrow n_{c5} = n_c - DSV_5 = 1.945 - (-0.195) = 2.14 \quad (i = 5)$

$\Delta_{53} = (1.945\ ) / (1.841 \times 2.14) = 0.960$

This shows that $\Delta_{53} < 1$. In other words, in contrast to the conventional approach our approach, decreases d_{53}. In the experiments section it is shown that the proposed approach is superior to the conventional approach Eq. (3.26).

4. EXPERIMENTAL DESIGN AND EVALUATION

In this section we will present two sets of experiments:

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

(b) IR Experiments:

1) to measure the performance of C^3M ;

2) to evaluate the effectiveness of D matrix optimization by adjusting term and/or document represantations.

Dealing with item (b) involves evaluation of effectiveness as well as efficiency. In an IR environment, efficiency involves issues such as cost, time, and volume of operations (CPU cycles, disk accesses per retrieved document) etc. [36, 46], which fall within the realm of a separate performance study. The goal of this paper is to evaluate the validity and effectiveness of the concepts and/or methodologies that have been introduced so far. In our effectiveness evaluations we will consider the number of relevant documents retrieved at various points during retrieval and effectiveness of C^3M in placing the relevant documents into fewer number of clusters.

4.1 Document Database

The document database used for the experiments contains the collection of papers published in the journal of *Association for Computing Machinery Transactions on Database Systems* (ACM-TODS), in the issues March-1976 through September-1984. The database consists of 214 documents. Each paper in the database (we will call TODS214) contains the title, keywords given by the author(s), and the abstract.

The index of this database is drawn from a set of terms obtained after the texts are cleaned out of the noise words (i.e., stop words) and remaining words stemmed. The details of the stemming algorithm and indexing software can be found in [31]. For a stem in TODS214 database, to qualify as an index term, the stem should appear within a range of frequencies in the documents. After determining an indexing vocabulary, T, we have generated both a binary and a weighted D matrix. For the binary and the weighted case d_{ij} indicates the existence or non-existence of term t_j in d_i and the number of occurrences of t_j in d_i, respectively.

4.2 Experimental Validation of Indexing-Clustering Relationships

Table 3 provides the information pertaining to the generation of D matrices. In Table 3 the frequency pair (min, max) indicates the frequency constraints that establish a stem as an index term. For example, the first row of the table indicates that a stem which appears at least in two and at most forty documents will be selected as a term. For this case the cardinality of T is 1060, which 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 3 will be identified by their frequency constraints, e.g., the D matrix corresponding to the first row of the table will be referred to as $D_{2\text{-}40}$.

The results of the experiments on indexing-clustering relationships are shown in Table 4. The second column of the table gives the estimated number of clusters --refer to Eq. (3.17). n_{cw} and n_{cb} indicate the number of clusters calculated by the CC concept, Eq. (3.8), for the weighted and binary versions of the corresponding D matrix, respectively. The quantities

$$x_{dw} = \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} / m \qquad \text{and} \qquad t_{gw} = \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} / n$$

indicate the weighted depth of indexing and term generality, respectively. Similarly, d_{cw} and d_{cb} indicate the average size of a document cluster for the weighted and binary cases, respectively. The average size of a term cluster is shown by d'_{cw} and d'_{cb} for the respective term clusters. This table also presents other information to observe the indexing-clustering relationships of Eqs. (3.16) through (3.19). The results of the experiments show that the indexing-clustering relationships hold very closely in the case of binary indexing (the reader can compare estimated n_c (second column) versus n_{cb}, t_g versus d_{cb}, and x_d versus d'_{cb}).

Experiments with the weighted D matrices also show that indexing-clustering relationships hold in the case of weighted indexing. However, the weights have slightly perturbed the indexing-clustering relationships. For example, in Table 4, on the average the n_{cw} values are 15.4% higher than the estimated n_c values. Notice that in the binary case the estimated and actual n_c values are identical in six of the nine experiments.

Table 3. Characteristics of the generated D matrices

Frequency			
Min	Max	n	t
2	40	1060	7446
3	40	757	6840
4	40	604	6381
2	30	1045	6916
3	30	742	6310
4	30	589	5851
2	20	988	5537
3	20	685	4931
4	20	532	4472

Table 4. Results of the indexing-clustering relationships experiments

Matrix	$\frac{t}{(214.n)}$	n_{cw}	n_{cb}	t_w	t_g	d_{cw}	d_{cb}	x_{dw}	x_d	d'_{cw}	d'_{cb}
$D_{2\text{-}40}$	31	34	30	10.70	7.02	6.29	7.13	53.02	34.79	31.18	35.33
$D_{3\text{-}40}$	24	27	23	13.90	9.04	7.93	9.30	49.15	31.96	28.04	32.78
$D_{4\text{-}40}$	20	24	20	16.34	10.56	8.92	10.70	46.13	29.82	25.17	30.20
$D_{2\text{-}30}$	32	36	32	10.01	6.62	5.94	6.69	48.90	32.32	29.03	32.66
$D_{3\text{-}30}$	25	29	25	12.99	8.50	7.38	8.56	45.04	29.49	25.59	29.68
$D_{4\text{-}30}$	22	26	21	15.20	9.93	8.23	10.19	42.01	27.34	22.65	28.05
$D_{2\text{-}20}$	38	43	38	8.15	5.60	4.98	5.63	37.64	25.87	22.98	26.00
$D_{3\text{-}20}$	30	35	30	10.55	7.20	6.11	7.13	33.78	23.04	19.57	22.83
$D_{4\text{-}20}$	25	30	25	12.37	8.41	7.13	8.56	30.75	20.90	17.73	21.2

4.3 Retrieval Experiments

4.3.1 Retrieval Performance Evaluation Measures

In CBR, the first step is to select the appropriate clusters, C_s, by using a matching function f_m. As indicated in Section 1, in the IR literature there are many different matching functions [36]. In this study, a CC-based matching function [6, 7, 29], will be used. Basically, this matching function indicates the mutual coupling between query and the individual documents of the collection. It is experimentally observed that IR performance with the CC-based matching function is comparable with the well know matching (or similarity) function which uses Eq. (1.1) [10, 31, 42]. Its compatibility with the Dice matching function [30, 36, 46] has been observed in [7, 31, 42].

C_s is the set of clusters having enough correlation (in our case coupling) with the particular query, that is

$$C_s = \{ C_i \mid f_m (C_i , q) > 0 \text{ and } r(C_i) < n_{sm} , 1 \le i \le n_c \}$$

where,

n_{sm} : number of clusters to be expanded (i.e., fully examined)

$f_m (C_i , q)$: correlation of C_i with the query,

$r(C_i)$: rank of C_i in the sorted list (clusters are assigned a rank in the decreasing order of the $f_m (C_i , q)$ values).

After ranking the clusters, the same is done to the D_s documents of the selected n_{sm} clusters:

$$D_s = \{ d \in C \text{ and } C \in C_s \}.$$

The set of documents examined by the user, D_{sr}, is defined as follows:

$$D_{sr} = \{ d \in D_s \mid r(d) \le r \text{ and } f_m(d, q) > 0 \}$$

The members of D_{sr} are the documents coming from C_s and having a rank, r(d), which is less than or equal to the r number of documents that the user wants to examine, and that the matching function must yield a correlation value greater than 0. By increasing r(d) from 1 to r we can obtain different recall, R_c, and precision, P_c, values which are described below.

$$R_c = | D_{sr} \cap D_r | / | D_r |$$

$$P_c = | D_{sr} \cap D_r | / | D_{sr} |$$

where D_r is the set of documents relevant to the query (i.e., relevant documents set, or relevant set). To obtain standard evaluation results, the precision values are given at the predetermined recall levels, i.e., R_i ($1 \le i \le 10$) at 0.1, 0.2, . . ., 1.0. When a computed recall value, R_c, falls in a range between two consecutive recall levels R_i and R_{i+1} ($1 \le i \le 9$) then the precision value P_c corresponding to R_c is taken as the precision at R_i [36, p. 167].

If one replaces the definition of D_{sr} with D (ie., all documents), then the foregoing discussion for recall and precision will be valid for full search, FS.

The maximum recall value (i.e., R_c when r= $|D_s|$ and r=m for CBR and FS, respectively) that can be observed for a query is called the "recall ceiling".

In our CBR experiments, if recall ceiling for a query is greater than zero and less than one, then the precision values P_i of R_i > RC are set to the average precision value of all queries obtained at recall level equal to 1 in FS [14]. If RC= 0 then $P_i = 0$ for $1 \le i \le 10$.

For the similar case in FS, P_i is set to the precision value calculated at RC of the query under processing. If RC is zero, then $P_i = 0$ for $1 \leq i \leq 10$.

After calculating the precision values at R_i ($1 \leq i \leq 10$), then the average system performance is obtained by computing the average precision value P_i at each corresponding recall level by using the precisions of all fifty eight queries used in the experiments. This is done both for FS and CBR.

The foregoing evaluation approach is to obtain the effectiveness when a fixed number of clusters are examined. We will also obtain the changes in recall and precision as we expand more and more clusters. In the experiments section n_{sm} will be varied from 1 to 10 (Tables 8 and 9).

Target Clusters:

In CBR, assuming that ideal conditions hold for document description, query formulation, centroid generation, and matching function we expect to select the clusters that contain only the relevant documents for our query. These clusters will be referred to as "target clusters." If C_i and C_j are two target clusters containing the same number of relevant documents and if $|C_i| < |C_j|$ then C_i will be preferable since its size is smaller. Therefore, the best target cluster for a query is the one that 1) contains the highest number of documents relevant to the query, and at the same time 2) has the smallest cardinality.

In the target cluster experiments we will first expand the best cluster for the query under consideration then calculate recall, precision, and the percent of the database to be expanded. This will be repeated until we retrieve all the target clusters containing all the documents relevant for the query. Then we will calculate average performance for all queries.

In the target cluster experiments (i) For a query with relevant set size of k, the maximum number of target clusters is k; (ii) Smaller cluster size will increase precision; in the extreme case, i.e., each cluster containing only one document, the precision will be one for all target clusters of a query. If cluster size is two for all the clusters, then for any target cluster the minimum precision will be 0.5; (iii) If the relevant set size for a query is one, then the recall will be one upon retrieving only one target cluster. This indicates that if the average relevant set size for all queries is small, then the average number of target clusters to be retrieved will be small; (iv) For a query of relevant set size k the average number of clusters to be retrieved can be approximated as k/d_c, where d_c is the average cluster size.

In our experiments we will also compare the performance of C^3M with random clustering. Comparisons of the performance of a clustering algorithm with that of the random case have been made in previous studies [25, 28, 44, 45]. To perform this comparison we will obtain averages of the following quantities for all the queries: number of clusters to be expanded, percentage of the database to be expanded, and precision when all target clusters are retrieved.

To obtain the random performance we will use the theorem given in [43]. Given are m records (documents) grouped into n_c number of blocks (clusters) where $1 < n_c \leq m$ with each cluster containing m/n_c number of documents. If k documents ($k \leq m - m / n_c$) are randomly selected from the m records (in other

words if the relevant set size of a query is k), then the expected number of blocks with at least one record selected is given by

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

This formula assumes that all blocks (in our case clusters) have the same size. C^3M distributes documents uniformly among clusters [5, 6, 26]. However, as it would be expected, cluster sizes are not identical. To employ the above theorem we may force documents into equally sized clusters. However, such an approach will impose adverse effects on our performance results. To alleviate the problem, in random clustering we will place documents randomly in the clusters of the partition formed by C^3M using the given D matrix in an experiment. To justify this the following corollary of the foregoing theorem is introduced.

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 are randomly selected from m documents, the probability P_j that cluster C_j will be selected is given by

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

Accordingly, in random clustering with varying sizes of clusters and a query with relevant size of k we will have the following:

a) the number of target clusters $= \sum_{j=1}^{n_c} P_j$

b) expected size of the database to be expanded (i.e., the total size of all target clusters) $= \sum_{j=1}^{n_c} |C_j| \times P_j$

c) precision when all k documents are retrieved $= k \times \left[\sum_{j=1}^{n_c} |C_j| \times P_j \right]^{-1}$

4.3.2 Experimental Environment

In the retrieval experiments we used the weighted D matrix as defined in the second row of Table 3. The reason for using a weighted matrix is its generality as compared to a binary matrix.

In cluster base retrieval (CBR) after clusters are formed by C^3M we generate the centroids of the clusters by following the CC approach [4, 6]. This approach for centroid generation will emphasize the terms with higher uniqueness values. The terms which have a uniqueness value (i.e., the diagonal entries of the C' matrix, $\delta'_j = c_{jj}'$) greater than or equal to the average

uniqueness value of the terms, δ', will appear at least in one of the centroid vectors. Weighted centroids are generated making the weight of a term, t_j, appearing in a centroid equal to the total weight of t_j in the member documents divided by the corresponding cluster size. The statistics for the clusters and centroids are provided in Table 5 under the heading "original D". This table shows that the indexing constraints of $D_{3\text{-}40}$ yield an average cluster size of 7.93 which is approximately equal to $\log_2 m$ ($\log_2 m$= 7.74). This is the expected size of a cluster for a database of the size of TODS214 [33]. The table shows similar information also for the optimized D matrix.

Table 5. Statistics for the clusters and the centroids used in the retrieval experiments

Quantity	Original D	Optimized with Δ
Number of Clusters	27	28
Average Number of Documents/Cluster	7.93	7.64
Average Number of Terms Used in Clusters	253.33	244.29
Average Number of Distinct Terms in Clusters	172.37	167.25
Average Number of Terms Used in Centroids	58.00	56.82
Number of Distinct Terms in Centroids	445	447
Proportion of Terms Used in Centroids	0.588	0.591

The query set contained a total of fifty-eight queries. We will refer to this query set as Q58. Thirty nine of these queries are constructed from five textbooks on databases. If a chapter contained two or more references to a TODS214 article the titles and subtitles of that chapter are used as the query text. The documents corresponding to these references are assumed to be relevant to the query. The query set is almost evenly distributed among the five texts. The other set of eighteen queries are taken from [31]. The natural language text of the queries is mapped into a query vector. A query vector Q is obtained from the intersection of Q_s and T where Q_s and T are, respectively, the stems corresponding to the query text and indexing vocabulary. In the experiments, binary query vectors are used and this is reasonable since most terms appear only once in a query text.

In the target cluster experiments we used the query set Q58. To validate the results obtained with respect to Q58 we have augmented the query set in Q58 with another query set containing 110 queries. This query set will be referred to as Q110. Q110 is created by using the citations of the papers published in the journal of ACM-TODS. If a TODS article cites three or more papers which are contained in our TODS214 collection, then it is taken as a query. We are interested only in the relevant set of the individual queries of Q110, i.e., Q110 is used only in target cluster experiments. Q110 queries almost evenly distributed in the year range of 1979 through 1987. The detailed information for the query sets are given in Table 6. Table 6.A indicates that in Q58 there is one query with relevant set size of one while for Q110 the number of queries with relevant set size of one or two is zero. The number of queries with relevant set size of three is eleven and thirtyone for Q58 and Q110. Q168 is the union of Q58 and Q110 respectively. Table 6.B indicates the number of queries that will retrieve all of the relevant documents if we expand sufficient number of clusters. For example, if we expand one target cluster at least one query will be satisfied for Q58, and therefore in Q168 since there is only one query with a relevant set size of one (refer to Table 6.A). The number of

queries with relevant set size of one or two is twelve, hence if we expand two target clusters obviously twelve queries will retrieve all of their relevant documents. (Notice that target clusters contain at least one relevant document for the query under consideration.) An effective clustering algorithm must satisfy high number of queries with less number of target clusters expanded. The entries of Table 6.C are self explanatory.

Table 6 - Characteristics of the queries

A. Relevant set size statistics for the queries

	Query Set	Relevant set size 1	2	3	4	5	6	7	8	9	10	11	12	13	14
Number of Queries	Q58	1	11	11	5	6	5	5	3	5	2	3	1	0	0
	Q110	0	0	31	32	19	16	8	0	2	0	1	0	0	1
	Q168	1	11	42	37	25	21	13	3	7	2	4	1	0	1

B. Number of queries that will be satisfied when h number of clusters expanded

	Query Set	No of Clusters Expanded (h) 1	2	3	4	5	6	7	8	9	10	11	12	13	14
Number of Queries	Q58	1	12	23	28	34	39	44	47	52	54	57	58	-	-
	Q168	1	12	54	91	116	137	150	153	160	162	166	167	167	168

C. Other information for the queries

Information	Query Set Q58	Q110	Q168
No of distinct doc. retrieved	126	134	165
Tota l no of relevant docs.	305	511	816
Avg. no of rel. doc./query	5.259	4.645	4.875

4.3.3 Experimental Results Using the Conventional Measures:

In order to see the effectiveness of CBR, first the average precision values at the recall levels are obtained and the values are compared with those of FS. Then CBR is repeated for the D matrices optimized with respect to Eqs. (3.26) and (3.31). In these experiments n_{sm} (the number of clusters expanded) is three. In a database of the size of TODS214 the user is assumed to be either completely satisfied or dissatisfied after examining about twenty documents. He (she) would either quit using the system afterwards or reformulate a new query for the same request. (Notice that in the experimental environment the average cluster size is 7.93 and n_{sm}= 3 which will provide the user approximately twenty documents to examine.) The results of the CBR experiments are provided in Table 7. The table also shows the percentage difference with respect to FS. As can be seen, CBR with both the ordinary D matrix (designated as CBR) and the optimized D matrix (designated as CBR', and the other primed entries are also associated with optimization) is worse than those in FS. However, CBR' is slightly better than CBR. In these and in the following experiments, it is observed that D matrix optimization with TDV alone (Eq. (3.26)) did not have any improving effect on CBR behavior.

Table 7. Precision vs recall for FS, CBR, and CBR'

Recall	FS	CBR	CBR'
0.1	0.40688	0.34849	0.37247
0.2	0.33010	0.29303	0.31220
0.3	0.26042	0.24369	0.26026
0.4	0.23240	0.23127	0.24208
0.5	0.22745	0.22088	0.22464
0.6	0.19427	0.18498	0.18375
0.7	0.19312	0.17906	0.17887
0.8	0.18265	0.16617	0.16359
0.9	0.17198	0.14917	0.14609
1.0	0.16585	0.14038	0.14609
% differnc. w.r.t. FS:		-8.51	-6.12

Table 8. Recall ceiling vs n_{sm}

n_{sm}	RC	RC'
1	0.27818	0.25963
2	0.35623	0.36094
3	0.40938	0.41675
4	0.46655	0.46464
5	0.48915	0.51019
6	0.53399	0.53762
7	0.55125	0.56136
8	0.57294	0.59158
9	0.58656	0.60364
10	0.59037	0.60058

Table 9. Precision vs n_{sm}

n_{sm}	Precision	Precision'
1	0.28598	0.27306
2	0.21904	0.23345
3	0.21436	0.22571
4	0.19955	0.22049
5	0.19117	0.21437
6	0.18301	0.20229
7	0.18543	0.20518
8	0.19232	0.21626
9	0.18580	0.21173
10	0.16745	0.18840
% improvement:		8.92

Table 8 shows the behavior of CBR and CBR' in terms of recall ceiling (RC) with respect to different n_{sm}. (Average RC for all queries in the case of FS is 0.69.) These experiments showed that RC' values are slightly better than RC values. In other words, the matrix optimization using Eq. (3.31) slightly improves the performance of the system in terms of recall ceiling. However, the improvement is negligible as expected. This is because, D matrix optimization does not introduce anything to improve the performance of the system in terms of recall (e.g., it does not introduce new terms to describe the documents).

In order to see the precision improvement of D matrix optimization we have performed a set of experiments for finding the average precision for various n_{sm}. The results are provided in Table 9. This table shows that our D matrix optimization is effective for precision improvement. The average CBR' precision values are 9.01% higher than those of CBR.

Using Target Clusters:

Table 10 shows the results of the target cluster experiments. The experiments are performed for all D matrices defined in Table 4. Since the results are quite similar for all matrices, we will give only the results for three D matrices with the number of clusters varying from 27 to 43 to show the effect of cluster size on performance. The results of the experiments are given both with Q58 and Q168. As we have indicated earlier, Q58 is a subset of Q168. Table 10 contains the following information: number of target clusters expanded, number of queries satisfied (i.e., the queries that retrieved all of their relevant set), average recall and precision, and the percentage of the database expanded. Let us look at Table 10.A for $D_{3\text{-}40}$ of Q58. We are able to retrieve all of the relevant set for eight queries by expanding only one cluster. The same is valid for twenty-six queries with two target clusters. The reader can compare these values with the values provided in Table 6.B. The comparison indicates that the clustering of C^3M is effective in putting the relevant documents of a query into the same cluster.

The results of Table 10 show that relevant sets of queries are concentrated in a very few clusters. For example in the case of $D_{3\text{-}40}$ of Q58, if we expand one cluster, or on the average 4.9% of the database, recall is 0.537 and precision is 0.267.

For $D_{3\text{-}40}$ C^3M generates 27 clusters and the average cluster size is 7.93 documents. In the other extreme, i.e., with $D_{2\text{-}20}$, there are 43 clusters and the average cluster size is 4.98, which is 63% of the case with $D_{3\text{-}40}$. However, notice that when we expand one target cluster, the recall in the case of $D_{2\text{-}20}$ is 0.430 which is still high. That is, although our cluster size is now 60% of that of $D_{3\text{-}40}$, we are still able to obtain 80% (0.430/0.537= 0.801) of the recall value of $D_{3\text{-}40}$. Also, because the cluster size is smaller with $D_{2\text{-}20}$ the percentage of the database to be expanded drops and precision increases. The results of Table 10 indicate a good performance for C^3M since relevant sets of documents are concentrated in a small number of clusters.

The comparison of the results of C^3M with those of the random case (RnC) is provided in Tables 11.A and 11.B for the query sets of Q58 and Q168, respectively. In this table, the columns "Avg. n_c", "% DB", and "Avg. Pre." indicate, respectively, the average number of target clusters, the percentage of the database expanded, and precision when the relevant set of all queries is retrieved. The improvement due to C^3M is computed as the percentage of performance differential with respect to C^3M.

The results of Table 11.A indicate that C^3M decreases the number of clusters to be expanded in comparison with the random case. The percentage decrease varies between 17% and 29% with the average being 25%. The clusters generated by C^3M are almost uniform (equally sized). This can be seen from the decrease in the percentage of database to be expanded. The decrease in the

Table 10. Retrieval with target clusters

A. For Q58

D-matrix	No of Target Clusters Exp.	No of Qry. Satisfied	Recall	Precision	% Database Expanded
$D_{3\text{-}40}$ (n_c= 27)	1	8	0.537	0.267	4.9
	2	26	0.769	0.222	8.4
	3	36	0.870	0.200	10.8
	4	44	0.929	0.189	12.5
	5	50	0.962	0.182	14.0
$D_{2\text{-}30}$ (n_c= 36)	1	4	0.478	0.316	3.6
	2	22	0.745	0.266	6.5
	3	32	0.858	0.237	8.5
	4	42	0.927	0.224	10.1
	5	52	0.965	0.213	11.2
$D_{2\text{-}20}$ (n_c= 43)	1	2	0.430	0.351	2.7
	2	18	0.709	0.291	5.4
	3	30	0.843	0.269	7.1
	4	41	0.915	0.253	8.5
	5	49	0.952	0.245	9.4

Table 10. Retrieval with target clusters (cont.)

B. For Q168

D-matrix	No of Target Clusters Exp.	No of Qry. Satisfied	Recall	Precision	% Database Expanded
$D_{3\text{-}40}$ (n_c= 27)	1	16	0.516	0.250	4.9
	2	63	0.757	0.205	8.6
	3	106	0.891	0.185	11.7
	4	142	0.956	0.175	13.9
	5	153	0.977	0.171	14.8
$D_{2\text{-}30}$ (n_c= 36)	1	9	0.464	0.287	3.6
	2	50	0.721	0.242	6.6
	3	95	0.872	0.217	9.1
	4	134	0.950	0.205	10.8
	5	155	0.979	0.198	11.7
$D_{2\text{-}20}$ (n_c= 43)	1	4	0.420	0.334	2.7
	2	34	0.681	0.272	5.4
	3	91	0.858	0.245	7.7
	4	130	0.939	0.231	9.2
	5	155	0.971	0.226	10.0

database size to be expanded is approximately equal to the decrease in the number of clusters to be expanded. The increase in precision due to C^3M compared to the random case varies between 33% and 60% with the average being 50%. With query set Q168 the decreases in the number of target clusters and the size of the database to be expanded are 22% and 20%, respectively. The increase in precision with respect to the random case is 42%. In other words, the performance figures for Q168 are worse than those of Q58. This might be due to the diversity of citations in the queries of Q110. In other words, some of

the references cited in an article may be less relevant than others. (Notice that Q110 is constructed using TODS articles.)

The results of C^3M are significantly better than those of the random case. This can be explained by about 20% decrease in the number of clusters to be expanded and about 40% improvement in precision, as compared to the random case. It is our belief that a comparison such as the one provided in Table 11 can be used as a standard benchmark in the evaluation of a clustering algorithm.

Table 11. The comparison of C^3M and random clustering (RnC) when we retrieve all relevant documents for all queries

A. For Q58

D matrix	C^3M Avg n_c	%DB	Avg Pre	RnC Avg n_c	%DB	Avg Pre	% imp. of C^3M n_c	%DB	Avg Pre
D_{2-40}	3.379	13.3	0.206	4.757	18.4	0.129	-29	-28	60
D_{3-40}	3.379	16.0	0.178	4.643	21.9	0.108	-27	-27	65
D_{4-40}	3.293	16.6	0.168	4.605	23.1	0.102	-29	-28	65
D_{2-30}	3.603	12.7	0.209	4.808	16.9	0.142	-25	-25	47
D_{3-30}	3.397	14.6	0.184	4.706	20.0	0.119	-28	-27	55
D_{4-30}	3.586	16.3	0.164	4.683	20.7	0.115	-23	-21	43
D_{2-20}	3.879	11.0	0.237	4.901	14.0	0.172	-21	-21	38
D_{3-20}	3.707	13.1	0.200	4.813	16.7	0.143	-23	-22	40
D_{4-20}	3.931	15.3	0.172	4.755	18.5	0.129	-17	-17	33
Average % improvement of C^3M over RnC:							-25	-24	50

B. For Q168

D matrix	C^3M Avg n_c	%DB	Avg Pre	RnC Avg n_c	%DB	Avg Pre	% imp. of C^3M n_c	%DB	Avg Pre
D_{2-40}	3.286	13.2	0.193	4.466	17.4	0.128	-26	-24	51
D_{3-40}	3.268	16.2	0.167	4.375	20.7	0.107	-25	-22	56
D_{4-40}	3.387	17.2	0.150	4.345	21.9	0.101	-22	-22	49
D_{2-30}	3.482	12.6	0.195	4.507	15.9	0.141	-23	-21	38
D_{3-30}	3.304	14.8	0.171	4.426	18.9	0.118	-25	-22	45
D_{4-30}	3.548	16.2	0.154	4.408	19.6	0.114	-20	-17	35
D_{2-20}	3.738	10.9	0.222	4.580	13.1	0.171	-18	-17	30
D_{3-20}	3.583	13.2	0.188	4.511	15.7	0.142	-21	-16	32
D_{4-20}	3.708	14.2	0.176	4.465	17.4	0.128	-17	-18	38
Average % improvement of C^3M over RnC:							-22	-20	42

The target cluster experiments are repeated for the binary version of the same D matrices. The performance results for the binary case are slightly worse than those of the weighted case. D matrix optimization is also tested for the target clusters. The results obtained are almost the same as those given in

Tables 10 and 11. This is because, the clusters obtained with the ordinary D and optimized D matrices are almost identical. The effect of D matrix the optimization is observed in the document and centroid vectors. The results of FS retrieval with the optimized D matrix are the same as FS retrieval using the original D matrix. This means that optimization improves CBR.

As we indicated earlier the comparisons of the performance of a clustering algorithm with that of the random case can be found in other studies especially in those on record clustering. In these studies the records likely to be retrieved for the same query are put into as few as blocks possible [25, 28, 45]. The study of [45] reported 100% improvement over random clustering for an adaptive clustering algorithm. The same algorithm is also tested in [28] and various improvements over the random case ranging between 5% and 52% with an average of 40% have been reported. For the same data the same study reported an average improvement of 59% over the random case using its own algorithm. Although these results are impressive, some comments are in order. The algorithm given in [45] is affected by the order of query execution. The algorithm given in [28] "deteriorates when there is an increased overlap of records between several queries" as quoted from the same study [28, p. 71]. This might be the case in a real life environment if we consider the 80-20 rule [45]. More importantly, the same set of queries are used [28, 45] for both generation of clusters and goodness test of the generated clusters in terms of percentage decrease in the number of clusters to be expanded. Approaches similar to that of [45] were used also by others [44, 16]. However, the approach used in this study is more generic. Furthermore, use of queries for cluster generation may produce undesirable performance for a different application.

5. CONCLUSION

The NP-completeness of the clustering problem has lead to heuristic approaches. In the study reported here a new clustering methodology called C^3M has been introduced. C^3M relies on its heuristic called the cover coefficient (CC) concept which is used in various aspects of clustering theory. A few examples are 1) the number of clusters within a document database can be determined, 2) a new method for selection of cluster seeds is introduced, and 3) new methods are introduced for the optimization of the document description matrix. The CC concept also relates indexing and clustering analytically. The computational cost of C^3M is comparable with that of the other clustering algorithms available in the literature. C^3M has all the desirable properties of good clustering algorithms. More importantly, with C^3M it is possible to analytically determine the number of clusters and cluster size beforehand.

In the experiments conducted, the indexing-clustering relationships indicated by the CC concept are validated. And, these relationships are strongly observed in the case of binary indexing. In the case of weighted indexing, the relationships are slightly distorted due to the noise effect of the weights.

The retrieval experiments have shown that CBR using C^3M improves the retrieval effectiveness and efficiency of an IR system. The comparisons made with respect to random distribution of documents among clusters have shown that, on the average, C^3M provides 40-50% improvement with respect to the random case in terms of precision. The improvement in terms of the reduction of the search space is 22-25%. This reduction can be significantly extended with hierarchical clustering [29]. The CC concept has been used also in index

vocabulary construction and optimization [31]. Also, advanced optimization techniques such as term discrimination value and document significance value, a concept introduced in this study, have been constructed using CC concepts [8]. The retrieval experiments performed using the document matrices optimized using both of these advanced optimization techniques have shown improvements in retrieval precision, as shown in the paper.

REFERENCES

1. Anderberg, M. R. *Cluster Analysis for Applications.* Academic Press, New Yor, 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 for Information Retrieval Systems 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. "Computation of Term/Document Discrimination Values by Use of the Cover Coefficient Concept." *Journal of the American Society for Information Science.* 38, 3 (May 1987), 171-183.

9. Can, F., Ozkarahan, E. A. "A Dynamic Cluster Maintenance System for Information Retrieval." In *Proceedings of the 10th Annual International ACM-SIGIR Conference* (June 1987), ACM, New York, 123-131.

10. 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.

11. Crawford, R. G. "The Computation of Discrimination Values." *Information Processing Management.* 11 (1975), 249-253.

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

13. 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.

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

15. 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).

16. 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.

17. 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.
18. 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.
19. Everitt, B. S. "Unresolved Problems in Cluster Analysis. *Biometrics.* 35, (1979), 169-181.
20. Everitt, B. S. *Cluster Analysis.* Halsted Press Div. of John Wiley and Sons, New York, 1980.
21. Grauer, R. T., Messier, M. "An Evaluation of Rocchio's Clustering Algorithm." In G. Salton, Ed. *The Smart Retrieval System-Experiments in Automatic Document Processing.* Prentice Hall, Englewood Cliffs NJ, 1971, Chap. 11.
22. 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.
23. Hartigan, J. A. *Clustering Algorithms.* John Wiley and Sons, New York, 1975.
24. Hodges, J. L., Lehmann, E. L. *Basic Concepts of Probability and Statistics.* Holden-Day Inc, San Fransisco, 1964.
25. Jakobsson, M. "Reducing Block Accesses in Inverted Files by Partial Clustering." *Information Systems.* 5 (1980), 1-5.
26. 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.
27. Maron, M. E. "Depth of Indexing." *Journal of the American Society for Information Science.* 19 (1978), 224-228.
28. Omiecinski, E. R. *Algorithms for Record Clustering and File Reorganization.* Ph. D. Dissertation. Field of Computer Science, Northwestern University, Evanston, Ill, 1984.
29. Ozkarahan, E. A., Can, F. "An Integrated Fact/Document Information System for Office Automation." *Information Technology: Research and Development.* 3, 3 (1984), 142-156.
30. Ozkarahan, E. *Database Machines and Database Management.* Prentice Hall, Englewood Cliffs NJ, 1986.
31. 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.
32. 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.
33. Salton, G. *Dynamic Information and Library Processing.* Prentice Hall, Englewood Cliffs NJ, 1975.
34. Salton, G. "A Theory of Indexing." *Regional Conference Series in Applied Mathematics*, No 18, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1975.
35. Salton, G., Wong, A. "Generation and Search of Clustered Files." *ACM Transactions on Database Systems.* 3, 4 (Dec. 1978), 321-346.
36. Salton, G., McGill, M. J. *Introduction to Modern Information Retrieval.* McGraw Hill, New York, 1983.
37. Salton, G. "Another Look at Automatic Text-Retrieval Systems." *Communications of the ACM.* 29, 7 (July 1986), 648-656.

38. 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.
39. Sneath, P. H. A., Sokal, R. R. *Numerical Taxonomy.* W. H. Freeman, San Fransisco, 1973.
40. Sparck, Jones, K. "Collection Properties Influencing Automatic Term Classification Performance." *Information Processing and Management.* 9 (1973), 499-513.
41. Stonebraker, M., et. al. "Document Processing in Relational Database System." *ACM Transactions on Office Information Systems.* 1 (1983), 143-158.
42. 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.
43. Yao, S. B. "Approximating Block Accesses in Database Organizations." *Communications of the ACM.* 20, 4 (April 1977), 260-261.
44. 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.
45. Yu, C. T., Suen, C., Lam, K., Siu, M. K. "Adaptive Record Clustering." *ACM Transactions on Database Systems.* 10 (1985), 180-204.
46. Van Rijsbergen, C. J. *Information Retrieval, 2nd ed.* Butterworths, London, 1979.
47. Willett, P. "Document Clustering Using an Inverted File Approach." *Journal of Information Science.* 2 (1980), 223-231.
48. Willett, P. "An Algorithm for the Calculation of Exact Term Discrimination Values." *Information Processing and Management.* 21, 3(1985), 225-232.
49. 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.