Computer Science and Systems Analysis

Computer Science and Systems Analysis

Technical Reports

Miami University *Year* 1991

Experiments on the Efficiency of Cluster Searches

Fazli Can* David Anderson†

*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-1991-001

Experiments on the Efficiency of Cluster Searches

Fazli Can and David Anderson

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

EXPERIMENTS ON THE EFFICIENCY OF CLUSTER SEARCHES

by

Fazli Can David Anderson
Systems Analysis Department
Miami University
Oxford, Ohio 45056

Working Paper #91-001 03/91

EXPERIMENTS ON THE EFFICIENCY OF CLUSTER SEARCHES

Fazli CAN*

David J. ANDERSON

Department of Systems Analysis

Miami University

Oxford, OH 45056

March 24, 1991

Abstract

The efficiency of various cluster based retrieval (CBR) strategies is analyzed. The possibility of combining CBR and inverted index search (IIS) is investigated. A method for combining the two approaches is proposed and shown to be cost effective in terms of paging and CPU time. The observations prove that the new method is much more efficient than conventional approaches. In the experiments, the effect of the number of selected clusters, centroid length, page size, and matching function is considered. The experiments show that the storage overhead of the new method would be moderately higher than that of IIS. The paper also examines the question: Is it beneficial to combine CBR and full search in terms of effectiveness?

* (513) 529-5950, fc74sanf@miamiu.bitnet

1. INTRODUCTION

The purpose of an information retrieval system (IRS) is to locate the documents which are relevant to the user query. Two well known approaches to query processing are full search (FS) and cluster based retrieval (CBR) [10, 17]. In IR, a cluster contains a homogeneous group of documents that are more strongly associated with each other than with those in other groups. Both FS and CBR use a matching (similarity) function to decide which documents are potentially relevant (i.e., match the query) and should be returned to the user.

For efficiency, FS is usually implemented by inverted index search (IIS). In IIS, associated with each term there is a list of <document no., weight> pairs including each document in which that term appears. The similarity of all database documents is then determined by traversing only query term lists. A typical implementation of CBR first matches queries with cluster representatives, called centroids, then matches the queries with documents in the selected clusters to find those that actually match the query.

As in any other information system, efficiency and effectiveness are the main concerns of an IRS. Efficiency and effectiveness are, respectively, associated with the time/space it takes the system to perform the search and with the quality of retrieval. The concern of this paper is efficiency of IR. More specifically, we compare the efficiency of IIS and CBR based on single level clustering. In the paper we use four different implementations of CBR. In one of them we investigate the possibility of combining IIS and CBR. First we show that the storage overhead of the new method is moderately higher than that of IIS. Later it is shown that the new strategy is cost effective in terms of paging and CPU time and is much more efficient than CBR based on a hierarchical clustering structure [19]. In the efficiency experiments the effect of number of selected clusters, page size, centroid length, and matching function is considered. In the paper we also show the potential benefits of combining CBR and IIS to improve effectiveness.

The paper is organized as follows. Section 2 provides the preliminary information on CBR and presents the description of the new CBR strategy. Section 3 covers the experimental environment in terms of the document database, the clustering algorithm used for the creation of the experimental environment, and the matching functions. Section 4 presents the data structures used for the implementation of IIS and CBR strategies. The storage cost of each method is included and compared. Section 5 defines the query processing efficiency measures and presents the results of the efficiency experiments. The potential benefits of combining FS and CBR in terms of effectiveness is shown in Section 6. The conclusion is given in Section 7.

2. CLUSTER BASED RETRIEVAL (CBR)

In the IR literature there are several publications on clustering and CBR. Further information can be found in [10, 11, 12, 15, 17, 20]. Most clustering research in IR is related to creation of new clustering algorithms and their effectiveness in terms of IR.

In CBR, the system does not necessarily return the documents with highest similarity to the query. This is because, the selected clusters do not necessarily contain the best-matching documents. However, unlike IIS (FS), CBR does not lose the relationships between documents and allows the users to browse through a document database. In CBR the efficiency of retrieval is expected to decrease with increasing number of selected clusters (n_S). A recent study on effectiveness of CBR shows that after a threshold, selecting more clusters does not increase the effectiveness in CBR [3]. In IIS case only the inverted index information of query terms is accessed from the disk medium. As a result the efficiency of IIS decreases with increasing query length [10, p. 293]. Since for each query term a new inverted index list must be processed. CBR selects some of the clusters; therefore, it has the potential of being more efficient than IIS if query vector is long. For example, in automatically generated query environments (e.g., using a seed document) the query length can be long (e.g., 200 terms) [14, 16].

A recent study on efficiency (and effectiveness) is reported in [18, 19]. The paper [19] compares the efficiency of IIS and complete link based hierarchical CBR. The findings of the study indicate that IIS is more efficient in terms of disk space and query processing time since the cluster search requires several centroid vectors. In [19] the number of clusters that are considered in the search depends on the number of documents to be retrieved. It begins by placing the root of the cluster tree into an empty heap. Later, the top of the heap corresponds to the cluster with the current largest similarity with the query. The top of the heap is popped until the heap is empty or sufficient number of documents have been retrieved. In the search, a popped document is retrieved. A popped cluster is replaced by its children that have nonzero similarities with the query. At the end the retrieved documents are presented to the user in decreasing similarity. In this search policy the number of visited nodes depends on both the query vector, the cluster tree structure, and the centroid vectors. Notice that this top-down search strategy needs some modifications for very large document databases. Since for a high level cluster containing thousands of documents it would be impossible to obtain a reasonable centroid. Later in the paper we provide the efficiency performance of this search strategy [19] and compare it with ours.

In a non hierarchical clustering environment the best-match CBR policy selects the best-matching n_S number of clusters and retrieves the best-matching d_S documents of the selected clusters. In this paper we use the best-match CBR policy.

The implementation of the best-match CBR (or simply CBR) is conventionally done in the following two ways.

ICDV: Match the query vectors with the centroid vectors (CV) and the document vectors (DV) of the members of the best-matching clusters.

ICDV: Match the query vectors with the inverted indexes of centroids (IC) and the document vectors (DV) of the members of the best-matching clusters.

In addition to these conventional methods we propose the following method of implementation for CBR.

ICIIS: Match the query vectors with the inverted indexes of centroids (IC) and the inverted indexes of all documents (IIS).

In the transition from the conventional methods to the new method there is a hybrid approach.

CVIIS: Match the query vectors with the centroid vectors (CV) and the inverted indexes of all documents (IIS).

Notice that ICIIS has the potential of being efficient since query vectors are usually very short and the number of clusters to be selected can be very large depending on the database size. For example, the average query vector length of the queries associated with some of the experimental document databases are as follows (database acronym, average query length): (CACM, 10.80), (CISI, 28.29), (CRAN, 9.17), (INSPEC, 15.82), (Keen, 10.30), (LISA, 16.50), (MED, 10.10), (NPL, 7.16), (TODS214, 13.36), (UKCIS, 7.4) [3, 8, 13]. However, as we stated before in automatically generated query environments, the query length can be long (e.g., 200) [14, 16]. In the paper we analyze the efficiency of all of these CBR strategies and IIS.

Now let us consider the possibility of implementation of these CBR strategies in terms of their internal memory requirements. The strategies CVDV and ICDV should not impose any problem. Our concern is ICIIS. For a document database containing n_C number of clusters the IC component of ICIIS requires $O(n_C)$ memory space to accumulate the partial similarities during the traversal of the query term lists. This is equal to $(4*n_C)$ bytes. Another array holds the best n_S partial similarities seen so far. During similarity calculations all partial similarities are initially set to zero, then they are updated as the term lists are traversed. If the current similarity of a cluster is large enough, it is inserted in the top similarity array of the centroids. After processing the all query terms, the top similarity array contains the selected clusters. For the ISS component we use the identical technique [1, 18, 19]. See [19, Figure 1] for an example. Therefore, ISS requires an array of size (4*m) bytes for a database of size m documents. For the update of the top similarity array we consider only the documents of the selected clusters. The implementation of CVIIS is similar. In all approaches the term weights of each vector (query, centroid, document) are normalized according to the matching function to be used [13].

The above discussion shows that ICIIS is realizable with an internal storage of $(4*m + 4*n_c)$ bytes for similarity arrays and plus $(4*m)$ bytes for cluster membership information of the documents. This excludes other program variables and the program itself. The storage overhead of the partial similarity vectors of clusters, $(4*n_c)$ bytes, and top similarity arrays of the best-matching clusters ,$(4*n_s)$ bytes, and best-matching documents, $(4*d_s)$ bytes, is negligible. All in all, the ICIIS strategy is feasible because of the large amounts of inexpensive memory available on today's computers. More conservatively, we can say that ICIIS would be feasible for all but the largest document databases (greater than one million documents) [1].

3. EXPERIMENTAL ENVIRONMENT

In this section a brief description of the experimental database, the clustering algorithm used for the creation of the CBR environment, the matching functions, and the IR quality of the CBR is provided.

3.1 Document Database and Clustering Algorithm

In the experiments the INSPEC database is used. The document-term (D) matrix and the queries of the database are commonly used. Some statistical characteristics of the INSPEC document database and the queries are provided in Table 1. In this table m, n, and t respectively, indicate the number of documents, terms, and total term assignments (the number of nonzero entries in the D matrix).

The clustering structure of the experiments is created using the cover-coefficient-based clustering methodology (CM^3) [3]. In CM^3 some of the documents are selected as cluster seeds. Then nonseed documents are assigned to one of the clusters initiated by the seed documents. The number of clusters, n_c, is determined using the cover-coefficient (CC) concept. According to CC, for an m (document) by n (term) D matrix the value range of n_c and the average cluster size (d_c) is as follows.

$$1 \le n_c \le \min(m, n), \quad (1, m/n) \le d_c \le m.$$

The computational complexity of CM^3 is $O(m * x_d * t_{gs})$, where x_d and t_{gs} indicate the average number of terms per document and the average number of seed documents per term, respectively.

Table I. Characteristics of the INSPEC Database and its Queries

A. Database

m	n	t	n_c
12,684	14.573	412,255	475

Table I. (cont) B. Queries

No. of queries	Avg. query length	Avg. relevant doc. per query	No. of distinct doc. retrieved	No. of distinct terms for q. def.s
77	15.82	33.03	1,940	577

3.2 Query Matching and IR

For query matching there are several matching functions depending on term weighting component of document and query terms. Term weighting basically has three components: the term frequency component, the collection frequency component, and the normalization component [13, 14]. The weights of the terms of a document and a query are defined as the inner product of document and query vectors. Salton and Buckley [13] obtained 1,800 different combinations of document/query term-weight assignments, of which 287 were found to be distinct. In the same study, six of these combinations are recommended due to their superior IR performance. In the experiments we used these six matching functions and the famous cosine similarity function. In this study the cosine similarity function is referred to as TW1 (term weighting 1) and the other six are referred to as TW2 through TW7. In [13] The effectiveness of these matching functions for the INSPEC database is reported in [3]. In that study they are again referred to as TW1 through TW7; in [13] they are, respectively, defined as (txc.txx), (tfc.nfx), (tfc.tfx), (tfc.bfx), (nfc.nfx), (nfc.tfx), and (nfc.bfx). For the query processing time efficiency experiments of CVDV and ICDV a trace of each query is obtained in terms of the selected clusters.

The study reported in [3] shows that CM^3 is 15.1 to 63.5 (with an average of 47.5) percent better than four other clustering algorithm [7] in CBR. The same study also shows that the IR effectiveness of the algorithm is compatible with a very demanding (in terms of CPU time and main memory) complete link clustering method that is know to have good retrieval performance [18, 19]. The effectiveness is measured in terms of precision (or equivalently the total number of relevant documents retrieved for all queries). Precision is defined as the ratio of the number of retrieved relevant documents to the number of retrieved documents. The effectiveness experiments show that selecting more clusters increases effectiveness. The increase in effectiveness would be expected to increase up to a certain n_S, after this (saturation) point, the retrieval effectiveness remains the same or improves very slowly [3, Figure 6]. For the INSPEC database this saturation point is observed at n_S= 50. In the experiments, precision is obtained after retrieving d_S (10, 20) number of documents from n_S (50) number of clusters [3, Table IX]. TW2 is observed as the most effective matching function. In a given experiment the same matching function is used both for cluster and document selection.

4. STORAGE STRUCTURES

The data structures used for the implementation of inverted index search (IIS), and all CBR strategies are defined in this section. These structures, when appropriate, are synonymous to those defined in [18, 19]. This section first describes the storage structures and how to determine their sizes, then the values for our experimental database INSPEC. The assumed storage environment is a disk device. The storage requirements are expressed in terms of bytes.

Since the number of pages (data blocks) is not a precise description of the relative sizes of data structures [18, p. 109].

4.1 Data Structures

For IIS we need to have the inverted index representation of the D matrix, IID, and a direct file of document vectors, DV, of the D matrix. It is assumed that the direct file DV would be needed for query expansion during query feedback [18].

An inverted index entry contains a fixed length header and a variable number of tuples that have the form <document id, term weight>. Each header is 12 bytes in length and contains a term number (4 bytes), the number of documents in which the term appears (4 bytes), and a pointer to the beginning of the document tuples for that term (4 bytes). Each tuple requires 8 bytes and consists of document number (4 bytes) and the weight of the term within that document (4 bytes). The storage requirement of IID is (12*n + 8*t).

Document vectors, DV, consists of a document number, a tuple for each term used in the corresponding document and a flag to signify the end of a vector. Each tuple contains number and weight of the corresponding term. Therefore the structure of a vector and the storage requirement of each component is as follows.

vector:	<document no.>	<term no., weight>*	<flag>
bytes:	4	8	4

In the above representation "*" indicates one or more occurrences of the <term no., weight> tuple. Accordingly, the storage requirement of DV is (8*m + 8*t).

In all CBR implementation techniques a direct file of centroid vectors, CVD, is needed. This is because in a dynamic document environment clusters must be maintained and the effect of the maintenance need to be reflected to the centroids [2, 4, 10]. In CBR strategies ICIIS and CVIIS the data structure DV is needed. However, in ICDV and CVDV the data structure DV is not required, since for these cases document vectors are available in clustered form. The storage requirements of CVD is ($8*n_c + 8*t_c$), where t_c indicates the total number of term assignments in the centroids.

The cluster membership information, CM, is also needed in all CBR techniques. CM is directly used in ICIIS and CVIIS. In all cases it is needed for cluster maintenance and centroid generation. For each cluster CM contains the cluster number, number of member documents, and the document numbers of the members. Accordingly its storage requirement is equal to ($8*n_c + 4*m$).

The CBR technique ICIIS strategy needs the inverted index of centroids, IIC, for cluster selection. IIC has the same structure as IID and its storage requirement is equal to ($12*n + 8*t_c$). ICIIS uses IIS for document selection. The IIS component requires IID.

The CBR technique CVIIS requires the centroid vectors for centroid selection. If we use direct file of centroid vectors, CVD, for query centroid matching the number of disk accesses is at least equal to n_c. (This assumes that each centroid vector begins on a new page for possible modification.) To reduce I/O time we use packed centroid vectors, CV. The data structure CV is the sequential version of the direct file CVD and its storage requirement (in terms of bytes) is like that of CVD ($8*n_c + 8*t_c$).

The CBR technique ICDV requires the inverted index of centroids, IIC, and the clustered document vectors, CDV. CDV contains the same information as DV does, i.e., the document vectors. However, in CDV case, document vectors are clustered and the document associated with the same cluster are stored in the same page(s). In CDV each cluster requires a header of 12 bytes. A cluster header contains the corresponding cluster number, number of member documents, and a pointer pointing to the vectors of the member documents. The storage requirement of CDV is then equal to ($12*n_c + 8*m + 8*t$).

The last CBR technique, CVDV, requires previously defined data structures CV and CDV.

The application of C^3M to the example D matrix given in Figure 1 produces the document clusters $c_1 = \{d_1, d_2\}$ and $c_2 = \{d_3, d_4, d_5\}$ [3]. The contents of the data structures for IID , IIC, CV, and CDV for the example D matrix are depicted in Figure 2.

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

Figure 1. The example D matrix.

In Figure 2 IID shows that term-1 (t_1) appears in two documents: document-1 (d_1) and d_2. Since the D matrix is binary the term weights are one. IIC provides the same information for centroids. For example, the header information for t_2 indicates that it appears in two centroids (the centroids of c_1 and c_2). In c_1 both members, d_1 and d_2, contain t_2, that is why the weight of t_1 in c_1 is two. Similarly in c_2 two members, d_4 and d_5, contain t_2. The third data structure of Figure 2, depicts the contents of CV. As defined before CV contains the centroid vectors. The first centroid (indicated by <1>) contains t_1, t_2, and t_5 with a weight of 2 and t_4 with a weight of 1. The end of each centroid is indicated by a flag (<0>). The cluster header part of the last data structure of Figure 2, CDV, indicates that c_1 and c_2, respectively, contain two and three member

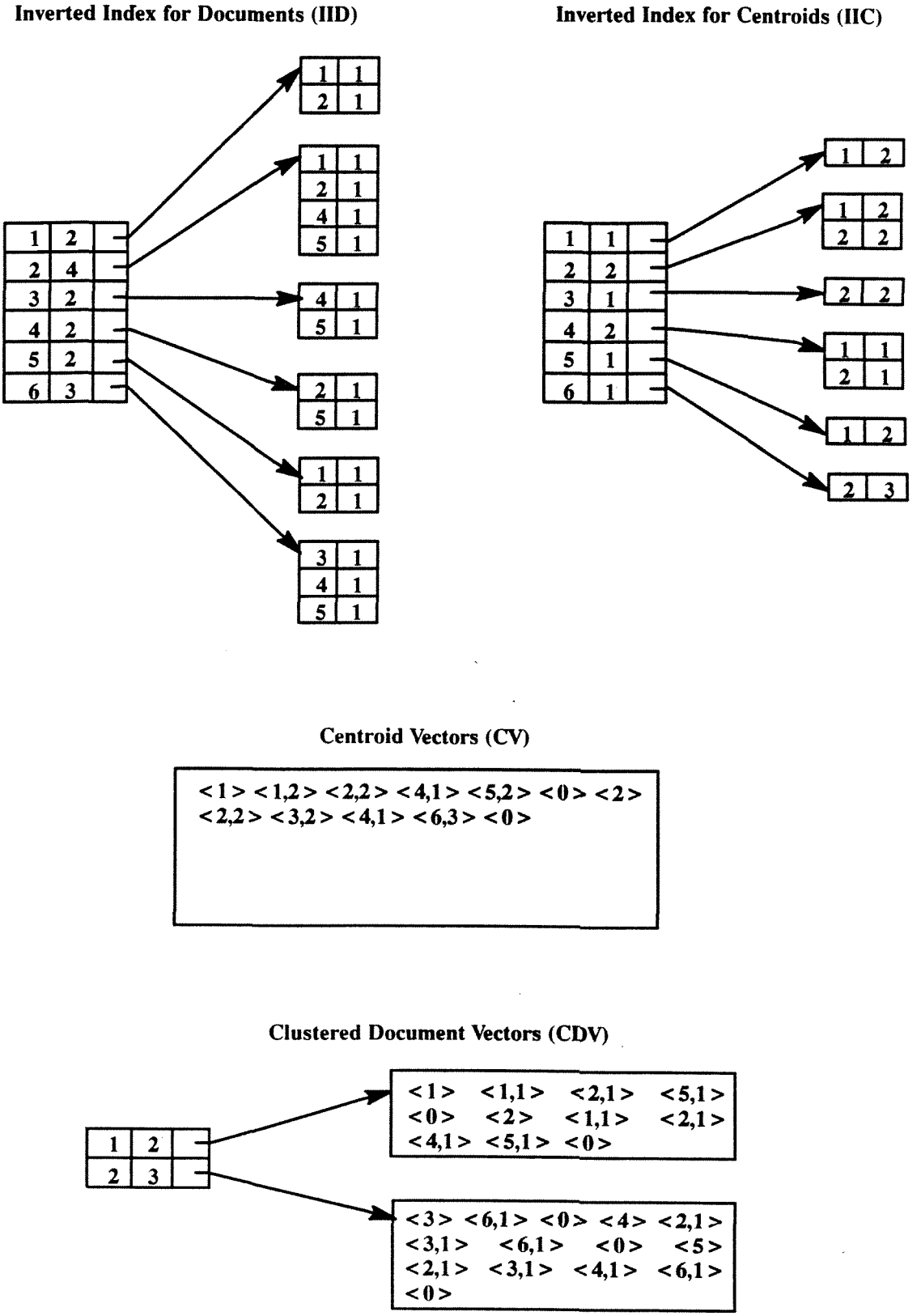

Figure 2. Data structures for the example D matrix.

documents. The first member of c_1 is d_1 (indicated by <1>) and d_1 is defined by terms t_1, t_2, and t_5, and the weight of each term is 1. The end of each document vector is indicated by a flag (<0>).

4.2 Storage Requirements

For the CBR experiments we used two (maximum) centroid lengths 250 and 500. The terms with the highest total number of occurrences within the documents of a cluster are chosen as centroid terms. The weight of a centroid term is defined as its total number of occurrences in the documents of the corresponding cluster. (In the IR experiments these weights are normalized according to the matching function.) The characteristics of the centroids are given in Table II. In this table the column "%D" indicates the total size of centroid vectors as a percentage of t of the D matrix of INSPEC.

Table II. Characteristics of the Centroids

Maximum Length	Average Length	Avg. No. of Cent./Term	% of Used Clus. Terms	Total No. of Distnict Terms	%D
250	237.09	12.96	51	8,689	27
500	399.06	14.74	86	12,857	46

DV = Document vectors (direct access file): $8*m + 8*t = 3,399,512$
IID = Inverted index for documents: $12*n + 8*t = 3,472,916$
CDV = Clustered document vectors: $12*n_c + 8*m + 8*t = 3,405,212$
IIC_{250} = Inverted index for centroids (cent. length = 250): $12*n + 8*t_{c,250} = 1,075,820$
CV_{250} = Centroid vectors (cent. length = 250): $8*n_c + 8*t_{c,250} = 904,744$
CVD_{250} = Centroid vectors (direct access file version): $8*n_c + 8*t_{c,250} = 904,744$
IIC_{500} = Inverted index for centroids (cent. length = 500): $12*n + 8*t_{c,500} = 1,691,308$
CV_{500} = Centroid vectors (cent. length = 500): $8*n_c + 8*t_{c,500} = 1,520,232$
CVD_{500} = Centroid vectors (direct access file version): $8*n_c + 8*t_{c,500} = 1,520,232$
CM = Cluster membership information: $8*n_c + 4*m = 54536$

(m= 12,684, n= 14,573, t= 412,255, $t_{c,250}$= 112,618, $t_{c,500}$= 189,554, n_c= 475)
Figure 3. Storage requirements of the data structures (in terms of bytes).

Table III. Storage Cost of Different Approaches
(% increase of CBR over IIS is included)

Centro. Length	Method	Cost Components	Total Cost	% change w.r.t. IIS
-	IIS	DV+IID	6,872,428	0
250	ICIIS	DV+IID+IIC_{250}+CVD_{250} +CM	8,907,528	30
	CVIIS	DV+IID+CV_{250}+CVD_{250} +CM	8,736,452	27
	ICDV	IIC_{250}+CDV+CVD_{250} +CM	5,440,312	-21
	CVDV	CV_{250}+CDV+CVD_{250} +CM	5,269,236	-23
500	ICIIS	DV+IID+IIC_{500}+CVD_{500} +CM	10,138,504	48
	CVIIS	DV+IID+CV_{500}+CVD_{500} +CM	9,967,428	45
	ICDV	IIC_{500}+CDV+CVD_{500} +CM	6,671,288	-3
	CVDV	CV_{500}+CDV+CVD_{500} +CM	6,500,212	-5

The storage requirement of the search techniques is determined by the variables m, n, t, t_C, and n_C. The t_C values of the centroid lengths 250 and 500 are, respectively, indicated by $t_{C,250}$ and $t_{C,500}$. The storage requirements of the data structures needed for the implementation of all search techniques are given in Figure 3. According to these storage components, the external storage space requirement of all retrieval strategies is given in Table III. The table also includes the description of cost components and the percentage increase of storage requirements of cluster searches with respect to IIS.

The results indicate that for two CBR techniques, ICIIS and CVIIS, there is a moderate overhead of less than fifty percent more storage with respect to IIS. For centroid length of 250, ICIIS requires only 30 percent more storage. This is much less than the storage overhead of the centroids of the complete link and the average link clustering structures [18, p. 112; 19, Table 2]. Furthermore, ICDV and CVDV provide some decrease (between three to twenty-three percent depending on the centroid length) in storage overhead with respect to IIS. The significant amount of extra disk space required in hierarchical CBR is due to the fact that the agglomerative clustering algorithms generate many clusters (of widely varying sizes) [18, p. 144]. On the other hand, C^3M generates reasonably small number of clusters with respect to m [3]. Another agglomerative hierarchical clustering algorithm, the single link, provides seven percent decrease in storage requirement with respect to IIS. However, it is ineffective in IR [18, 20].

5. TIME EFFICIENCY

The total processing time includes the I/O time and the CPU time. In this section first we define how to measure I/O time and the CPU time and then present the experimental results.

5.1 Measurement of I/O Time

In the measurement of I/O time we have five storage components to consider. These are IID (inverted index of documents), IIC (inverted index of centroids), CV (centroid vectors), CDV (clustered document vectors), and CM (cluster membership information). The data structure CM is only necessary for ICIIS and CVIIS. We assume that CM is kept in main memory. Otherwise each query has the potential of imposing n_S number of disk accesses for cluster membership information.

The inverted index headers of IID and IIC are stored sequentially on as few pages as possible without splitting headers across pages. The cluster headers of the data structure CDV are assumed to be stored in main memory. In general the number of clusters is much smaller than the number of documents. According to the concepts of C^3M, $\max(n_C) = \min(m, n)$ and in general $n_C << m$. This is because in an operational environment n would be much smaller than m, and the

expected value of n_c is (n/x_d) [3]. Therefore, the internal storage requirement of the cluster headers is negligible.

The I/O time of each query is determined independently of the others. The time needed to retrieve the text is ignored. However, it should be noticed that in an operational environment the text I/O in a clustered environment would require considerably less time than that of an unclustered environment [5].

In the experiments no paging strategy is assumed. Since pages are consumed in the order they brought to main memory. For instance, a page containing term lists is used only once for the current query. The term headers may contain information for more than one query term. Then we may assume that the needed term information is saved in the working storage area immediately after reading a term header page into main memory. Therefore, the I/O time is determined by the distinct number of pages accessed for the query. Then in IIS, the I/O cost of each query term is one or zero page access for the header information. The number of page accesses for the term list is determined by the generality of the term. The term generality of t_j, t_{gj}, is defined as the number of documents containing t_j. Accordingly number of page accesses for the term list of t_j is determined as follows.

$$\lceil (8 * t_{gj}) / \text{page size} \rceil$$

In the experiments we assume that the I/O time of each page is the same. Actually this is impossible. Since the I/O time is mainly determined by the seek time and the rotational latency time. Therefore, a page which is very close to the previously accessed page requires significantly less I/O time. It is assumed that the I/O time of a page is 30 milli second independent of its size. Since the data transfer time is a very small portion of the page I/O time (typically less than five percent) the effect of this assumption on the results is negligible.

5.2 Measurement of CPU Time

The CPU time measurement is done in terms of the number of idealized instructions. For this purpose each array indexing, pointer update, comparison, and floating point addition and multiplication is considered as one idealized instruction. It is assumed that each idealized instruction requires one micro second. The time needed for the initialization of the variables is ignored.

In query processing there are two ways of similarity calculation: using the inverted indexes IIS and IC; and using the clustered document vectors, CDV, and centroid vectors, CV. The algorithms of similarity calculations are provided in Figure 4. The same figure also shows the number of idealized instruction(s) to be executed for each statement. These algorithms and the assumptions are equivalent to the ones defined in [18, 19].

```
for (each query term) do
    begin
        if (at end of list) then                                        (1 inst.)
            go to next list
        else
            begin
                sim[current vec id]:=sim[current vec id]+q. term wgt*vec term wgt;   (3 inst.)
                increase list pointer to point to the next vec id in list           (1 inst.)

            end
    end
```

Figure 4.a. Inverted index (IIS, IC) similarity calculation.

```
sim:= 0;
    while (true) do
        begin
            if (q. term no > vector term no) then                       (1 inst.)
                begin
                    increase vector pointer;                            (1 inst.)
                    if (at end of vector) then exit loop                (1 inst.)
                end
            else if (q. term no < vector term no) then                  (1 inst.)
                    begin
                        increase query pointer;                         (1 inst.)
                        if (at end of query) then exit loop             (1 inst.)
                    end
                else {there is a match}
                    begin
                        sim:= sim + (q. term wgt * vector term wgt);     (2 inst.)
                        increase query pointer;                         (1 inst.)
                        if (at end of query) then exit loop;            (1 inst.)
                        increase vector pointer;                        (1 inst.)
                        if (at end of vector) then exit loop            (1 inst.)
                    end
        end
```

4.b. Vector (document, centroid) similarity calculation.

Figure 4. Implementation of similarity calculations using inverted index and vectors.

5.3 Experimental Results

In this section the effect of various search parameters (number of selected clusters, page size, centroid length, and matching function) on efficiency is presented. In these analyses the number of page accesses is used, since as it is shown later it is the major component of the query processing time. At the end the overall evaluation and a comparison with the results of [19] are provided. In this section, unless otherwise specified, the results are given for the TW2 matching function, the centroid length of 250, and for the first fifty best matching clusters. This is because TW2 is the most effective function of our experiments and the centroid length of 250 is also used in the experiments of [19].

5.3.1 Effect of Number of Selected Clusters (n_S)

As we defined before in our experiments the best-match CBR policy is used. That is, we first select the best-matching n_S number of clusters, then choose the best-matching d_S documents from the selected clusters. Figure 5 shows the effect of n_S on the average number of page accesses for the page size of 4K. The CBR search strategies CVIIS and ICIIS are independent of n_S. However, the number of page accesses for ICDV and CVDV increases with the increase of n_S. Because, each selected cluster brings new document vectors to match with the query vector. The ICIIS technique is the most efficient CBR for $n_S > 20$. (Notice that the average query vector of the experiments is 15.82!) For lower values of n_S, ICDV is the most efficient CBR strategy. However, we know that for the INSPEC database to obtain effective results we have to open about fifty clusters [3, Figure 6]. The observations indicate that for long query vectors ICDV has the potential of being most effective strategy. Since for the selection of documents, ICIIS must access one or more pages of IID for each query term. However, ICDV only accesses the document vectors of the selected clusters. For a very large database expected value of n_S is much larger than expected length of a query vector. Therefore, for most cases ICIIS will remain the most efficient strategy for CBR. The effect of n_S in connection with the other experiment parameters is defined in the following sections.

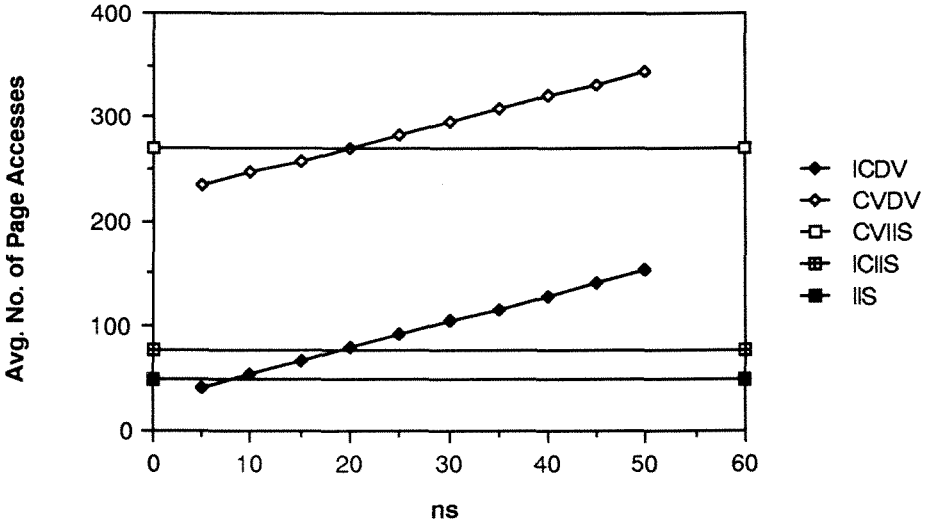

Figure 5. The effect of n_S on average no. of page accesses (page size= 4K)

5.3.2 Effect of Page Size

The effect of page size on average number of page accesses is shown in Table IV. The experiments indicate that the increase in page size decreases the I/O time. This is an expected behavior. However, it should be noticed that we cannot increase the page size indefinitely since

larger page sizes would imply more internal page fragmentation. Table IV indicates that for all page sizes IIS is the most efficient method. It is followed by ICIIS. The least efficient method is CVDV.

The comparison of CVIIS and ICDV shows that CVIIS is more efficient for smaller page sizes; however, for larger page sizes the reverse is true. In CVIIS and ICDV the heavy I/O components are, respectively, centroid vectors (CV) and clustered document vectors (CDV). CV and CDV are more influential in determining the overall efficiency of the methods. The decrease in I/O for ICDV is much more than the decrease in I/O for CVIIS due to the increase in page size. This is because the size of the selected cluster documents is much smaller than that of centroid vectors. For the page size of 0.5 K, ICIIS is much more efficient than ICDV. Since in ICDV, for each selected cluster, several document vector pages are accessed due to small page sizes.

By definition the number of page accesses for ICIIS and CVIIS is independent of n_s. However, the change in n_s affects ICDV and CVDV. Clearly for smaller n_s values the average number of page accesses for ICDV and CVDV is less. In most of the cases ICIIS becomes the most efficient method even for n_s values smaller than the average query length. This is especially true for smaller page sizes. For n_s values greater than the average query length ICIIS is the most efficient method for all page sizes.

Table IV. Effect of Page Size on Average No. of Page Accesses

Page Size (K bytes)	IIS	ICIIS	CVIIS	ICDV	CVDV
.5	236.38	302.45	2004.38	3619.62	5321.54
1	129.29	174.52	1013.29	1048.22	1886.99
2	75.86	110.43	517.85	350.99	758.42
4	48.82	77.58	269.82	152.23	344.47

Table IV (cont.). (Centroid length= 500)

Page Size (K bytes)	ICIIS	CVIIS	ICDV	CVDV
.5	312.04	3206.38	4158.49	7052.83
1	178.82	1614.29	1188.03	2623.49
2	112.04	818.86	389.32	1096.14
4	77.53	420.82	159.34	502.62

5.3.3 Effect of Centroid Length

The effect of centroid length on paging, specifically percentage increase in paging with the centroid length of 500 with respect to the centroid length of 250 is depicted in Table V. The effect of centroid length on ICIIS is minor (zero to three percent). This is because, in general the terms brought by a longer centroid are not necessarily the query terms which have not been selected by using a shorter centroid length or shorter centroids may contain all of the query terms.

The ICIIS with 4K pages results in 77.58 and 77.53 number of page accesses, respectively, for centroid length of 250 and 500. The lower number of page accesses of ICIIS with centroid

length 500 is due to arbitrary selection of terms which appear in only one cluster number. In the experiments this policy eliminated very few of the query terms from the longer centroids. That is, due to this arbitrariness, centroid length 500 is not necessarily a superset of centroid length 250 for all centroids.

The effect of the centroid length length is more noticeable on CVIIS and CVDV. Since both of these techniques use the data structure CV (compact centroid vectors) and an increase in centroid length directly affects the cost figures coming from CV. The increase in ICDV is moderately low (since newly added terms are not query terms) with respect to CVIIS and CVDV and it ranges between five to fifteen percent.

The centroid length experiments indicate that after a threshold centroid length ICIIS remains the same and becomes the most efficient CBR strategy. The same is untrue for the other CBR strategies since they incur more paging cost with the increased centroid length.

Table V. The Effect of Centroid Length on Paging
(% increase in paging with cent. len. of 500 w.r.t. cent. len. of 250)

Page size (K)	ICIIS	CVIIS	ICDV	CVDV
.5	3	60	15	33
1	3	59	13	39
2	2	58	11	45
4	0	56	5	46

5.3.4 Effect of Matching Function

The effect of the matching function (TW1 through TW7) on the experiments was negligible (for all cases always less than one percent). This is natural for IIS and ICIIS since their efficiency is independent of the matching function. The other cases can be explained as follows: different matching functions select not necessarily the same but similar set of clusters, or clusters with compatible sizes, or both. More importantly, C^3M provides a uniform distribution of documents among clusters. Therefore, choosing this set or that set does not make much difference.

We must state that the same cannot be said for an agglomerative hierarchical clustering environment. Since such an environment contains numerous clusters with considerable differences in size [18, p. 144]. Therefore, a top-down search policy using different matching functions may choose considerably different search paths with considerably different efficiency results. Furthermore, it would be difficult to replicate CBR experiments in a hierarchical clustering environment. Since the agglomerative hierarchical clustering algorithms do not necessarily define a unique hierarchy for a given data set. Since decision ties are broken arbitrarily. Furthermore small changes in the similarities between documents may result in considerable differences in the clustering structure [18, pp. 27-28].

5.3.5 Overall Evaluation

The efficiency figures of all search strategies in terms of mean number of pages, mean number of instructions, and mean number of seconds for page size of 4K are given in Table VI. The overall evaluation is done using this page size, since it is common. The same information for hierarchical (complete link) CBR for centroid lengths of 75, 100, and 250 are depicted in Table VII [19].

The experimental results indicate that CPU time constitutes a small portion of the total query processing time. For example, for IIS it is less than five percent. For ICIIS, it is less than four percent for both centroid sizes. The same is also true for the other CBR strategies.

Among all CBR strategies, ICIIS is the most efficient strategy in terms of both paging and CPU time. For example, for the centroid length of 250, CVIIS, ICDV, and CVDV, respectively, require 253 percent, 100 percent, and 353 percent more mean processing time with respect to ICIIS. For the centroid length 500 the percentage increases are (in the same order) 450, 110, and 559.

Table VI. Efficiency Performance for IIS and CBR (page size= 4K)

		Max centroid length = 250			
	IIS	ICIIS	CVIIS	ICDV	CVDV
Mean pages	48.82	77.58	269.82	152.23	344.47
Mean instr.	67,893	81,195	414,383	257,471	589,659
Mean secs.	1.53	2.41	8.51	4.82	10.92

Table VI. (continued)

	Max centroid length = 500			
	ICIIS	CVIIS	ICDV	CVDV
Mean pages	77.53	420.82	159.34	502.62
Mean instr.	84,727	626,732	270,336	812,341
Mean secs.	2.41	13.25	5.05	15.89

Table VII. Efficiency Performance for Hierarchical CBR (taken from [19, Table 6])

	Max centroid length		
	75	100	250
Mean pages	218.1	257.2	418.3
Mean instr.	221,141	276,867	517,999
Mean secs.	6.76	7.99	13.07

The efficiency performance of hierarchical CBR given in Table VII indicates that the centroid length is influential in efficiency. For the centroid length 250 the efficiency of CVDV is compatible with that of hierarchical CBR. This may be coincidental or may be due to the fact that the

technique used in [19] is somewhat equivalent to the CVDV approach: centroid and document vectors are matched during query processing.

The comparison of Table VI and Table VII shows that ICIIS and ICDV are more efficient than hierarchical CBR. This is true for all centroid lengths. For the common centroid lengths (250) of both studies the mean processing time of ICIIS and hierarchical cluster searches, respectively, are 2.41 and 13.07 seconds. In other words, hierarchical CBR requires 442 percent more processing time. The same comparison with respect to ICDV indicates 171 percent more processing time. Both in ICIIS and ICDV, inverted centroid (IC) data structure contributes to the improvement. Further improvements in ICIIS comes from IIS. Similarly, in a hierarchical clustering environment inverting the top level centroids of the tree and starting the cluster search by using an inverted search on top level centroids should save time [18, 19]. For further improvements one can incorporate IIS to hierarchical search. This prevents the calculation of similarity values for documents containing no common term with the query vector.

6. EFFECTIVENESS OF ICIIS

In the previous section we showed that ICIIS is an efficient CBR strategy. Furthermore, ICIIS provides the results of FS without a cost. In this section we want to show that the combined FS and CBR has potential benefits in terms of increasing the effectiveness of IR.

Efforts to choose FS or CBR in specific environments have proven fruitless. The study reported in [9] considers statistical characteristics of queries to choose between CBR and FS. The findings of the study indicate that the automatic selection criteria will be unable to choose between different search mechanisms using the statistical characteristics of queries. The study reported in [6] investigated the use of learning automata for the selection of search strategies. However, since the alternative strategies were too similar to each other, the learning automata approach did not provide any improvement.

Our experiments show that FS and CBR return considerably different sets of documents. This becomes more noticeable if FS and CBR use different matching functions. For example, for the centroid length of 250 and after examining twenty best-matching documents of the first fifty best-matching clusters, the average of the percentage of common entries between FS and CBR for all matching functions is sixtynine. (The average is found by averaging the individual results of the queries.) The same average using different matching functions for FS and CBR is fortyfive. The same figures for the first ten best-matching documents, respectively, are seventyfour and fourtyfour. This means that the results of FS and CBR can be combined to improve effectiveness.

The "optimum" search using FS and CBR provides considerable effectiveness (precision) improvements. In an optimum search, FS and CBR both find d_S documents. Then the relevant documents from each search are combined after the elimination of duplicates then other

documents retrieved by FS or CBR are added until the desired number of documents (d_S) have been retrieved.

The percentage increase in precision with respect to FS using the combined optimum search for the CBR conditions (n_S = 50, d_S = 20, centroid length= 250) is provided in Table VIII. The average increase in this table is twenty-seven percent. The same average for (n_S = 50, d_S = 10, centroid length= 250) is twenty-six percent. The table indicates that the optimum search provides considerable improvement. In Table VIII the improvement in off-diagonal entries is more noticeable. They correspond to the combination of different matching functions for FS and CBR. Of course the implementation of combining different matching functions is more difficult in terms of space efficiency, since different matching functions require different normalizations for the document vectors. We are planning to implement methods to combine the available results of FS and CBR rather than choosing the results of one of them [6, 9]. A simple method that may work in our environment is defined in [8].

Table VIII. Percentage Increase in Precision Using Combined Optimum Search (for n_S = 50, d_S = 20, centroid length= 250)

		CBR						
		TW1	TW2	TW3	TW4	TW5	TW6	TW7
FS	TW1	22	27	24	32	40	38	48
	TW2	20	15	19	16	23	25	21
	TW3	21	27	17	31	34	31	33
	TW4	26	20	28	16	27	31	23
	TW5	32	32	38	33	18	23	16
	TW6	31	32	33	36	25	19	25
	TW7	42	28	36	28	21	26	22

7. CONCLUSION

In this study the efficiency of various CBR strategies is analyzed. A new method, ICIIS, for combining IIS and CBR is proposed and shown to be cost effective in terms of paging and CPU time. The observations prove that ICIIS is much more efficient than conventional CBR strategies including hierarchical cluster search. The storage overhead of ICIIS is moderate and less than that of the hierarchical clustering environment. Since the clustering structure of the experiments are created by the cover-coefficient-based clustering methodology (C^3M), the results also indicate that C^3M creates not only effective [3] but also efficient IR environment.

For very long query vectors a conventional CBR strategy, ICDV, may be more efficient than ICIIS. However, ICIIS is open to further efficiency improvements. (The same is partly true for ICDV.) Since inverted index searches can be optimized with a slight effectiveness deterioration [1]. This may make ICIIS more efficient than ICDV both for short and long query vectors. Further investigation is needed.

In the experiments the effects of several parameters (number of selected clusters, centroid length, page size, and matching function) are examined. By definition the efficiency of ICIIS is independent of the number of selected clusters and the similarity function used in a given experiment. Therefore, with ICIIS retrieving more clusters/documents has the same cost as retrieving fewer clusters/documents. The experiments also suggest that after a threshold, the effect of centroid length on the efficiency of ICIIS would be negligible. These characteristics are desirable.

Another advantage of ICIIS is the fact that it provides the results of FS without a cost. Another way of saying the same thing is it provides the results of CBR with a little cost over FS. (For the INSPEC database the difference is less than one second.) In the paper it is experimentally shown that the combined FS and CBR has potential benefits in terms of increasing the effectiveness of IR. This will be investigated in our future research.

REFERENCES

1. Buckley, C., Lewit, A. F. "Optimization of Inverted Vector Searches." In *Proceedings of the 8th Annual International ACM-SIGIR Conference* (Montreal, Quebec, June 1985). ACM, New York, 1985, 97-110.

2. Can, F. "Incremental Clustering for Dynamic Information Processing." (Submitted for publication.)

3. Can, F., Ozkarahan, E. A. "Concepts and Effectiveness of the Cover-Coefficient-Based Clustering Methodology for Text Databases." *ACM Transactions on Database Systems.* 15, 4 (Dec. 1990), 483-517.

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

5. Can, F. "Validation of Clustering Structures in Information Retrieval." In *Proceedings of the Canadian Conference on Electrical and Computer Engineering* (Montreal, Quebec, September 1989). EIC, Montreal, 1989, 572-575.

6. Croft, W. B., Thompson, R. "The Use of Adaptive Mechanisms for Selection of Search Strategies in Document Retrieval Systems." In *Research and Development in Information Retrieval*, C. J. VanRijsbergen, Ed. Cambridge University Press, Cambridge, 1984, 95-110.

7. El-Hamdouchi, A., Willett, P. "Comparison of Hierarchical Agglomerative Clustering Methods for Document Retrieval." *The Computer Journal.* 32, 3 (June 1989), 220-227.

8. Griffiths, A., Luckhurst, C., Willett, P. "Using Interdocument Similarity Information in Document Retrieval Systems." *Journal of the American Society for Information Science.* 37, 1 (1986), 3-11.

9. McCall, F. M., Willett, P. "Criteria for the Selection of Search Strategies in Best-Match Document-Retrieval Systems." *Int. J. Man-Machine Studies.* 25 (1986), 317-326.

10. Salton, G. *Dynamic Information and Library Processing.* Prentice Hall, Englewood Cliffs NJ, 1975.

11. Salton, G., Wong, A. "Generation and Search of Clustered Files." *ACM Transactions on Database Systems.* 3, 4 (Dec. 1978), 321-346.

12. Salton, G., McGill, M. J. *Introduction to Modern Information Retrieval.* McGraw Hill, New York, 1983.

13. Salton, G., Buckley, C. "Term-Weighting Approaches in Automatic Text Retrieval." *Information Processing and Management.* 24, 5 (1988), 513-523.

14. Salton, G., Buckley, C. "Parallel Text Search Methods." *Com. of the ACM.* 31, 2 (February 1988), 202-215.

15. Salton, G. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison Wesley, Reading, Massachusetts, 1989.

16. Stanfill, C., Kahle, B. "Parallel Free-Text Search on the Connection Machine System." *Com. of the ACM.* 29, 12 (December 1986), 1229-1239.

17. Van Rijsbergen, C. J. *Information Retrieval, 2nd ed.* Butterworths, London, 1979.

18. Voorhees, E. M. *The Effectiveness and Efficiency of Agglomerative Hierarchical Clustering in Document Retrieval.* PhD Thesis, Cornell University, Ithaca, NY, 1986.

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

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