On the Use of Clustered File Organization in Information Search and Retrieval*

Gerard Salton
Jose Araya

TR 89-989
April 1989

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*Research supported in part by NSF grant IRI 87-02735.

On the Use of Clustered File Organizations in Information Search and Retrieval

Gerard Salton and Jose Araya *

Abstract

In modern retrieval environments, collection searches are normally conducted on-line under user control. Iterative collection searches can then be performed where tentative queries are initially processed, to be successively improved and refined during the search process. When searches are carried out directly by the users, classified document organizations are especially useful, because collection browsing becomes possible, and access is provided to complete groups of related documents.

The state-of-the-art in automatic document classification is summarized in the present study, and evaluation data are provided to demonstrate the effectiveness and efficiency of clustered file search operations.

* Department of Computer Science, Cornell University, Ithaca, NY 14853.
This study was supported in part by the National Science Foundation under grant IRI 87-02735.

1. Introduction

Conventional automatic search systems normally depend on the availability of large term indexes specifying the addresses of applicable documents for all available search terms. Such *inverted term indexes* provide fast collection searches because the great bulk of the stored documents is ignored in any particular search. In fact, every document treated in an inverted file search is known in advance to carry at least one index term in common with the query, since each such document appears on one of the lists in the inverted index. For this reason, conventional inverted index processing can provide fast responses and acceptable search output.

The use of inverted indexes also entails disadvantages, first because it is never pleasant to have to build and maintain large auxiliary indexes, and second because the information pertaining to a particular document is scattered through many term lists in the index, making it difficult to access the complete data set for individual documents, or groups of related documents. The structure of inverted indexes thus precludes the collection browsing operations that are normally available in libraries with open shelf arrangements.

In on-line search environments, where users participate in the search process, collection browsing is especially desirable, because query formulations are simplified when the collections are organized by subject, and related items can therefore be accessed jointly. Furthermore, enhanced retrieval performance is often obtained by extending the retrieval activities from particular items known to be relevant to other neighboring or related items. This suggests that classified, or *clustered document collections* be considered for use in automatic information retrieval. In such an environment the documents are collected into groups, or clusters, of related documents, the assumption being that items included in a common class resemble each other in subject coverage, and that such items might then be usefully accessed as a group.

In many cases, efficient search strategies are implementable for clustered file organizations, because reasonable search results are often obtained by concentrating the search to particular document classes, the remainder of the collection being rejected. The clustered organization then simulates a classified library where the collection searches are confined to the items located on a few shelves in the same general vicinity.

In considering the information retrieval process in a clustered file environment, various problems must be addressed. The first is the automatic file classification process itself which necessarily entails substantial expense for large files. Additional questions deal with the efficiency and effectiveness of searches in clustered file

environments. These questions are treated in the remainder of this note.

2. Clustered File Organizations

In a library context, the term "browsing" often refers to operations in the catalog designed to provide access to terms or term classes related to certain initially available terms. [1,2] In automated search environments, *collection browsing* is more important than term browsing. In particular, it may be useful to retrieve items related to certain previously retrieved items; alternatively, one may want to extend a search from a particular class of items to other related document classes.

In considering the file classification process, several types of file organization can be distinguished. On the one hand, the collection can be subdivided into a small number of relatively large clusters. Because many different items are then included in each cluster, the pairwise associations, or similarities, between items in a given class may be relatively small. A typical example of such a loose cluster arrangement is shown in Fig. 1(a), where 16 documents, represented by x's in the Figure, are subdivided into two large classes. In Fig. 1, the distance between two x's is assumed to be inversely related to document similarity. That is, when two x's appear close together, the corresponding documents cover closely related subject areas. The reverse is true when two x's appear far apart.

An alternative cluster arrangement is obtained by dividing the collection into a large number of smaller clusters. This is shown in Fig. 1(b) for the documents previously used in Fig. 1(a). The document relationships remain the same in both cases, but the subdivision into five small classes represents a much tighter clustering arrangement where the pairwise similarities between items included in a common class are now much greater than they were for the earlier subdivision into the two large classes.

In practice one finds that it is much easier and less expensive to generate a loose cluster arrangement into a few large classes than a tight arrangement into many small classes. Unfortunately, when retrieval performance is considered, the advantage normally rests with the tight cluster arrangement, because the choice of large, heterogeneous document classes may lead to the retrieval of extraneous items that are of no interest to the users. Search precision as well as search recall may suffer as a result.

In information retrieval, *hierarchical* cluster arrangements present special advantages. In such a case, the collection is iteratively subdivided into increasingly smaller clusters. In particular, a few large clusters, representing

broad subject classes may be further divided into smaller clusters representing narrower topics, that themselves include still smaller classes and smaller topic areas. A typical hierarchical cluster arrangement is shown in Fig. 2. In the example, the complete collection is divided into three large clusters (labelled A, B, and C in Fig. 2), each of which is divided in turn into a number of smaller classes. When a hierarchical cluster arrangement is used, it is possible to conduct broad searches by accessing the large subject clusters; alternatively, specific searches can be made by concentrating on the smaller subclasses.

It is not possible in this note to consider in detail the many different *cluster generation* methods that are described in the literature. [3-6] For present purposes, it may be useful to distinguish the *heuristic* clustering methods that use only partial information about similarities between documents, from the more conventional complete clustering systems where the pairwise similarities between all document pairs must be used in the clustering operation. A typical heuristic method is the *single-pass* classification where documents are processed sequentially in random order, and each document is immediately assigned to the closest available cluster. [3]

The heuristic clustering methods are normally much less expensive to perform than the classical procedures because the large expense of first computing similarities for all pairs of documents can be avoided. On the other hand, the heuristic cluster organizations may produce uncertain retrieval results. For this reason, it is safer to use the previously mentioned hierarchical classification methods that are based on complete pairwise similarity information.

Two main types of hierarchical classification strategies are used, known aa the *divisive* and the *agglomerative* approaches, respectively. In the divisive approach, the document collection is assumed to fit into one global cluster which is then successively subdivided into smaller subunits in the best possible way. The agglomerative approach, on the other hand, starts with N individual clusters each containing a single document. These small clusters are then successively merged until eventually a single, final cluster is produced. The following typical framework is valid for all agglomerative methods:

1. Compute a similarity measure for all pairs of documents based on coincidences between the term sets assigned to the documents. For N documents, there are N(N-1)/2 such similarity pairs.

2. Place each of N documents in a class of its own.

3. Form a class (or cluster) by combining the most similar pair of existing classes.

4. Repeat step 3 if the number of classes left is greater than 1.

Different cluster structures are produced depending on the interpretation of the cluster similarity computation of step 3. If the similarity between a pair of clusters is taken to be the similarity between the *most similar* pair of items, one of which appears in each cluster, large classes of items are formed that are linked only imperfectly (since in each case only a single pair of items needs to be closely related before two particular clusters can be combined into one). The clusters produced by the merging of most similar pairs are known as *connected components*, and the clustering method is the *single-link* system.

On the other hand, when the similarity between the *least similar* pair of items from two clusters is used as a criterion of cluster similarity for cluster merging purposes, then all pairs of items in two particular clusters must be tightly linked before the two clusters can be merged. The resulting clusters are known as *cliques*, and the corresponding *complete-link* clustering system produces a large number of small, closely related groups of items.

Consider, as an example, the clustering process of Fig. 3 for the 6 items labeled **A** through **F**. The item similarity matrix is shown in Fig. 3(a). Assuming first that a single-link clustering system must be used, the operation begins by forming a single class **AF** from the two items **A** and **F** that exhibit a similarity coefficient of 0.9. Next, class **AF** is merged with item **E** to form class **AEF**, since the similarity between **A** and **E** is equal to 0.8. (Note that items **AF** and **E** can be merged at a 0.8 similarity level even though the similarity between **E** and **F** is only 0.3). Document **B** is next merged with class **AEF** at a 0.8 similarity level to form class **ABEF**, because the similarity between **B** and **F** also equals 0.8. Finally **D** is merged into the earlier class, at a similarity level of 0.6 because sim(A,D) = 0.6, and **C** is merged at the 0.5 level because sim(C,E) = 0.5. At this point the single class **ABCDEF** involves all documents and the merging operation ceases. The complete single-link cluster with a global similarity of 0.5 is shown in Fig. 3(b).

When a complete-link clustering strategy is used, to construct clusters at a 0.5 similarity level, items **A** and **F** are merged as before into class **AF** at a similarity level of 0.9. At this point item **E** cannot be merged with **AF** as before, because in the complete link process the similarity between **AF** and **E** is the minimum between sim(A,E) and sim(E,F), that is 0.3 instead of 0.8 as before. For the same reason, element **B** can also not be merged into class

AF. The next merging operation consists in combining items **E** and **B** into class **EB** at a similarity level of 0.7. No further merges are possible if the minimum clustering level of 0.5 is to be maintained. The complete link process thus produces four distinct small clusters at a similarity level of 0.5 as shown in Fig. 3(c) instead of the single large cluster of Fig. 3(b) generated by the single-link method.

It may be noted that in practice the complete similarity matrix between items cannot be considered globally in the manner suggested by the previous example when the collection size is large. In fact, similarity matrices for even moderately-sized collections of a few thousand items are unlikely to fit into the internal stores of most computers. In that case, the similarity matrix must be processed one row at a time, and several passes through the similarity array may be needed to generate a complete cluster arrangement.

3. Cluster Searching

The effectiveness of a cluster search is predicated on the assumption that documents appearing in common clusters are likely to be jointly relevant to particular user queries. In other words, the cluster hypothesis states that items with large pairwise similarities are likely to cover related subject areas. [7,8] In practice reasonable search results may be obtainable for clustered file searches even when the cluster hypothesis is only partially satisfied.

For search purposes, it is convenient to assign to each cluster a representative, central item to serve as a cluster identifier. This representative item, known as the *centroid*, may be considered to be the average, or the central, document included in each particular cluster. Instead of comparing a user query with the individual documents in a given cluster, a query-centroid comparison can then be used to decide whether a particular cluster should be retrieved or not.

Two main types of cluster searches must be distinguished, known respectively as top-down and bottom-up. In a *top-down search*, a query is first compared with the centroids corresponding to the largest, highest-level clusters. From there, the search proceeds downward in the cluster structure in order to reach successively smaller, lower-level clusters, until eventually the individual documents themselves are obtained. In a *bottom-up search*, on the other hand, the bottom-most centroids (corresponding to low-level clusters that directly contain individual documents) are immediately reached, using an auxiliary inverted index constructed for the low-level centroid terms. These bottom-most centroids then immediately provide access to certain stored documents. A bottom-up search

thus eliminates consideration of upper-level clusters, at the cost of building an auxiliary index of low-level centroid terms.

Consider, as an example, the cluster tree of Fig. 4(a), consisting of two high-level clusters represented by "hypercentroids", successively subdivided into superclasses represented by "supercentroids", low-level classes represented by centroids, and finally individual documents. In a top-down search, the query would first be compared with all existing hypercentroids. For hypercentroids exhibiting a sufficiently large similarity with the query, the corresponding supercentroids would be accessed next. Supercentroids with sufficiently large query similarity would next give access to particular centroids, and those would lead in turn to certain individual documents. A particular search might thus involve query comparisons with hypercentroids H_1 and H_2, supercentroids S_1 and S_2, centroids C_4 and C_5 and finally documents D_9 and D_{10} for the example of Fig. 4. The corresponding bottom-up search, on the other hand, would access all lower-level centroids C_1 to C_8 through the centroid index. The query similarity with centroid C_8 would then lead to the retrieval of documents D_9 and D_{10}.

The extensive cluster search experiments performed over the last few years lead to the conclusion that the large, loosely knit clusters derived from a single-link clustering process cannot be used effectively with top-down cluster searches, because the centroid-query matches are then incapable of correctly predicting the correct search path through the cluster structure. When only large document classes are available, the bottom-up searches are therefore preferred. [9,10] However, searches conducted with such large, single-link type clusters appear to produce results that are inferior to those obtainable with conventional inverted term indexes. For this reason, it is now believed that tight cluster structures must be used if reasonable search effectiveness is expected. [11-14]

Search experiments performed with four different document collections ranging in size from a thousand documents in medicine to nearly 13,000 documents in computer engineering, show that the average deterioration of the search precision of single-link cluster searches is about 25 percent compared with conventional inverted file searches. On the other hand, the top-down complete-link cluster searches may provide an average improvement of over 5 percent in search precision, compared with ordinary inverted file searches. [14] This suggests that the complete-link cluster organization should be considered seriously in operational situations.

In addition to the recall-precision performance of the cluster searches, a comparison between cluster searching and inverted file searching also involves consideration of search efficiency in terms of response time, or total

processing time per query. Table 1(a) gives elapsed time per query (including both processing and input-output times) for top-down complete-link searches using four experimental document collections. As the Table shows, the cluster searches always take substantially longer than the corresponding inverted file searches. The results of Table 1(a) are particularly discouraging for the large INSPEC collection, comprising 12,684 documents, where the elapsed time of 2 seconds per query for the inverted file compares with 26.3 seconds for the complete-link cluster search.

The large cluster search time is explained by the fact that the complete-link process generates a very large number of superclusters as the top level of the cluster hierarchy. For the INSPEC collection, 802 clusters appear on the top hierarchy level, all of which must be considered in a search in order to determine the proper search path to be followed for each query. These top-level clusters cannot be combined into larger clusters in a complete-link system, because in each case these clusters contain certain documents that exhibit a zero similarity with documents in the other top-level clusters. A cluster merge operation at a similarity level larger than 0 is then effectively prevented.

This suggests that the standard complete-link hierarchy could be modified by forcing additional cluster merges at the top level thereby creating an artificial superstructure as shown in Fig. 5 for the sample INSPEC collection. With the additional cluster superstructure, the number of top-level clusters is reduced from 802 to 45 for INSPEC. Corresponding reductions in the number of top-level clusters are obtained for the other smaller experimental collections. The results of Table 1(b) show that the top-down cluster searches are much more competitive with the inverted file searches for the new cluster structures than the searches performed with the original cluster hierarchies. [15] Indeed the elapsed time per query is reduced from 26.3 seconds per query for INSPEC to 5.9 seconds for the extended modified hierarchy.

An alternative way of speeding up the cluster searches, is to build an inverted index for the top-level centroids of the conventional cluster structure. This index provides fast access to the top level of the conventional hierarchy without matching the query explicitly with all top-level centroids. The elapsed time results for this inverted hierarchy appear in the right-most column of Table 1(b). The efficiency of the centroid inversion provides results that are comparable to those obtainable with the extended cluster hierarchy. [15]

The timing results of Table 1 were obtained using a relatively slow Microvax computer. This accounts for the relatively slow operating time of over 5 seconds per query for a cluster search of 13,000 items. To determine the

practical possibilities of cluster searching, it is useful to operate with larger collections and faster machines.

The performance results of Table 2 were obtained with a SUN-4 workstation, providing much greater computing power than the Microvax. Results are shown for the conventional INSPEC collection, as well as for an extended collection consisting of the original INSPEC items supplemented by 31,431 news articles covering the same subjects as INSPEC. The same 77 queries are used for both the INSPEC and the NEWS collections.

The average search precision results obtained at fixed recall points, shown on the left side of Table 2, indicate that the overall precision is lower for the NEWS collection than for INSPEC. This is explained by the fact that the added news articles are all considered to be nonrelevant to the INSPEC queries. The results of Table 2 show that the search effectiveness is somewhat lower for the extended hierarchy than for the other file organizations. The inverted hierarchy produces the same search effectiveness as the standard hierarchy and, as the Table shows, this effectiveness is somewhat superior to that obtained conventionally with ordinary inverted file searches. The elapsed time results appear on the right side of Table 2. Once again the extended hierarchy is not competitive with the inverted file searches for the large collections. However, the search times of 2 to 3 seconds per query for the inverted hierarchy fall into the range of practicality for collections of 50,000 documents.

To summarize, it appears that clustered file organizations can be usefully implemented for medium size collections of 50 to 100,000 items. If tight cluster organizations are used, such as complete-link structures, a search effectiveness somewhat superior to that obtainable with standard inverted file organization is obtained. The response times are slower for the clustered file searches than for the inverted file process, but the actual elapsed times obtained for the SUN-4 searches remain within practical limits for operational user-controlled searches. Whether the clustered file organizations are viable for larger collections above 100,000 documents remains to be shown in future experiments.

References

[1] M.J. Bates, An Exploratory Paradigm for On-Line Information Retrieval, *Proceedings Sixth International Research Forum on Information Science*, Frascati, Italy, September 1985.

[2] J.F. Cove and B.C. Walsh, Online Text Retrieval via Browsing, *Information Processing and Management*, 24:1, 1988, 31-37.

[3] G. Salton and A. Wong, Generation and Search of Clustered Files, *ACM Transactions on Database Systems*, 3:4, December 1978, 321-346.

[4] C.J. van Rijsbergen, Automatic Classification in Information Retrieval, *Drexel Library Quarterly*, 14:75-89, 1978.

[5] P. Willett, Recent Trends in Hierarchic Document Clustering: A Critical Review, *Information Processing and Management*, 24:5, 1988, 577-597.

[6] E.M. Voorhees, Implementing Agglomerative Hierarchic Clustering Algorithms for Use in Document Retrieval, *Information Processing and Management*, 22:6, 1986, 465-476.

[7] N. Jardine and C.J. van Rijsbergen, The Use of Hierarchic Clustering in Information Retrieval, *Information Storage and Retrieval*, 7:9, December 1971, 217-240.

[8] E.M. Voorhees, The Cluster Hypothesis Revisited, *Proceedings of Eighth International ACM/SIGIR Conference on Research and Development in Information Retrieval*, Association for Computing Machines, New York, June 1985, 188-196.

[9] C.J. van Rijsbergen and W.B. Croft, Document Clustering: An Evaluation of Some Experiments with the Cranfield 1400 Collection, *Information Processing and Management*, 11:5/7, 171-182, 1975.

[10] W.B. Croft, Clustering Large Files of Documents Using the Single Link Method, *Journal of the ASIS*, 28:6, November 1977, 341-344.

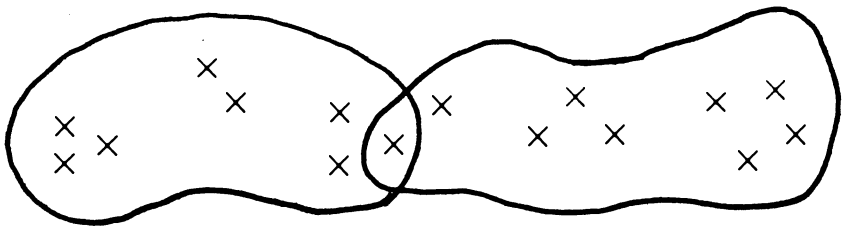
[11] A. Griffiths, L.A. Robinson and P. Willett, Hierarchic Agglomerative Clustering Methods for Automatic Document Classification, *Journal of Documentation*, 40:3, September 1984, 175-205.

[12] A. Griffiths, H.C. Luckhurst, and P. Willett, Using Interdocument Similarity Information in Document Retrieval Systems, *Journal of the ASIS*, 37:1, 3-11, 1986.

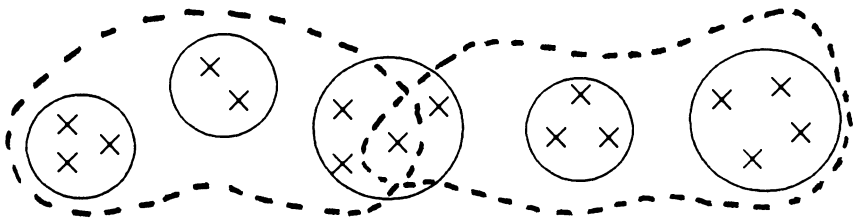
[13] E.M. Voorhees, The Efficiency of Inverted Index and Cluster Searches, *Proceedings of the 1986 ACM Conference on Research and Development in Information Retrieval*, F. Rabitti, editor, Pisa, Italy, September 1986, 164-174.

[14] E.M. Voorhees, The Effectiveness and Efficiency of Agglomerative Hierarchic Clustering in Document Retrieval, Ph.D. Dissertation, Cornell University, Ithaca, NY, January 1986.

[15] J. Araya, Comparisons of Inverted File Processing with Clustered File Manipulations, Technical Report, Department of Computer Science, Cornell University, Ithaca, NY, 1989.

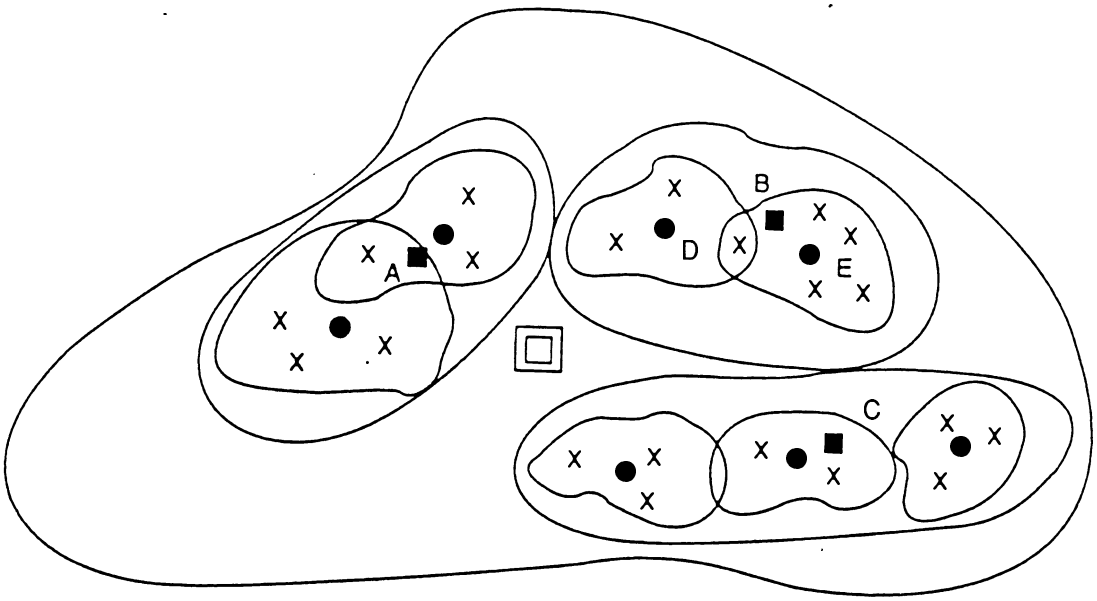


a) small number of large clusters



b) large number of small clusters

Sample Cluster Organization
Fig. 1



Hypercentroid

Supercentroids

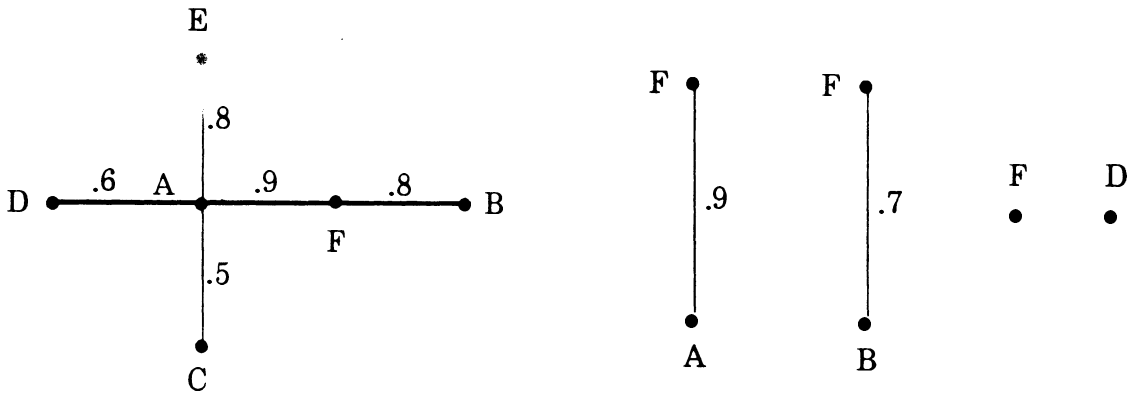
Centroids

Documents

Hierarchical Cluster Arrangement
Fig. 2

	A	B	C	D	E	F
A	●	.3	.5	.6	.8	.9
B	.3	●	.4	.5	.7	.8
C	.5	.4	●	.3	.5	.2
D	.6	.5	.3	●	.4	.1
E	.8	.7	.5	.4	●	.3
F	.9	.8	.2	.1	.3	●

a) Pairwise Similarity Matrix for Items A to F
(max similarity = 1, min similarity = 0)

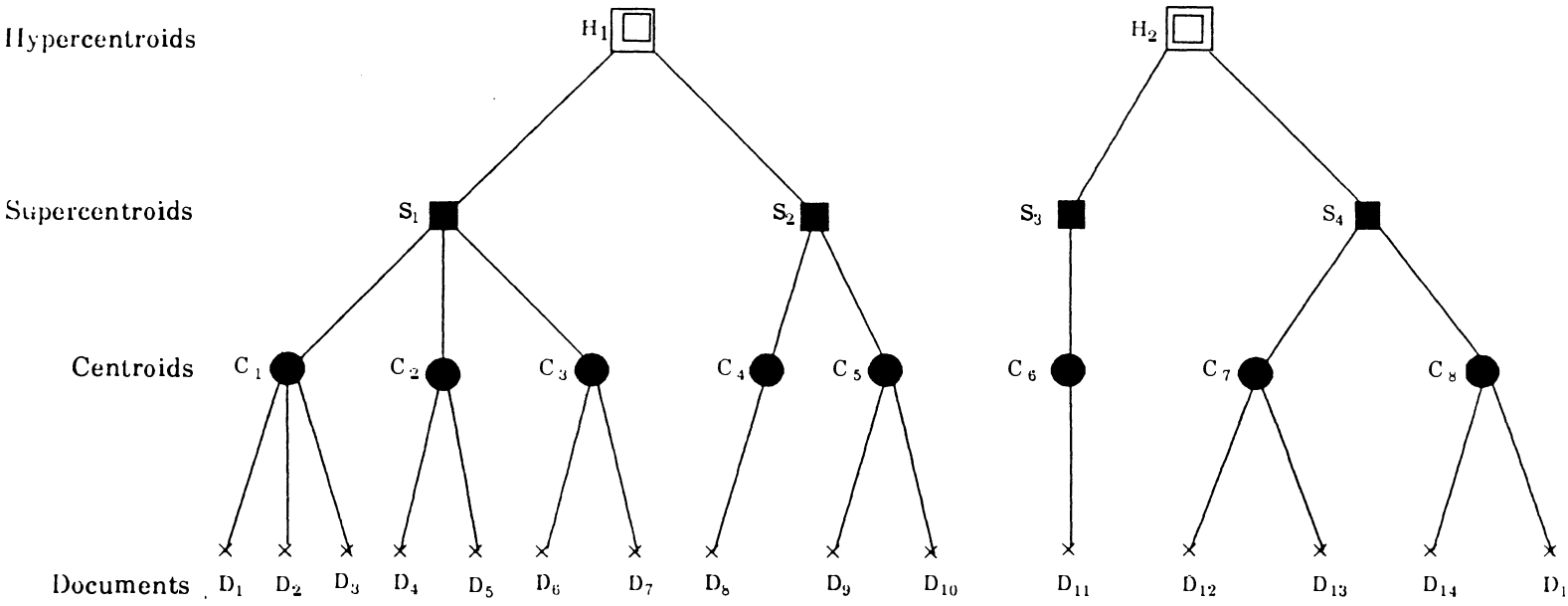


b) single link cluster

c) complete link cluster

Cluster Structures at 0.5 Similarity Level

Fig. 3



a) Typical Hierarchical Cluster Organization

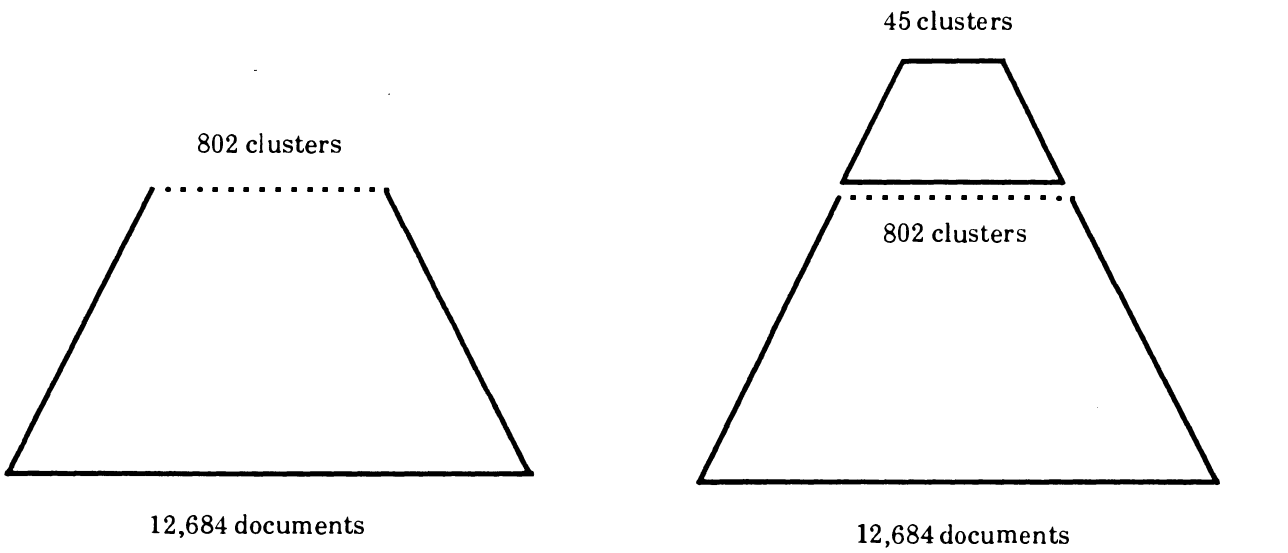
Top-down search: use all hypercentroids (H_1, H_2)
use best supercentroids (say S_2)
use best centroids (say C_5)
use documents in last cluster (D_9, D_{10})

Bottom-up search: use all low-level centroids(C_1 to C_7)
use documents in low-level clusters
(say D_6, D_7, D_9, D_{10})

b) Search Strategies

Typical Top-Down and Bottom-Up Searches

Figure 4



a) initial complete-link hierarchy

b) modified extended complete-link hierarchy

Cluster Hierarchy Modification for INSPEC Collection
Fig. 5

	Number of Top Level Centroids	Number of Documents	Elapsed Time per Query Inverted Index	Top-Down Complete-Link
CACM collection (50 queries)	327	3204	0.6 sec.	2.0 sec.
CISI (35 queries)	115	1460	0.3 sec.	5.1 sec.
MEDLARS (30 queries)	78	1033	0.5 sec.	6.7 sec.
INSPEC (77 queries)	802	12684	2.0 sec.	26.3 sec.

a) Processing Time for Standard Complete Link Hierarchy

	Extended Complete Link Hierarchy	Elapsed Time per Query		
	Number of Top Level Centroids	Inverted File Process	Extended Complete Link Hierarchy	Top Level Hierarchy Inversion
CACM	45	0.6	1.1	1.3
CISI	13	0.3	1.4	2.8
MEDLARS	15	0.5	2.9	3.3
INSPEC	45	2.0	5.9	5.2

b) Processing Time for Modified Complete Link Hierarchies

Processing Time for Complete Link Cluster Searches
(Microvax computer; CPU + I/O times = elapsed time)

Table 1

	Precision Performance				Elapsed Time per Query			
	Inverted Index	Standard Hierarchy	Extended Hierarch	Inverted Hierarch	Inverted Index	Standard Hierarchy	Extended Hierarchy	Inverted Hierarchy
INSPEC (12684 documents)	.384	.391	.360	.391	0.8	21.9	1.1	2.0
NEWS (54115 documents)	.287	.295	.214	.295	0.6	43.8	17.5	2.9

Performance Figures for Large Collections (SUN 4 computer)

Table 2