Hierarchical Clustering Algorithms for Document Datasets[*]

Ying Zhao and George Karypis

Department of Computer Science, University of Minnesota, Minneapolis, MN 55455
Technical Report #03-027

{yzhao, karypis}@cs.umn.edu

Abstract

Fast and high-quality document clustering algorithms play an important role in providing intuitive navigation and browsing mechanisms by organizing large amounts of information into a small number of meaningful clusters. In particular, clustering algorithms that build meaningful hierarchies out of large document collections are ideal tools for their interactive visualization and exploration as they provide data-views that are consistent, predictable, and at different levels of granularity. This paper focuses on document clustering algorithms that build such hierarchical solutions and (i) presents a comprehensive study of partitional and agglomerative algorithms that use different criterion functions and merging schemes, and (ii) presents a new class of clustering algorithms called *constrained agglomerative algorithms*, which combine features from both partitional and agglomerative approaches that allows them to reduce the early-stage errors made by agglomerative methods and hence improve the quality of clustering solutions. The experimental evaluation shows that, contrary to the common belief, partitional algorithms always lead to better solutions than agglomerative algorithms; making them ideal for clustering large document collections due to not only their relatively low computational requirements, but also higher clustering quality. Furthermore, the constrained agglomerative methods consistently lead to better solutions than agglomerative methods alone and for many cases they outperform partitional methods, as well.

1 Introduction

Hierarchical clustering solutions, which are in the form of trees called *dendrograms*, are of great interest for a number of application domains. Hierarchical trees provide a view of the data at different levels of abstraction. The consistency of clustering solutions at different levels of granularity allows flat partitions of different granularity to be extracted during data analysis, making them ideal for interactive exploration and visualization. In addition, there are many times when clusters have subclusters, and the hierarchical structure is indeed a natural constrain on the underlying application domain (*e.g.*, biological taxonomies, phylogenetic trees, *etc*) [14].

Hierarchical clustering solutions have been primarily obtained using agglomerative algorithms [35, 23, 15, 16, 21], in which objects are initially assigned to their own cluster and then pairs of clusters are repeatedly merged until the whole tree is formed. However, partitional algorithms [27, 20, 29, 6, 42, 19, 37, 5, 13] can also be used to obtain hierarchical clustering solutions via a sequence of repeated bisections. In recent years, various researchers have

[*] This work was supported by NSF CCR-9972519, EIA-9986042, ACI-9982274, ACI-0133464, and by Army High Performance Computing Research Center contract number DAAD19-01-2-0014. Related papers are available via WWW at URL: **http://www.cs.umn.edu/˜karypis**

recognized that partitional clustering algorithms are well-suited for clustering large document datasets due to their relatively low computational requirements [8, 24, 1, 36]. Nevertheless, there is the common belief that in terms of clustering quality, partitional algorithms are actually inferior and less effective than their agglomerative counterparts. This belief is based both on experiments with low dimensional datasets as well as a limited number of studies in which agglomerative approaches in general outperformed partitional K-means based approaches [24, 31]. For this reason, existing reviews of hierarchical document clustering methods focused mainly on agglomerative methods and entirely ignored partitional methods [40, 25]. In addition, most the previous studies evaluated various clustering methods by how well the resulting clustering solutions can improve retrieval [40, 25, 31]. The comparisons in terms of how well the resulting hierarchical trees are consistent with the existing class information are limited and only based on very few datasets [24].

This paper focuses on hierarchical document clustering algorithms and makes two key contributions. First, motivated by recent advances in partitional clustering [8, 24, 12, 5, 13], we revisited the question of whether or not agglomerative approaches generate superior hierarchical trees than partitional approaches and performed a comprehensive experimental evaluation of six partitional and nine agglomerative methods using twelve datasets derived from various sources. Specifically, for partitional clustering, we compare six recently studied criterion functions [45] that have been shown to produce high-quality solutions, whereas for agglomerative clustering, we studied three traditional merging criteria (*i.e.*, single-link, complete-link, and group average (UPGMA)) and a new set of merging criteria (introduced in this paper) that were derived from the six partitional criterion functions. Our experiments show that most of the partitional methods generate hierarchical clustering solutions that are consistently and substantially better than those produced by the various agglomerative algorithms. These result suggest that partitional clustering algorithms are ideal for obtaining hierarchical solutions of large document datasets due to not only their relatively low computational requirements, but also better performance in terms of cluster quality.

Second, we present a new class of agglomerative algorithms in which we introduce intermediate clusters obtained by partitional clustering algorithms to constrain the space over which agglomeration decisions are made. We referred to them as *constrained agglomerative algorithms*. These algorithms generate the clustering solution by using an agglomerative algorithm to build a hierarchical subtree for each partitional cluster and then agglomerate these clusters to build the final hierarchical tree. Our experimental evaluation shows that these methods consistently lead to better solutions than agglomerative methods alone and for many cases they outperform partitional methods, as well. To understand these improvements, we studied the impact that constraining has on the quality of the neighborhood of each document and found that constraining leads to purer neighborhoods as it can identify the right subspaces for the various classes.

The rest of this paper is organized as follows. Section 2 provides some information on how documents are represented and how the similarity or distance between documents is computed. Section 3 describes different criterion functions as well as criterion function optimization of hierarchical partitional algorithms. Section 4 describes the various agglomerative algorithms and their merging criteria, whereas Section 5 describes the constrained agglomerative algorithm. Section 6 provides the detailed experimental evaluation of the various hierarchical clustering methods as well as the experimental results of the constrained agglomerative algorithms. Section 7 analyzes the impact of constrained agglomerative algorithms on the quality of the neighborhoods and provides a theoretical explanation for the performance difference of some agglomerative methods. Finally, Section 8 provides some concluding remarks. A shorter version of this paper has previously appeared in [44].

2 Preliminaries

Through-out this paper we will use the symbols n, m, and k to denote the number of documents, the number of terms, and the number of clusters, respectively. We will use the symbol S to denote the set of n documents that we want to cluster, $S_1, S_2, \dots, S_k$ to denote each one of the k clusters, and $n_1, n_2, \dots, n_k$ to denote the sizes of the corresponding clusters.

The various clustering algorithms that are described in this paper use the vector-space model [33] to represent each document. In this model, each document d is considered to be a vector in the term-space. In particular, we employed the *tf-idf* term weighting model, in which each document can be represented as

$$(tf_1 \log(n/df_1), tf_2 \log(n/df_2), \dots, tf_m \log(n/df_m)).$$

where tf_i is the frequency of the ith term in the document and df_i is the number of documents that contain the ith term. To account for documents of different lengths, the length of each document vector is normalized so that it is of unit length ($\|d_{tfidf}\| = 1$), that is each document is a vector in the unit hypersphere. In the rest of the paper, we will assume that the vector representation for each document has been weighted using *tf-idf* and it has been normalized so that it is of unit length. Given a set A of documents and their corresponding vector representations, we define the ***composite*** vector D_A to be $D_A = \sum_{d \in A} d$, and the ***centroid*** vector C_A to be $C_A = \frac{D_A}{|A|}$.

In the vector-space model, the cosine similarity is the most commonly used method to compute the similarity between two documents d_i and d_j, which is defined to be $\cos(d_i, d_j) = \frac{{d_i}^t d_j}{\|d_i\| \|d_j\|}$. The cosine formula can be simplified to $\cos(d_i, d_j) = {d_i}^t d_j$, when the document vectors are of unit length. This measure becomes one if the documents are identical, and zero if there is nothing in common between them (*i.e.*, the vectors are orthogonal to each other).

3 Hierarchical Partitional Clustering Algorithm

Partitional clustering algorithms can be used to compute a hierarchical clustering solution using a repeated cluster bisectioning approach [36, 45]. In this approach, all the documents are initially partitioned into two clusters. Then, one of these clusters containing more than one document is selected and is further bisected. This process continues $n - 1$ times, leading to n leaf clusters, each containing a single document. It is easy to see that this approach builds the hierarchical agglomerative tree from top (*i.e.*, single all-inclusive cluster) to bottom (each document is in its own cluster).

3.1 Clustering Criterion Functions

A key characteristic of most partitional clustering algorithms is that they use a global criterion function whose optimization drives the entire clustering process. For those partitional clustering algorithms, the clustering problem can be stated as computing a clustering solution such that the value of a particular criterion function is optimized. In this paper we use six different clustering criterion functions that are defined in Table 1 and were recently compared and analyzed in a study presented in [45]. These functions optimize various aspects of intra-cluster similarity, inter-cluster dissimilarity, and their combinations, and represent some of the most widely-used criterion functions for document clustering.

The $\mathcal{I}_1$ criterion function (Equation 1) maximizes the sum of the average pairwise similarities (as measured by the

Table 1: Clustering Criterion Functions.

$\mathcal{I}_1$	maximize	$\sum_{r=1}^{k} n_r \left(\frac{1}{n_r^2} \sum_{d_i, d_j \in S_r} \cos(d_i, d_j) \right) = \sum_{r=1}^{k} \frac{\lVert D_r \rVert^2}{n_r}$	(1)
$\mathcal{I}_2$	maximize	$\sum_{r=1}^{k} \sum_{d_i \in S_r} \cos(d_i, C_r) = \sum_{r=1}^{k} \lVert D_r \rVert$	(2)
$\mathcal{E}_1$	minimize	$\sum_{r=1}^{k} n_r \cos(C_r, C) = \sum_{r=1}^{k} n_r \frac{{D_r}^t D}{\lVert D_r \rVert}$	(3)
$\mathcal{H}_1$	maximize	$\frac{\mathcal{I}_1}{\mathcal{E}_1} = \frac{\sum_{r=1}^{k} \lVert D_r \rVert^2 / n_r}{\sum_{r=1}^{k} n_r {D_r}^t D / \lVert D_r \rVert}$	(4)
$\mathcal{H}_2$	maximize	$\frac{\mathcal{I}_2}{\mathcal{E}_1} = \frac{\sum_{r=1}^{k} \lVert D_r \rVert}{\sum_{r=1}^{k} n_r {D_r}^t D / \lVert D_r \rVert}$	(5)
$\mathcal{G}_1$	minimize	$\sum_{r=1}^{k} \frac{\text{cut}(S_r, S - S_r)}{\sum_{d_i, d_j \in S_r} \cos(d_i, d_j)} = \sum_{r=1}^{k} \frac{{D_r}^t (D - D_r)}{\lVert D_r \rVert^2}$	(6)

cosine function) between the documents assigned to each cluster weighted according to the size of each cluster and has been used successfully for clustering document datasets [32]. The $\mathcal{I}_2$ criterion function (Equation 2) is used by the popular vector-space variant of the K-means algorithm [8, 24, 11, 36]. In this algorithm each cluster is represented by its centroid vector and the goal is to find the solution that maximizes the similarity between each document and the centroid of the cluster that is assigned to. Comparing $\mathcal{I}_1$ and $\mathcal{I}_2$ we see that the essential difference between them is that $\mathcal{I}_2$ scales the within-cluster similarity by the $\|D_r\|$ term as opposed to the n_r term used by $\mathcal{I}_1$. $\|D_r\|$ is the square-root of the pairwise similarity between all the document in S_r and will tend to emphasize clusters whose documents have smaller pairwise similarities compared to clusters with higher pairwise similarities.

The $\mathcal{E}_1$ criterion function (Equation 3) computes the clustering by finding a solution that separates the documents of each cluster from the entire collection. Specifically, it tries to minimize the cosine between the centroid vector of each cluster and the centroid vector of the entire collection. The contribution of each cluster is weighted proportionally to its size so that larger clusters will weight higher in the overall clustering solution. $\mathcal{E}_1$ was motivated by multiple discriminant analysis and is similar to minimizing the trace of the between-cluster scatter matrix [14].

The criterion functions that we described so far, view each document as a multidimensional vector. An alternate way of modeling the relations between documents is to use graphs. The document-to-document similarity graph G_s [2]is obtained by treating the pairwise similarity matrix of the dataset as the adjacency matrix of G_s. Viewing the documents in this fashion, a number of edge-cut-based criterion functions can be used to cluster document datasets [7, 17, 34, 13, 43, 10]. The $\mathcal{G}_1$ (Equations 6) function [13] views the clustering process as that of partitioning the documents into groups that minimize the edge-cut of each partition. However, because this edge-cut-based criterion function may have trivial solutions the edge-cut of each cluster is scaled by the sum of the cluster's internal edges [13]. Note that $\text{cut}(S_r, S - S_r)$ in Equation 6 is the edge-cut between the vertices in S_r and the rest of the vertices $S - S_r$, and can be re-written as ${D_r}^t(D - D_r)$ since the similarity between documents is measured using the cosine function.

3.2 Criterion Function Optimization

Our partitional algorithm uses an approach inspired by the K-means algorithm to optimize each one of the above criterion functions, and is similar to that used in [36, 45]. Initially, a random pair of documents is selected from the collection to act as the *seeds* of the two clusters. Then, for each document, its similarity to these two seeds is computed and it is assigned to the cluster corresponding to its most similar seed. This forms the initial two-way clustering. This clustering is then repeatedly refined so that it optimizes the desired clustering criterion function.

The refinement strategy that we used consists of a number of iterations. During each iteration, the documents are visited in a random order. For each document, d_i, we compute the change in the value of the criterion function obtained by moving d_i to the other cluster. If there exist some moves that lead to an improvement in the overall value of the criterion function, then d_i is moved to the cluster that leads to the highest improvement. If no such move exists, d_i remains in the cluster that it already belongs to. The refinement phase ends as soon as we perform an iteration in which no documents are moved between clusters. Note that unlike the traditional refinement approach used by K-means type of algorithms, the above algorithm moves a document as soon as it is determined that it will lead to an improvement in the value of the criterion function. This type of refinement algorithms are often called *incremental* [14]. Since each move directly optimizes the particular criterion function, this refinement strategy always converges to a local minimum. Furthermore, because the various criterion functions that use this refinement strategy are defined in terms of cluster composite and centroid vectors, the change in the value of the criterion functions as a result of single document moves can be computed efficiently.

The greedy nature of the refinement algorithm does not guarantee that it will converge to a global minimum, and the local minimum solution it obtains depends on the particular set of seed documents that were selected during the initial clustering. To eliminate some of this sensitivity, the overall process is repeated a number of times. That is, we compute N different clustering solutions (*i.e.*, initial clustering followed by cluster refinement), and the one that achieves the best value for the particular criterion function is kept. In all of our experiments, we used $N = 10$. For the rest of this discussion when we refer to the clustering solution we will mean the solution that was obtained by selecting the best out of these N potentially different solutions.

3.3 Cluster Selection

A key step in the repeated cluster bisectioning approach is the method used to select which cluster to be bisected next. In our study, we experimented with two different cluster selection methods. The first method uses the simple strategy of bisecting the largest cluster available at that point of the clustering solution. Our earlier experience with this approach showed that it leads to reasonably good and balanced clustering solutions [36, 45]. However, its limitation is that it cannot gracefully operate in datasets in which the natural clusters are of different sizes, as it will tend to partition those larger clusters first. To overcome this problem and obtain more natural hierarchical solutions we developed a method that among the current k clusters, selects the cluster which leads to the $k + 1$ clustering solution that optimizes the value of the particular criterion function (among the different k choices). Our experiments showed that this approach performs somewhat better than the previous scheme, and is the method that we used in the experiments presented in Section 6.

3.4 Computational Complexity

One of the advantages of our partitional algorithm and that of other similar partitional algorithms, is that it has relatively low computational requirements. A two-way clustering of a set of documents can be computed in time linear on the number of documents, as in most cases the number of iterations required by the greedy refinement algorithm is small (less than 20), and is to a large extent independent on the number of documents. Now if we assume that during each bisection step, the resulting clusters are reasonably balanced (*i.e.*, each cluster contains a fraction of the original documents), then the overall amount of time required to compute all $n-1$ bisections is $O(n \log n)$.

4 Hierarchical Agglomerative Clustering Algorithm

Unlike the partitional algorithms that build the hierarchical solution for top to bottom, agglomerative algorithms build the solution by initially assigning each document to its own cluster and then repeatedly selecting and merging pairs of clusters, to obtain a single all-inclusive cluster. Thus, agglomerative algorithms build the tree from bottom (*i.e.*, its leaves) toward the top (*i.e.*, root).

4.1 Cluster Selection Schemes

The key parameter in agglomerative algorithms is the method used to determine the pair of clusters to be merged at each step. In most agglomerative algorithms, this is accomplished by selecting the most *similar* pair of clusters, and numerous approaches have been developed for computing the similarity between two clusters[35, 23, 20, 15, 16, 21]. In our study we used the single-link, complete-link, and UPGMA schemes as well as the various partitional criterion functions described in Section 3.1.

The single-link [35] scheme measures the similarity of two clusters by the maximum similarity between the documents from each cluster. That is, the similarity between two clusters S_i and S_j is given by

$$\text{sim}_{\text{single-link}}(S_i, S_j) = \max_{d_i \in S_i,\ d_j \in S_j} \{\cos(d_i, d_j)\}. \tag{7}$$

In contrast, the complete-link scheme [23] uses the minimum similarity between a pair of documents to measure the same similarity. That is,

$$\text{sim}_{\text{complete-link}}(S_i, S_j) = \min_{d_i \in S_i,\ d_j \in S_j} \{\cos(d_i, d_j)\}. \tag{8}$$

In general, both the single- and the complete-link approaches do not work very well because they either base their decisions on limited amount of information (single-link) or they assume that all the documents in the cluster are very similar to each other (complete-link approach). The UPGMA scheme [20] (also known as group average) overcomes these problems by measuring the similarity of two clusters as the average of the pairwise similarity of the documents from each cluster. That is,

$$\text{sim}_{\text{UPGMA}}(S_i, S_j) = \frac{1}{n_i n_j} \sum_{d_i \in S_i,\ d_j \in S_j} \cos(d_i, d_j) = \frac{{D_i}^t D_j}{n_i n_j}. \tag{9}$$

The partitional criterion functions, described in Section 3.1, can be converted into cluster selection schemes for agglomerative clustering using the general framework of stepwise optimization [14] as follows. Consider an n-document dataset and the clustering solution that has been computed after performing l merging steps. This solution will contain exactly $n-l$ clusters, as each merging step reduces the number of clusters by one. Now, given this $(n-l)$-way clustering solution, the pair of clusters that is selected to be merged next, is the one that leads to an $(n-l-1)$-way solution that optimizes the particular criterion function. That is, each one of the $(n-l) \times (n-l-1)/2$ pairs of possible merges is evaluated, and the one that leads to a clustering solution that has the maximum (or minimum) value of the particular criterion function is selected. Thus, the criterion function is *locally* optimized within this particular stage of the agglomerative algorithm. This process continues until the entire agglomerative tree has been obtained.

4.2 Computational Complexity

There are two main computationally expensive steps in agglomerative clustering. The first step is the computation of the pairwise similarity between all the documents in the data set. The complexity of this step is, in general, $O(n^2)$ because the average number of terms in each document is small and independent of n.

The second step is the repeated selection of the pair of most similar clusters or the pair of clusters that best optimizes the criterion function. A naive way of performing this step is to recompute the gains achieved by merging each pair of clusters after each level of the agglomeration, and select the most promising pair. During the lth agglomeration step, this will require $O((n-l)^2)$ time, leading to an overall complexity of $O(n^3)$. Fortunately, the complexity of this step can be reduced for single-link, complete-link, UPGMA, $\mathcal{I}_1$, $\mathcal{I}_2$, $\mathcal{E}_1$, and $\mathcal{G}_1$. This is because the pair-wise similarities or the improvements in the value of the criterion function achieved by merging a pair of clusters i and j does not change during the different agglomerative steps, as long as i or j is not selected to be merged. Consequently, the different similarities or gains in the value of the criterion function can be computed once for each pair of clusters and inserted into a priority queue. As a pair of clusters i and j is selected to be merged to form cluster p, then the priority queue is updated so that any gains corresponding to cluster pairs involving either i or j are removed, and the gains of merging the rest of the clusters with the newly formed cluster p are inserted. During the lth agglomeration step, that involves $O(n-l)$ priority queue delete and insert operations. If the priority queue is implemented using a binary heap, the total complexity of these operations is $O((n-l)\log(n-l))$, and the overall complexity over the $n-1$ agglomeration steps is $O(n^2 \log n)$.

Unfortunately, the original complexity of $O(n^3)$ of the naive approach cannot be reduced for the $\mathcal{H}_1$ and $\mathcal{H}_2$ criterion functions, because the improvement in the overall value of the criterion function when a pair of clusters i and j is merged might change for all pairs of clusters. As a result, they cannot be pre-computed and inserted into a priority queue.

5 Constrained Agglomerative Clustering

One of the advantages of partitional clustering algorithms is that they use information about the entire collection of documents when they partition the dataset into a certain number of clusters. On the other hand, the clustering decisions made by agglomerative algorithms are local in nature. This local nature has both its advantages as well as its disadvantages. The advantage is that it is easy for them to group together documents that form small and reasonably cohesive clusters, a task in which partitional algorithms may fail as they may split such documents across cluster boundaries early during the partitional clustering process (especially when clustering large collections). However, its disadvantage is that if the documents are not part of particularly cohesive groups, then the initial merging decisions may contain some errors, which will tend to be multiplied as the agglomeration progresses. This is especially true for the cases in which there are a large number of equally good merging alternatives for each cluster.

One potential way of eliminating this type of errors is to use a partitional clustering algorithm to constrain the space over which agglomeration decisions are made by only allowing each document to merge with other documents that are part of the same partitionally discovered cluster. In this approach, a partitional clustering algorithm is used to compute a k-way clustering solution in the same way as computing a two-way clustering described in Section 3. Then, each of these clusters, referred as *constraint clusters*, is treated as a separate collection and an agglomerative algorithm is used to build a tree for each one of them. Finally, the k different trees are combined into a single tree by merging them using an agglomerative algorithm that treats the documents of each subtree as a cluster that has already been formed during agglomeration. The advantage of this approach is that it is able to benefit from the global *view* of the collection used by partitional algorithms and the local *view* used by agglomerative algorithms. An additional advantage is that the computational complexity of constrained clustering is $O(k((n/k)^2 \log(n/k)) + k^2 \log k)$, where k is the number of constraint clusters. If k is reasonably large, *e.g.*, k equals $\sqrt{n}$, the original complexity of $O(n^2 \log n)$ for agglomerative algorithms is reduced to $O(n^{2/3} \log n)$.

6 Experimental Results

We experimentally evaluated the performance of the various clustering methods to obtain hierarchical solutions using a number of different datasets. In the rest of this section we first describe the various datasets and our experimental methodology followed by a description of the experimental results. The datasets as well as the various algorithms are available in the CLUTO clustering toolkit [22], which can be downloaded from *http://www.cs.umn.edu/˜cluto*.

6.1 Document Collections

We used a total of eleven different datasets, whose general characteristics are summarized in Table 2. The smallest of these datasets contains 878 documents and the largest contains 4,069 documents. To ensure diversity in the datasets, we obtained them from different sources. For all datasets we used a stop-list to remove common words and the words were stemmed using Porter's suffix-stripping algorithm [30]. Moreover, any term that occurs in fewer than two documents was eliminated.

Table 2: Summary of data sets used to evaluate the various clustering criterion functions.

Data	Source	# of documents	# of terms	# of classes
fbis	FBIS (TREC)	2463	12674	17
hitech	San Jose Mercury (TREC)	2301	13170	6
reviews	San Jose Mercury (TREC)	4069	23220	5
la1	LA Times (TREC)	3204	21604	6
tr31	TREC	927	10128	7
tr41	TREC	878	7454	10
re0	Reuters-21578	1504	2886	13
re1	Reuters-21578	1657	3758	25
k1a	WebACE	2340	13879	20
k1b	WebACE	2340	13879	6
wap	WebACE	1560	8460	20

The *fbis* dataset is from the Foreign Broadcast Information Service data of TREC-5 [38], and the classes correspond to the categorization used in that collection. The *hitech* and *reviews* datasets were derived from the San Jose Mercury newspaper articles that are distributed as part of the TREC collection (TIPSTER Vol. 3). Each one of these datasets was constructed by selecting documents that are part of certain topics in which the various articles were categorized (based on the *DESCRIPT* tag). The *hitech* dataset contains documents about computers, electronics, health, medical, research, and technology; and the *reviews* dataset contained documents about food, movies, music, radio, and restaurants. In selecting these documents we ensured that no two documents share the same *DESCRIPT* tag (which can contain multiple categories). The *la1* dataset was obtained from the articles of the Los Angeles Times that was used in TREC-5 [38]. The categories correspond to the *desk* of the paper that each article appeared and include documents from the entertainment, financial, foreign, metro, national, and sports desks. Datasets *tr31* and *tr41* were derived from TREC-5 [38], TREC-6 [38], and TREC-7 [38] collections. The classes of these datasets correspond to the documents that were judged relevant to particular queries. The datasets *re0* and *re1* are from Reuters-21578 text categorization test collection Distribution 1.0 [26]. We divided the labels into two sets and constructed datasets accordingly. For each dataset, we selected documents that have a single label. Finally, the datasets *k1a*, *k1b*, and *wap* are from the WebACE project [28, 18, 3, 4]. Each document corresponds to a web page listed in the subject hierarchy of Yahoo! [41]. The datasets *k1a* and *k1b* contain exactly the same set of documents but they differ in how the documents were assigned to different classes. In particular, *k1a* contains a finer-grain categorization than that contained in *k1b*.

6.2 Experimental Methodology and Metrics

The quality of a clustering solution was determined by analyzing how the documents of the different classes are distributed in the nodes of the hierarchical trees produced by the various algorithms and was measured using two different metrics.

The first is the *FScore measure* [24] that identifies for each class of documents the node in the hierarchical tree that best represents it and then measures the overall quality of the tree by evaluating this subset of clusters. In determining how well a cluster represents a particular class, the FScore measure treats each cluster as if it was the result of a query for which all the documents of the class were the desired set of relevant documents. Given such a view, then the suitability of the cluster to the class is measured using the F value [39] that combines the standard precision and recall functions used in information retrieval [39]. Specifically, given a particular class L_r of size n_r and a particular cluster S_i of size n_i, suppose n_{ri} documents in the cluster S_i belong to L_r, then the F value of this class and cluster is defined to be

$$F(L_r, S_i) = \frac{2 * R(L_r, S_i) * P(L_r, S_i)}{R(L_r, S_i) + P(L_r, S_i)},$$

where $R(L_r, S_i) = n_{ri}/n_r$ is the recall value and $P(L_r, S_i) = n_{ri}/n_i$ is the precision value defined for the class L_r and the cluster S_i. The FScore of class L_r is the maximum F value attained at any node in the hierarchical tree T. That is,

$$\mathit{FScore}(L_r) = \max_{S_i \in T} F(L_r, S_i).$$

The FScore of the entire hierarchical tree is defined to be the sum of the individual class specific FScores weighted according to the class size. That is,

$$\mathit{FScore} = \sum_{r=1}^{c} \frac{n_r}{n} \mathit{FScore}(L_r),$$

where c is the total number of classes. A perfect clustering solution will be the one in which every class has a corresponding cluster containing the same set of documents in the resulting hierarchical tree, in which case the FScore will be one. In general, the higher the FScore values, the better the clustering solution is.

The second is the *entropy measure* that unlike FScore, which evaluates the overall quality of a hierarchical tree using only a small subset of its nodes, it takes into account the distribution of the documents in all the nodes of the tree. Given a particular node S_r of size n_r, the entropy of this node is defined to be

$$E(S_r) = -\frac{1}{\log q} \sum_{i=1}^{q} \frac{n_r^i}{n_r} \log \frac{n_r^i}{n_r},$$

where q is the number of classes in the dataset and n_r^i is the number of documents of the ith class that were assigned to the rth node. Then, the ***entropy*** of the entire tree is defined to be

$$E(T) = \sum_{r=1}^{t} \frac{1}{t} E(S_r),$$

where t is the number of nodes of the hierarchical tree T. In general, the lower the entropy values the better the clustering solution is. Note that unlike the FScore measure and the previous uses of the entropy measure to evaluate k-way clustering solutions in [36, 45], the above definition of the entropy measure is an unweighted average over all the nodes of the tree. This was motivated by the fact that due to the nature of the problem, the clusters corresponding to the top levels of the tree will be both large and in general have poor entropy values. Consequently, if the entropies are averaged in a cluster-size weighted fashion, these top-level nodes will be dominating, potentially obscuring any differences that may exist at the lower-level tree nodes.

When comparing different hierarchical algorithms for one dataset it is hard to determine which one is better based only on the results of one run. In order to statistically compare the performance of two algorithms on one dataset, we sampled each dataset ten times and all comparisons and conclusions are based on the results of the sampled datasets. Specifically, for each one of the eleven datasets we created ten sampled subsets by randomly selecting 70% of the documents from the original dataset. For each one of the subsets, we obtained clustering solutions using the various hierarchical algorithms described in Sections 3–5. In addition, for most of our comparisons we used t-test based statistical tests to determine the significance of the results.

6.3 Comparison of Partitional and Agglomerative Trees

Our first set of experiments was focused on evaluating the quality of the hierarchical clustering solutions produced by various agglomerative and partitional algorithms. For agglomerative algorithms, nine cluster selection schemes or criterion functions were tested including the six criterion functions discussed in Section 3.1, and the three traditional cluster selection schemes (*i.e.*, single-link, complete-link and UPGMA). We named this set of agglomerative methods directly with the name of the criterion function or selection scheme, *e.g.*, "$\mathcal{I}_1$" means the agglomerative clustering method with $\mathcal{I}_1$ as the criterion function and "UPGMA" means the agglomerative clustering method with UPGMA as the selection scheme. We also evaluated various repeated bisectioning algorithms using the six criterion functions discussed in Section 3.1. We named this set of partitional methods by adding a letter "p" in front of the name of the criterion function, *e.g.*, "p$\mathcal{I}_1$" means the repeated bisection clustering method with $\mathcal{I}_1$ as the criterion function.

For each dataset and hierarchical clustering method, we first calculated the average of the FScore/entropy values of the clustering solutions obtained for the ten sampled subsets of that dataset, which is referred as the FScore/entropy value for the dataset. Then, we summarized these FScore/entropy values in two ways, one is by comparing each pair of methods to see which method outperforms the other for most of the datasets and the other is by looking at the average performance of each method over the entire set of datasets.

Dominance Matrix Our first way of summarizing the results is to create a 15×15 *dominance matrix* that is shown in Table 3. The rows and columns of this matrix correspond to the various methods whereas its values correspond to the number of datasets for which the method of the row outperforms the method of the column. For example, the value in the entry of the row $\mathcal{I}_2$ and the column $\mathcal{E}_1$ is eight, which means for eight out of the eleven datasets, the $\mathcal{I}_2$ method outperforms the $\mathcal{E}_1$ method. We also tested the significance of the comparisons between various methods using the paired t-test [9] based on the FScore values for the eleven datasets. These results are shown in the second subtable of Table 3. A similar comparison based on the entropy measure is presented in Table 4.

A number of observations can be made by analyzing the results in Table 3. First, partitional methods outperform agglomerative methods. By looking at the left bottom part of the dominance matrix, we can see that all the entries are close to eleven (except for some entries in the row of p$\mathcal{I}_1$), indicating that each partitional method performs better than agglomerative methods for all or most of the datasets. Second, by looking at the submatrix of the comparisons within agglomerative methods (*i.e.*, the left top part of the dominance matrix), we can see that the UPGMA method performs the best followed by $\mathcal{I}_2$, $\mathcal{H}_1$ and $\mathcal{H}_2$, whereas slink, clink and $\mathcal{I}_1$ perform the worst. Note that $\mathcal{I}_1$ and UPGMA are

Table 3: Dominance and statistical significance matrix for various hierarchical clustering methods evaluated by FScore. Note that "$\gg$" ("$\ll$") indicates that schemes of the row perform significantly better (worse) than the schemes of the column, and $<$ ($>$) indicates the relationship is not significant. For all statistical significance tests, p-value=0.05.

Dominance Matrix

	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$	UPGMA	slink	clink	p$\mathcal{E}_1$	p$\mathcal{G}_1$	p$\mathcal{H}_1$	p$\mathcal{H}_2$	p$\mathcal{I}_1$	p$\mathcal{I}_2$
$\mathcal{E}_1$	0	7	1	2	6	1	0	11	11	0	0	0	0	0	0
$\mathcal{G}_1$	4	0	2	4	8	1	0	11	9	0	0	0	0	0	0
$\mathcal{H}_1$	10	9	0	8	11	6	1	11	11	0	0	0	1	1	0
$\mathcal{H}_2$	9	7	3	0	10	3	0	11	11	0	0	0	0	1	0
$\mathcal{I}_1$	4	3	0	1	0	0	0	11	10	0	0	0	0	0	0
$\mathcal{I}_2$	10	10	5	7	11	0	0	11	11	0	1	1	1	3	1
UPGMA	11	11	10	11	11	11	0	11	11	3	1	1	2	5	1
slink	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
clink	0	2	0	0	1	0	0	10	0	0	0	0	0	0	0
p$\mathcal{E}_1$	11	11	11	11	11	10	8	11	11	0	3	3	2	8	2
p$\mathcal{G}_1$	11	11	11	11	11	10	10	11	11	8	0	5	5	8	1
p$\mathcal{H}_1$	11	11	11	11	11	10	10	11	11	8	6	0	4	9	2
p$\mathcal{H}_2$	11	11	10	10	11	10	9	11	11	9	5	5	0	9	4
p$\mathcal{I}_1$	11	11	9	10	11	8	6	11	11	3	3	2	2	0	3
p$\mathcal{I}_2$	11	11	11	11	11	10	10	11	11	9	10	8	7	8	0

Statistical Significance Matrix

	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$	UPGMA	slink	clink	p$\mathcal{E}_1$	p$\mathcal{G}_1$	p$\mathcal{H}_1$	p$\mathcal{H}_2$	p$\mathcal{I}_1$	p$\mathcal{I}_2$
$\mathcal{E}_1$	-	$>$	$\ll$	$\ll$	$>$	$\ll$	$\ll$	$\gg$	$\gg$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
$\mathcal{G}_1$	$<$	-	$\ll$	$\ll$	$\gg$	$\ll$	$\ll$	$\gg$	$\gg$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
$\mathcal{H}_1$	$\gg$	$\gg$	-	$>$	$\gg$	$<$	$\ll$	$\gg$	$\gg$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
$\mathcal{H}_2$	$\gg$	$\gg$	$<$	-	$\gg$	$<$	$\ll$	$\gg$	$\gg$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
$\mathcal{I}_1$	$<$	$\ll$	$\ll$	$\ll$	-	$\ll$	$\ll$	$\gg$	$\gg$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
$\mathcal{I}_2$	$\gg$	$\gg$	$>$	$>$	$\gg$	-	$\ll$	$\gg$	$\gg$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
UPGMA	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	-	$\gg$	$\gg$	$\ll$	$\ll$	$\ll$	$\ll$	$<$	$\ll$
slink	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	-	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
clink	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\gg$	-	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
p$\mathcal{E}_1$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	-	$<$	$<$	$\ll$	$\gg$	$<$
p$\mathcal{G}_1$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$>$	-	$>$	$<$	$\gg$	$\ll$
p$\mathcal{H}_1$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$>$	$<$	-	$<$	$\gg$	$\ll$
p$\mathcal{H}_2$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$>$	$>$	-	$\gg$	$<$
p$\mathcal{I}_1$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$>$	$\gg$	$\gg$	$\ll$	$\ll$	$\ll$	$\ll$	-	$\ll$
p$\mathcal{I}_2$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$>$	$\gg$	$\gg$	$>$	$\gg$	-

motivated similarly, but perform very differently. We will discuss this trend in detail later in Section 7. Third, from the submatrix of the comparisons within partitional methods (*i.e.*, the right bottom part of the dominance matrix), we can see that p$\mathcal{I}_2$ leads to better solutions than the other partitional methods for most of the datasets followed by p$\mathcal{H}_2$, whereas p$\mathcal{I}_1$ perform worse than the other partitional methods for most of the datasets. Also, as the paired t-test results show, the relative advantage of the partitional methods over the agglomerative methods and UPGMA over the rest of the agglomerative methods is statistically significant.

Relative FScores/Entropy To quantitatively compare the relative performance of the various methods we also summarized the results by averaging the *relative FScores* for each method over the eleven different datasets. For each dataset, we divided the FScore obtained by a particular method by the largest FScore (*i.e.*, corresponding to the best performing scheme) obtained for that particular dataset over the 15 methods. These ratios, referred to as *relative FScores*, are less sensitive than the actual FScore values, and were averaged over the various datasets. Since higher FScore values are better, all these relative FScore values are less than one. A method having an *average relative FScore* close to 1.0 indicates that this method performs the best for most of the datasets. On the other hand, if the average relative FScore is low, then this method performs poorly. The results of the average relative FScores for various hierarchical clustering methods are shown in Table 5(a). The entries that are bold-faced correspond to the methods that perform the best and the entries that are underlined correspond to the methods that perform the best among agglomerative methods or partitional methods alone. A similar comparison based on the entropy measure is

Table 4: Dominance and statistical significance matrix for various hierarchical clustering methods evaluated by entropy. Note that “$\gg$” (“$\ll$") indicates that schemes of the row perform significantly better (worse) than the schemes of the column, and $<$ ($>$) indicates the relationship is not significant. For all statistical significance tests, p-value=0.05.

Dominance Matrix

	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$	UPGMA	slink	clink	p$\mathcal{E}_1$	p$\mathcal{G}_1$	p$\mathcal{H}_1$	p$\mathcal{H}_2$	p$\mathcal{I}_1$	p$\mathcal{I}_2$
$\mathcal{E}_1$	0	9	7	2	9	6	3	11	4	0	0	0	0	1	0
$\mathcal{G}_1$	1	0	0	1	3	0	0	11	0	0	0	0	0	0	0
$\mathcal{H}_1$	2	11	0	1	9	0	1	11	2	0	0	0	0	0	0
$\mathcal{H}_2$	3	10	8	0	9	7	3	11	4	0	0	0	0	0	0
$\mathcal{I}_1$	1	7	1	1	0	0	0	11	1	0	0	0	0	0	0
$\mathcal{I}_2$	3	11	3	2	11	0	1	11	3	0	0	0	0	0	0
UPGMA	8	11	9	8	11	9	0	11	8	1	0	0	0	4	0
slink	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
clink	7	11	8	7	8	8	1	11	0	0	0	0	0	1	0
p$\mathcal{E}_1$	11	11	11	11	11	11	10	11	11	0	2	6	2	11	1
p$\mathcal{G}_1$	11	11	11	11	11	11	11	11	11	7	0	6	5	11	2
p$\mathcal{H}_1$	11	11	11	11	11	11	11	11	11	3	1	0	3	11	0
p$\mathcal{H}_2$	11	11	11	11	11	11	10	11	11	7	4	7	0	11	1
p$\mathcal{I}_1$	9	11	10	9	11	10	7	11	9	0	0	0	0	0	0
p$\mathcal{I}_2$	11	11	11	11	11	11	11	11	11	10	7	10	7	11	0

Statistical Significance Matrix

	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$	UPGMA	slink	clink	p$\mathcal{E}_1$	p$\mathcal{G}_1$	p$\mathcal{H}_1$	p$\mathcal{H}_2$	p$\mathcal{I}_1$	p$\mathcal{I}_2$
$\mathcal{E}_1$	-	$\gg$	$>$	$>$	$\gg$	$>$	$<$	$\gg$	$<$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
$\mathcal{G}_1$	$\ll$	-	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\gg$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
$\mathcal{H}_1$	$<$	$\gg$	-	$\ll$	$\gg$	$<$	$\ll$	$\gg$	$<$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
$\mathcal{H}_2$	$<$	$\gg$	$\gg$	-	$\gg$	$>$	$\ll$	$\gg$	$<$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
$\mathcal{I}_1$	$\ll$	$\gg$	$\ll$	$\ll$	-	$\ll$	$\ll$	$\gg$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
$\mathcal{I}_2$	$<$	$\gg$	$>$	$<$	$\gg$	-	$\ll$	$\gg$	$<$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
UPGMA	$>$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	-	$\gg$	$\gg$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
slink	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	-	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
clink	$>$	$\gg$	$>$	$>$	$\gg$	$>$	$\ll$	$\gg$	-	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$	$\ll$
p$\mathcal{E}_1$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	-	$<$	$>$	$<$	$\gg$	$<$
p$\mathcal{G}_1$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$>$	-	$\gg$	$>$	$\gg$	$<$
p$\mathcal{H}_1$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$<$	$\ll$	-	$\ll$	$\gg$	$\ll$
p$\mathcal{H}_2$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$>$	$<$	$\gg$	-	$\gg$	$<$
p$\mathcal{I}_1$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\ll$	$\ll$	$\ll$	$\ll$	-	$\ll$
p$\mathcal{I}_2$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$>$	$>$	$\gg$	$>$	$\gg$	-

presented in Table 5(b).

Looking at the results in Table 5(a) we can see that in general they are in agreement with the results presented earlier in Tables 3 and 4. First, the repeated bisection method with the $\mathcal{I}_2$ criterion function (*i.e.*, “p$\mathcal{I}_2$”) leads to the best solutions for most of the datasets. Over the entire set of experiments, this method is either the best or always within 6% of the best solution. On average, the p$\mathcal{I}_2$ method outperforms the other partitional methods and agglomerative methods by 1%–5% and 6%–34%, respectively. Second, the UPGMA method performs the best among agglomerative methods. On average, UPGMA outperform the other agglomerative methods by 4%–28%. Third, partitional methods outperform agglomerative methods. Except for the p$\mathcal{I}_1$ method, each one of the remaining five partitional methods on the average performs better than all the nine agglomerative methods by at least 5%. Fourth, single-link, complete-link and $\mathcal{I}_1$ performed poorly among agglomerative methods and p$\mathcal{I}_1$ performs the worst among partitional methods. Fifth, $\mathcal{I}_2$, $\mathcal{H}_1$ and $\mathcal{H}_2$ are the agglomerative methods that lead to the second best hierarchical clustering solutions among agglomerative methods, whereas p$\mathcal{H}_2$ and p$\mathcal{E}_1$ are the partitional methods that lead to the second best hierarchical clustering solutions among partitional methods.

Finally, comparing the relative performance of the various schemes using the two different quality measures, we can see that in most cases they are in agreement with each other. The only exception is that the relative performance in terms of entropy values achieved by clink, $\mathcal{E}_1$ and p$\mathcal{E}_1$ is somewhat higher. The reason for that is because these schemes tend to lead to more balanced hierarchical trees [14, 45] and because of this structure they have better entropies. To see this consider the example shown in Figure 1. Suppose A, B, C and D are documents of different classes clustered

Table 5: The relative FScore/entropy values averaged over the different datasets for the hierarchical clustering solutions obtained via various hierarchical clustering methods.

(a) FScore Measure

	Agglomerative Methods									Partitional Methods					
	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$	UPGMA	slink	clink	p$\mathcal{E}_1$	p$\mathcal{G}_1$	p$\mathcal{H}_1$	p$\mathcal{H}_2$	p$\mathcal{I}_1$	p$\mathcal{I}_2$
Average	0.855	0.855	0.889	0.879	0.836	0.890	0.929	0.649	0.760	0.968	0.971	0.972	0.978	0.932	**0.987**

(b) Entropy Measure

	Agglomerative Methods									Partitional Methods					
	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$	UPGMA	slink	clink	p$\mathcal{E}_1$	p$\mathcal{G}_1$	p$\mathcal{H}_1$	p$\mathcal{H}_2$	p$\mathcal{I}_1$	p$\mathcal{I}_2$
Average	1.381	1.469	1.381	1.370	1.434	1.369	1.277	3.589	1.340	1.046	1.035	1.072	1.039	1.186	**1.027**

using the two hierarchical trees shown in Figures 1(a) and 1(b), such that the tree in Figure 1(b) is more balanced. Since only the intermediate nodes (gray nodes) are different between these two trees, and because the entropy of the "A|B|C" node is higher than that of the "C|D" node, the more balanced tree of Figure 1(b) will have lower entropy.

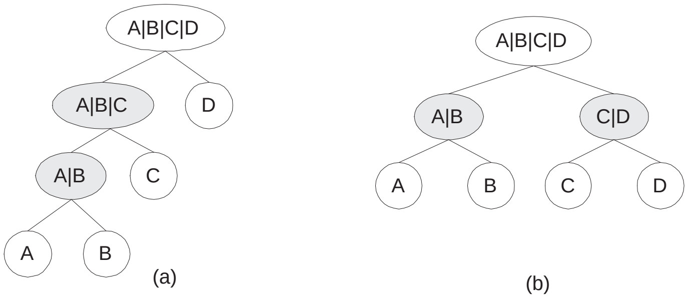


Figure 1: An example of how more balanced trees tend to have better entropies.

6.4 Constrained Agglomerative Trees

Our second set of experiments was focused on evaluating the constrained agglomerative clustering methods. These results were obtained by first using the various partitional methods to find the constraint cluster and then using UPGMA as the agglomerative scheme to construct the final hierarchical solutions as described in Section 5. UPGMA was selected because it performed the best among the various agglomerative schemes. For each dataset and criterion function we performed four experiments on constrained agglomerative methods with 10, 20, $n/40$ and $n/20$ constraint clusters, where n is the total number of documents in each dataset, and compared them against UPGMA and the corresponding partitional scheme (*i.e.*, using the same criterion function) using the FScore measure. The statistical significance of the comparisons was tested using two samples mean test [9] based on the FScore values obtained by the two clustering schemes for the ten sampled subsets of each dataset. The null hypothesis is that the two clustering schemes have the same performance, and the alternative hypothesis is that a certain constrained agglomerative method outperforms UPGMA or the corresponding partitional method.

The results of testing whether constrained agglomerative methods outperformed UPGMA and the corresponding partitional methods are summarized in Table 6. The columns of this table correspond to the criterion function used in constrained agglomerative methods and partitional methods. The first four rows correspond to the comparisons between the constrained agglomerative methods with 10, 20, $n/40$ and $n/20$ constraint clusters and UPGMA, whereas the last four rows correspond to the comparisons between the constrained agglomerative methods and the correspond-

Table 6: Comparison of constrained agglomerative methods with 10, 20, $n/40$ and $n/20$ constraint clusters with UPGMA and repeated bisection methods with various criterion functions.

Method	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{E}_1$	$\mathcal{G}_1$
10 vs. UPGMA	54.5%	100%	81.8%	81.8%	100%	81.8%
20 vs. UPGMA	45.5%	100%	81.8%	90.9%	100%	90.9%
$n/40$ vs. UPGMA	77.7%	100%	100%	88.9%	100%	100%
$n/20$ vs. UPGMA	72.7%	100%	90.9%	90.9%	100%	100%
10 vs. rb	45.5%	54.5%	81.8%	36.4%	36.4%	45.5%
20 vs. rb	63.6%	54.5%	90.9%	54.5%	36.4%	72.7%
$n/40$ vs. rb	66.6%	44.5%	77.7%	55.6%	66.6%	66.6%
$n/20$ vs. rb	90.9%	54.5%	81.8%	63.6%	63.6%	54.5%

ing partitional methods. The value shown in each entry is the proportion of the datasets, for which the constrained agglomerative method significantly (*i.e.*, p-value < 0.05) outperformed UPGMA or the corresponding partitional method. For example, the value of the entry of the row "$n/40$ vs. UPGMA " and the column $\mathcal{I}_1$ is 77.7%, which means for 77.7% of the datasets the constrained agglomerative method with $\mathcal{I}_1$ as the partitional criterion and $n/20$ constraint clusters statistically significantly outperformed the UPGMA method.

¿From the results in Table 6 we can see that various constrained agglomerative methods outperform the agglomerative method (UPGMA) for almost all the datasets. Moreover such improvements can be achieved even with small number of constraint clusters. Also, for many cases the constrained agglomerative methods perform even better than the corresponding partitional methods. Among the six criterion functions, $\mathcal{H}_1$ achieves the best improvement over the corresponding partitional method, whereas $\mathcal{I}_2$ achieves the least improvement.

7 Discussion

The experiments presented in Section 6 showed three interesting trends. First, various partitional methods (except for p$\mathcal{I}_1$) significantly outperform all agglomerative methods. Second, constraining the agglomeration space, even with a small number of partitional clusters, improves the hierarchical solutions obtained by agglomerative methods alone. Third, agglomeration methods with various objective functions described in Section 3.1 perform worse than the UPGMA method. For instance, both $\mathcal{I}_1$ and UPGMA try to maximize the average pairwise similarity between the documents of the discovered clusters. However, UPGMA tends to perform consistently better than $\mathcal{I}_1$. In the remainder of this section we present an analysis that explains the cause of these trends.

7.1 Analysis of the constrained agglomerative method

In order to better understand how constrained agglomerative methods benefit from partitional constraints and potentially why partitional methods perform better than agglomerative methods, we looked at the quality of the nearest neighbors of each document and how well this quality relates to the quality of the resulting hierarchical trees. We evaluated the quality of the nearest neighbors by the entropy measure defined in Section 6.2 based on the class label of each neighbor. For this study we looked at the five nearest neighbors (5-nn) of each document and used ten constraint clusters in various constrained agglomerative algorithms. However, these observations carry over to other number of nearest neighbors as well.

Quality of constrained and unconstrained neighborhoods Our first analysis compares the quality of the nearest neighbors of each document when the nearest neighbors are selected from the entire dataset (*i.e.*, there is no constraint cluster) and when the nearest neighbors are selected from the same constraint cluster as the document

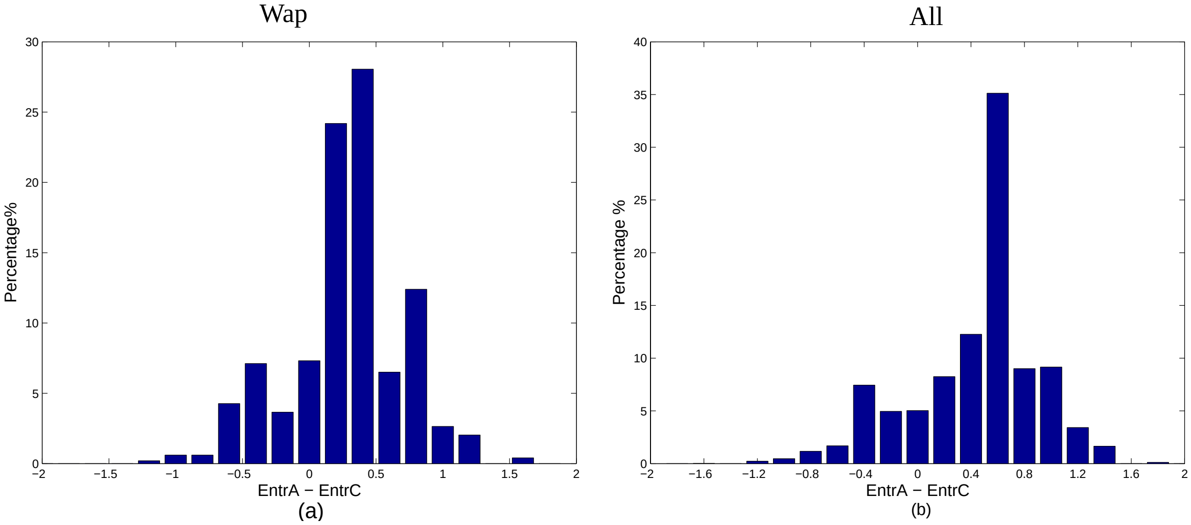


Figure 2: The distribution of the 5-nn entropy differences of each document without any constraint (EntrA) and with ten partitional cluster constraints obtained by p$\mathcal{I}_2$ (EntrC) for (a)dataset Wap and (b) all datasets.

(*i.e.*, constraints are enforced by constrained agglomerative methods). For each document we computed EntrA - EntrC, where EntrA is the entropy value of the 5-nn obtained without any constraint and EntrC is the entropy value of the 5-nn obtained after enforcing partitional constraints generated using the partitional method with $\mathcal{I}_2$. Figure 2(a) shows how these differences (EntrA - EntrC) are distributed for the Wap dataset, whereas Figure 2(b) shows the same distribution over all the datasets. The X-axis in Figure 2 represents the differences of the 5-nn entropy values (EntrA - EntrC), whereas the Y-axis represents the percentage of the documents that have the corresponding 5-nn entropy difference values. Note that since lower entropy values are better, differences that are positive (*i.e.*, bars on the right of the origin) correspond to the instances in which the constrained scheme resulted in purer 5-nn neighborhoods.

¿From these charts we can see that in about 80% of the cases the entropy values with partitional cluster constraints are lower than those without any constraint, which means that the constraints improve the quality of each document's neighborhood. Note that the quality of the nearest neighbors directly affects the overall performance of agglomerative methods, because their key operation is that of grouping together the most similar documents. As a result, a scheme that starts from purer neighborhoods will benefit the overall algorithm and we believe that this is the reason as to why the constraint agglomerative algorithms outperform the traditional agglomerative algorithms.

To verify how well these improvements in the 5-nn quality correlate with the clustering improvements achieved by the constrained agglomerative algorithms (shown in Table 6), we computed the difference in the FScore values between the UPGMA and the constrained agglomerative trees (FScoreA - FScoreC) and plotted them against the corresponding average 5-nn entropy differences (EntrA - EntrC). These plots are shown in Figure 3. Each dataset is represented by six different data points (one for each criterion function) and these points were fit with a linear least square error line. In addition, for each dataset we computed the Pearson correlation coefficient between the differences and the absolute values of these coefficients are shown in Figure 3 as well. From these results we can see that for most dataset there is indeed a high correlation between the 5-nn quality improvements and the overall improvements in cluster quality. In particular, with the exceptions of fbis and tr31, for the remaining datasets the correlation coefficients are very high (greater than 0.85). The correlation between 5-nn improvements and the overall improvements in cluster quality is somewhat weaker for fbis and tr31 as they have absolute correlation coefficients of 0.158 and 0.574, respectively.

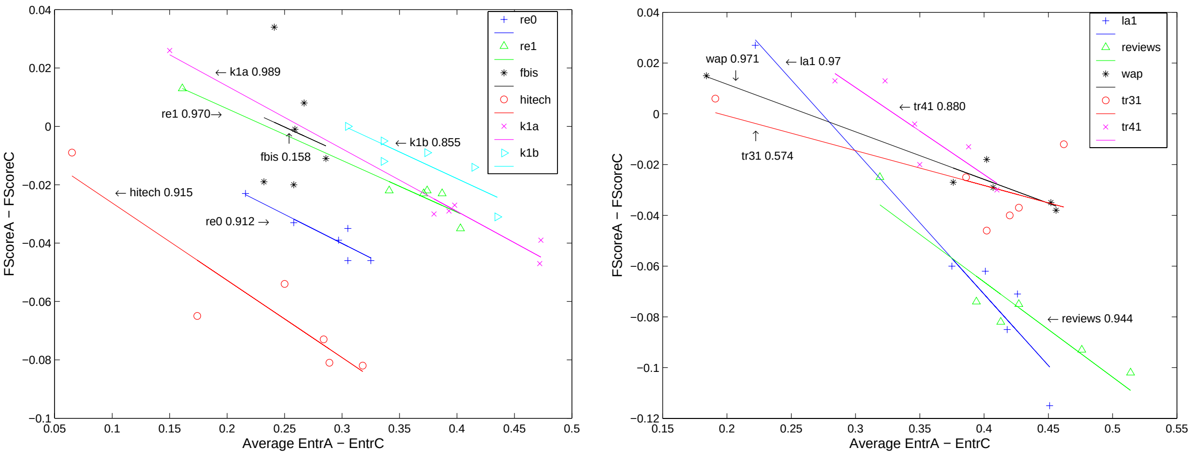
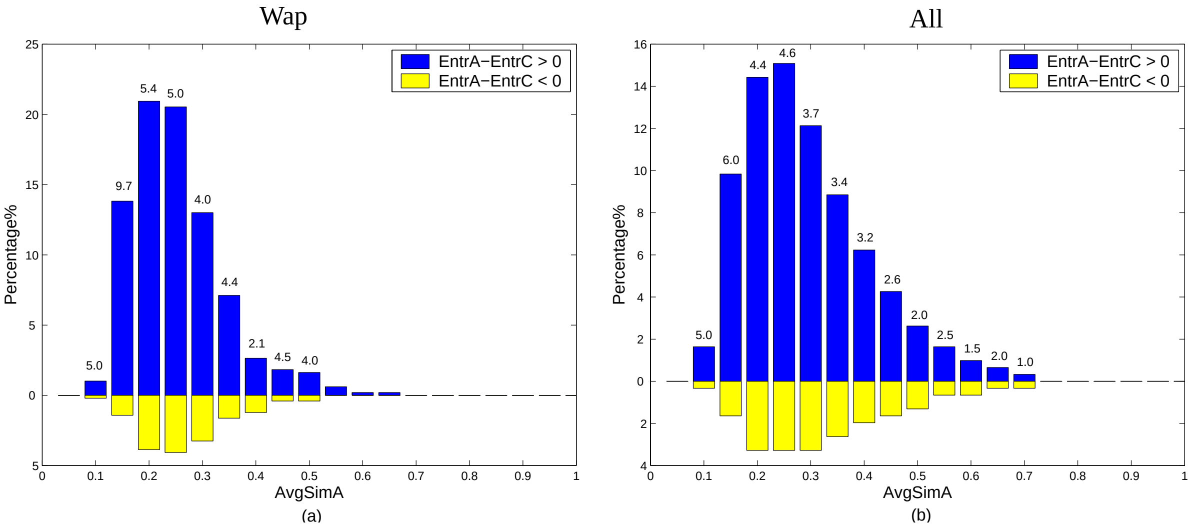


Figure 3: Correlation between the improvement of average five nearest neighbor entropy values and the improvement of FScore values for each dataset



(a)

(b)

Figure 4: The distribution of the 5-nn average pairwise similarities with any constraint (AvgSimA) for (a)dataset Wap and (b) all datasets.

Entropy differences vs. tightness To see why partitional constraints improve the quality of the neighborhood we further investigated how these improvements relate to the tightness of the original neighborhood of each document without any constraint. Specifically, for each document that has non-zero 5-nn entropy differences we calculated the average pairwise similarity of the 5-nn without any constraint (AvgSimA) and plotted the distribution of these similarities in Figure 4. The X-axis represents the unconstrained 5-nn average pairwise similarity (AvgSimA), whereas the Y-axis represents the percentage of the documents that have the corresponding AvgSimA values. The information for each bar is broken into two parts. The first (dark bars) shows the percentage of the documents with positive 5-nn entropy difference values (*i.e.*, whose neighborhoods were improved by enforcing partitional cluster constraints). The second (light bars) shows the percentage of the documents with negative 5-nn entropy difference values (*i.e.*, whose neighborhoods were not improved by enforcing partitional cluster constraints). The number above each bar represents the ratio of the number of the documents with positive 5-nn entropy difference values over the number of the documents with negative values.

As shown in Figure 4, most of the improvements happen when the average similarity of the unconstrained neighborhood is relatively low. Since the similarity between two documents is to a large extent a measure of the number of dimensions they share, documents with a low similarity to each other will have few dimensions in common. Now if we assume that each document class represents a set of documents that have a certain number of common dimensions (*i.e.*, subspace), then the fact that a document has 5-nn with low similarities suggests that either the dimensions that define the document class are few or the document is peripheral to the class. In either case, since the number of dimensions that are used to determine the class membership of these documents is small, documents from other classes can share the same number of dimensions just by random chance. Thus, besides the raw pairwise similarity between two documents, additional information is required in order to identify the right set of dimensions that each document should use when determining its neighborhood. The results shown in Figure 4 suggest that this information is provided by the partitional constraints. By taking a more global view at the clustering process, partitional schemes can identify the low dimensional subspaces that the various documents cluster in.

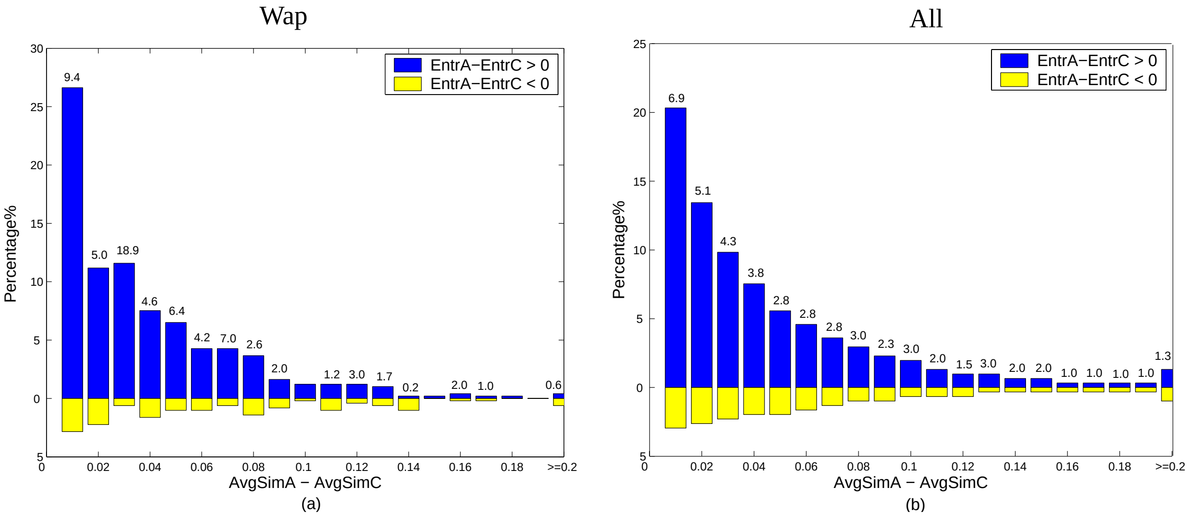


Figure 5: The distribution of the 5-nn average pairwise similarity differences o/w constraints (AvgSimA - AvgSimC) for (a)dataset Wap and (b) all datasets.

Entropy differences vs. tightness differences We also looked at how the entropy improvements of the various neighborhoods relate to the tightness difference. For each document that has non-zero 5-nn entropy differences we calculated AvgSimA - AvgSimC, where AvgSimA is the average pairwise similarity of the 5-nn without any constraint and AvgSimC is the average pairwise similarity of the 5-nn with partitional constraints. Figure 5 shows the distribution of these average similarity differences for Wap and over all the datasets. Note that as in Figure 4, the cases that lead to 5-nn entropy improvements were separated from those that lead to degradations. The number above each bar represents the ratio of the number of the documents with positive 5-nn entropy difference values over the number of the documents with negative values. Note that after enforcing partitional cluster constraints the average pairwise similarity always decreases or stays as the same, (*i.e.*, AvgSimA - AvgSimC is always equal to or greater than zero).

The results of Figure 5 reveal two interesting trends. First, for the majority of the documents the differences in the average 5-nn similarities between the constrained and the unconstrained neighborhoods is small (*i.e.*, the bars corresponding to low “AvgSimA-AvgSimC” entries account for a large fraction of the documents). This should not be

Table 7: Max FScore values achieved for each class by $\mathcal{I}_1$ and UPGMA for datasets tr31 and reviews

tr31

Class Name	Class Size	Max FScore ($\mathcal{I}_1$)	Max FScore (UPGMA)
301	352	0.95	0.95
306	**227**	**0.62**	**0.78**
307	111	0.81	0.69
304	**151**	**0.46**	**0.67**
302	63	0.73	0.71
305	21	0.86	0.92
310	2	0.67	0.67

reviews

Class Name	Class Size	Max FScore ($\mathcal{I}_1$)	Max FScore (UPGMA)
food	**999**	**0.61**	**0.75**
movie	1133	0.73	0.78
music	**1388**	**0.60**	**0.77**
radio	137	0.65	0.66
rest	412	0.61	0.65

surprising since it is a direct consequence of the fact that the constraining was obtained by clustering the documents in the first place (*i.e.*, grouping similar documents together). The second trend is that when the average 5-nn differences are small the constraining scheme more often than not leads to 5-nn neighborhoods that have better entropy. This can be easily observed by comparing the ratios shown at the top of each bar that are high for low differences and decrease as the average similarity difference increases. These results verify our earlier observations that when the neighborhood of each document contains equally similar documents that belong both to the same and different classes, then the guidance provided by the constraint clusters helps the documents to select the *right* neighboring documents leading to 5-nn neighborhoods with better entropy and subsequently improves the overall clustering solution.

7.2 Analysis of $\mathcal{I}_1$ and UPGMA

One surprising observation from the experimental results presented in Section 6.3 is that $\mathcal{I}_1$ and UPGMA behave very differently. Recall from Section 4.1 that the UPGMA method selects to merge the pair of clusters with the highest average pairwise similarity. Hence, to some extent, via the agglomeration process it tries to maximize the average pairwise similarity between the documents of the discovered clusters. On the other hand, the $\mathcal{I}_1$ method tries to find a clustering solution that maximizes the sum of the average pairwise similarity of the documents in each cluster, weighted by the size of the different clusters. Thus, $\mathcal{I}_1$ can be considered as the criterion function that UPGMA tries to optimize. However, our experimental results showed that $\mathcal{I}_1$ performed significantly worse than UPGMA.

To better understand how $\mathcal{I}_1$ and UPGMA perform differently, we looked at the maximum FScore values achieved for each individual class of each dataset. As an example Table 7 shows the maximum FScore values achieved for each class for two datasets (reviews and tr31) using the $\mathcal{I}_1$ and UPGMA agglomerative schemes. The columns labeled "Max FScore ($\mathcal{I}_1$)" and "Max FScore (UPGMA)" show the maximum FScore values achieved for each individual class by $\mathcal{I}_1$ and UPGMA, respectively. From these results we can see that even though both $\mathcal{I}_1$ and UPGMA do a comparable job in clustering most of the classes (*i.e.*, similar FScore values), for some of the large classes $\mathcal{I}_1$ performs worse than UPGMA (shown using a bold-faced font in Table 7). Note that these findings are not only true for these two datasets but also true for the rest of the datasets as well.

When looking at the hierarchical trees carefully, we found that for the classes that $\mathcal{I}_1$ performed significantly worse than UPGMA, $\mathcal{I}_1$ prefers to first merge in a loose subcluster of a different class, before it merges a tight subcluster of the same class. This happens even if the subcluster of the same class has higher cross similarity than the subcluster of the different class. This observation can be explained by the fact that $\mathcal{I}_1$ tends to merge loose clusters first, which is shown in the rest of this section.

From their definitions, the difference between $\mathcal{I}_1$ and UPGMA is that $\mathcal{I}_1$ takes into account the cross similarities as well as internal similarities of the clusters to be merged together. Let S_i and S_j be two of the candidate clusters of size n_i and n_j, respectively, also let μ_i and μ_j be the average pairwise similarity between the documents in S_i and S_j, respectively (*i.e.*, $\mu_i = {C_i}^t C_i$ and $\mu_j = {C_j}^t C_j$), and let ξ_{ij} be the average cross similarity between the documents in

S_i and the documents in S_j (*i.e.*, $\xi_{ij} = \frac{{D_i}^t D_j}{n_i n_j}$). UPGMA's merging decisions are based only on ξ_{ij}. On the other hand, $\mathcal{I}_1$ will merge the pair of clusters that optimizes the overall objective functions. The change of the overall value of the criterion function after merging two clusters S_i and S_j to obtain cluster S_r is given by,

$$\begin{aligned}
\Delta\mathcal{I}_1 &= \frac{\|D_r\|^2}{n_r} - \frac{\|D_i\|^2}{n_i} - \frac{\|D_j\|^2}{n_j} = n_r\mu_r - n_i\mu_i - n_j\mu_j \\
&= (n_i + n_j)\frac{n_i^2\mu_i + n_j^2\mu_j + 2n_i n_j \xi_{ij}}{(n_i + n_j)^2} - n_i\mu_i - n_j\mu_j = \frac{n_i n_j}{n_i + n_j}(2\xi_{ij} - \mu_i - \mu_j). \qquad (10)
\end{aligned}$$

From Equation 10, we can see that smaller μ_i and μ_j values will result in greater $\Delta\mathcal{I}_1$ values, which makes looser clusters easier to be merged first. For example, consider three clusters S_1, S_2 and S_3. S_2 is tight (*i.e.*, μ_2 is high) and of the same class as S_1, whereas S_3 is loose (*i.e.*, μ_3 is low) and of a different class. Suppose S_2 and S_3 have similar size, which means the value of $\Delta\mathcal{I}_1$ will be determined mainly by $(2\xi_{ij} - \mu_i - \mu_j)$, then it is possible that $(2\xi_{13} - \mu_1 - \mu_3)$ is greater than $(2\xi_{12} - \mu_1 - \mu_2)$ because μ_3 is less than μ_2, even if S_2 is closer to S_1 than S_3 (*i.e.*, $\xi_{12} > \xi_{13}$). As a result, if two classes are close and of different tightness, $\mathcal{I}_1$ may merge subclusters from each class together at early stages and fail to form proper nodes in the resulting hierarchical tree corresponding to those two classes.

8 Concluding Remarks

In this paper we experimentally evaluated nine agglomerative algorithms and six partitional algorithms to obtain hierarchical clustering solutions for document datasets. We also introduced a new class of agglomerative algorithms by constraining the agglomeration process using clusters obtained by partitional algorithms. Our experimental results showed that partitional methods produce better hierarchical solutions than agglomerative methods and that the constrained agglomerative methods improve the clustering solutions obtained by agglomerative or partitional methods alone. We analyzed that in most cases enforcing partitional cluster constraints improves the quality of the neighborhood of each document, especially when the document has low similarities to others or has many documents with similar similarities. These improvements of neighborhoods correlate well with the improvements of overall clustering solutions, which suggests that constrained agglomerative schemes benefit from starting with purer neighborhoods and hence lead to clustering solutions with better quality.

References

[1] Charu C. Aggarwal, Stephen C. Gates, and Philip S. Yu. On the merits of building categorization systems by supervised clustering. In *Proc. of the Fifth ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, pages 352–356, 1999.

[2] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *Proc. of the Sixth ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, pages 407–416, 2000.

[3] D. Boley, M. Gini, R. Gross, E.H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the world wide web using WebACE. *AI Review)*, 11:365–391, 1999.

[4] D. Boley, M. Gini, R. Gross, E.H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore.

Partitioning-based clustering for web document categorization. *Decision Support Systems (accepted for publication)*, 1999.

[5] Daniel Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4), 1998.

[6] P. Cheeseman and J. Stutz. Baysian classification (autoclass): Theory and results. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI/MIT Press, 1996.

[7] Chung-Kuan Cheng and Yen-Chuen A. Wei. An improved two-way partitioning algorithm with stable performance. *IEEE Transactions on Computer Aided Design*, 10(12):1502–1511, December 1991.

[8] D.R. Cutting, J.O. Pedersen, D.R. Karger, and J.W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the ACM SIGIR*, pages pages 318–329, Copenhagen, 1992.

[9] Jay Devore and Roxy Peck. *Statistics: the exploration and analysis of data*. Duxbury Press, Belmont, CA, 1997.

[10] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Knowledge Discovery and Data Mining*, pages 269–274, 2001.

[11] Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1/2):143–175, 2001.

[12] I.S. Dhillon and D.S. Modha. Concept decomposition for large sparse text data using clustering. Technical Report Research Report RJ 10147, IBM Almadan Research Center, 1999.

[13] Chris Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst Simon. Spectral min-max cut for graph partitioning and data clustering. Technical Report TR-2001-XX, Lawrence Berkeley National Laboratory, University of California, Berkeley, CA, 2001.

[14] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, 2001.

[15] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: An efficient clustering algorithm for large databases. In *Proc. of 1998 ACM-SIGMOD Int. Conf. on Management of Data*, 1998.

[16] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: a robust clustering algorithm for categorical attributes. In *Proc. of the 15th Int'l Conf. on Data Eng.*, 1999.

[17] Lars Hagen and Andrew Kahng. Fast spectral methods for ratio cut partitioning and clustering. In *Proceedings of IEEE International Conference on Computer Aided Design*, pages 10–13, 1991.

[18] E.H. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. WebACE: A web agent for document categorization and exploartion. In *Proc. of the 2nd International Conference on Autonomous Agents*, May 1998.

[19] E.H. Han, G. Karypis, V. Kumar, and B. Mobasher. Hypergraph based clustering in high-dimensional data sets: A summary of results. *Bulletin of the Technical Committee on Data Engineering*, 21(1), 1998.

[20] A.K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[21] G. Karypis, E.H. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.

[22] George Karypis. CLUTO a clustering toolkit. Technical Report 02-017, Dept. of Computer Science, University of Minnesota, 2002. Available at http://www.cs.umn.edu˜cluto.

[23] B. King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 69:86–101, 1967.

[24] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proc. of the Fifth ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, pages 16–22, 1999.

[25] A. Leouski and W. Croft. An evaluation of techniques for clustering search results, 1996.

[26] D. D. Lewis. Reuters-21578 text categorization test collection distribution 1.0. *http://www.research.att.com/∼lewis*, 1999.

[27] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. 5th Symp. Math. Statist, Prob.*, pages 281–297, 1967.

[28] J. Moore, E. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, and B. Mobasher. Web page categorization and feature selection using association rule and principal component clustering. In *7th Workshop on Information Technologies and Systems*, Dec. 1997.

[29] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *Proc. of the 20th VLDB Conference*, pages 144–155, Santiago, Chile, 1994.

[30] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[31] J. Puzicha, T. Hofmann, and J. Buhmann. A theory of proximity based clustering: Structure detection by optimization. *PATREC: Pattern Recognition, Pergamon Press*, 33(4):617–634, 2000.

[32] Jan Puzicha, Thomas Hofmann, and Joachim M. Buhmann. A theory of proximity based clustering: Structure detection by optimization. *PATREC: Pattern Recognition, Pergamon Press*, 33:617–634, 2000.

[33] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[34] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[35] P. H. Sneath and R. R. Sokal. *Numerical Taxonomy*. Freeman, London, UK, 1973.

[36] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[37] A. Strehl and J. Ghosh. Scalable approach to balanced, high-dimensional clustering of market-baskets. In *Proceedings of HiPC*, 2000.

[38] TREC. Text REtrieval conference. *http://trec.nist.gov*, 1999.

[39] C. J. Van Rijsbergen *Information Retrieval, 2nd edition*. butterworths, London, 1979.

[40] Peter Willett. Recent trends in hierarchic document clustering: a critical review. *Information Processing and Management*, 24(5):577–597, 1988.

[41] Yahoo! Yahoo! http://www.yahoo.com.

[42] K. Zahn. Graph-tehoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, (C-20):68–86, 1971.

[43] H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Bipartite graph partitioning and data clustering. In *CIKM*, 2001.

[44] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proc. of Int'l. Conf. on Information and Knowledge Management*, pages 515–524, 2002.

[45] Y. Zhao and G. Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. In *Journal of Machine Learning*, accepted for publication, 2003.