Evaluation of Hierarchical Clustering Algorithms for Document Datasets*

Ying Zhao and George Karypis

Department of Computer Science, University of Minnesota, Minneapolis, MN 55455
Techincal Report #02-022

{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, hierarchical clustering solutions provide a view of the data at different levels of granularity, making them ideal for people to visualize and interactively explore large document collections.

The focus of this paper is to evaluate different hierarchical clustering algorithms and toward this goal we compared various partitional and agglomerative approaches. Our experimental evaluation showed that partitional algorithms always lead to better clustering solutions than agglomerative algorithms, which suggests that partitional clustering algorithms are well-suited for clustering large document datasets due to not only their relatively low computational requirements, but also comparable or even better clustering performance.

We also present a new class of clustering algorithms called *constrained agglomerative algorithms* that combine the features of both partitional and agglomerative algorithms. Our experimental results showed that they consistently lead to better hierarchical solutions than agglomerative or partitional algorithms alone.

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 are indeed a natural constrain on the underlying application domain (*e.g.*, biological taxonomy, phylogenetic trees) [9].

Hierarchical clustering solutions have been primarily obtained using agglomerative algorithms [27, 19, 10, 11, 18], in which objects are initially assigned to its own cluster and then pairs of clusters are repeatedly merged until the whole tree is formed. However, partitional algorithms [22, 16, 24, 5, 33, 13, 29, 4, 8] can also be used to obtain hierarchical clustering solutions via a sequence of repeated bisections. In recent years, various researchers have recognized that partitional clustering algorithms are well-suited for clustering large document datasets due to their relatively low computational requirements [6, 20, 1, 28]. However, 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 was as a limited number of studies in which agglomerative approaches outperformed partitional K-means based approaches. For example, Larsen [20] observed that group average greedy agglomerative clustering outperformed various partitional clustering algorithms in document datasets from TREC and Reuters.

In light of recent advances in partitional clustering [6, 20, 7, 4, 8], we revisited the question of whether or not agglomerative approaches generate superior hierarchical trees than partitional approaches. The focus of this paper is

*This work was supported by NSF CCR-9972519, EIA-9986042, ACI-9982274, ACI-0133464, by Army Research Office contract DA/DAAG55-98-1-0441, by the DOE ASCI program, and by Army High Performance Computing Research Center contract number DAAH04-95-C-0008. Related papers are available via WWW at URL: **http://www.cs.umn.edu/~karypis**

to compare various agglomerative and partitional approaches for the task of obtaining hierarchical clustering solution. The partitional methods that we compared use different clustering criterion functions to derive the solutions and the agglomerative methods use different schemes for selecting the pair of clusters to merge next. For partitional clustering algorithms, we used six recently studied criterion functions [34] that have been shown to produce high-quality partitional clustering solutions. For agglomerative clustering algorithms, we evaluated three traditional merging criteria (*i.e.*, single-link, complete-link, and group average (UPGMA)) and a new set of merging criteria derived from the six partitional criterion functions. Overall, we compared six partitional methods and nine agglomerative methods.

In addition to the traditional partitional and agglomerative algorithms, we developed a new class of agglomerative algorithms, in which we introduced intermediate clusters obtained by partitional clustering algorithms to constrain the space over which agglomeration decisions are made. We refer to them as *constrained agglomerative algorithms*. These algorithms generate hierarchical trees in two steps. First, for each of the intermediate partitional clusters, an agglomerative algorithm builds a hierarchical subtree. Second, the subtrees are combined into a single tree by building an upper tree using these subtrees as leaves.

We experimentally evaluated the performance of these methods to obtain hierarchical clustering solutions using twelve different datasets derived from various sources. Our experiments showed that partitional algorithms always generate better hierarchical clustering solutions than agglomerative algorithms and that the constrained agglomerative methods consistently lead to better solutions than agglomerative methods alone and in most cases they outperform partitional methods as well. We believe that the observed poor performance of agglomerative algorithms is because of the errors they make during early agglomeration. The superiority of partitional algorithm also suggests that partitional clustering algorithms are well-suited for obtaining hierarchical clustering solutions of large document datasets due to not only their relatively low computational requirements, but also comparable or better performance.

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 various agglomerative algorithms and the constrained agglomerative algorithms. Section 5 provides the detailed experimental evaluation of the various hierarchical clustering methods as well as the experimental results of the constrained agglomerative algorithms. Section 6 discusses some important observations from the experimental results. Finally, Section 7 provides some concluding remarks.

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, $S_1, S_2, \ldots, S_k$ to denote each one of the k clusters, and $n_1, n_2, \ldots, n_k$ to denote the sizes of the corresponding clusters.

The various clustering algorithms that are described in this paper use the vector-space model [26] 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

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

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).

Vector Properties By using the cosine function as the measure of similarity between documents we can take advantage of a number of properties involving the composite and centroid vectors of a set of documents. In particular, if S_i and S_j are two sets of unit-length documents containing n_i and n_j documents respectively, and D_i, D_j and C_i, C_j are their corresponding composite and centroid vectors then the following is true:

1. The sum of the pair-wise similarities between the documents in S_i and the document in S_j is equal to ${D_i}^t D_j$. That is,
$$\sum_{d_q \in D_i, d_r \in D_j} \cos(d_q, d_r) = \sum_{d_q \in D_i, d_r \in D_j} {d_q}^t d_r = {D_i}^t D_j. \quad (1)$$

2. The sum of the pair-wise similarities between the documents in S_i is equal to $\|D_i\|^2$. That is,
$$\sum_{d_q, d_r \in D_i} \cos(d_q, d_r) = \sum_{d_q, d_r \in D_i} {d_q}^t d_r = {D_i}^t D_i = \|D_i\|^2. \quad (2)$$

   Note that this equation includes the pairwise similarities involving the same pairs of vectors.

3 Hierarchical Partitional Clustering Algorithm

Partitional clustering algorithms can be used to compute a hierarchical clustering solution using a repeated cluster bisectioning approach [28, 34]. 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). In the rest of this section we describe the various aspects of the partitional clustering algorithm that we used in our study.

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.

The clustering criterion functions that we used in our study can be classified into four groups: internal, external, hybrid and graph-based. The ***internal*** criterion functions focus on producing a clustering solution that optimizes a function defined only over the documents of each cluster and does not take into account the documents assigned to different clusters. The ***external*** criterion functions derive the clustering solution by focusing on optimizing a function that is based on how the various clusters are different from each other. The ***graph based*** criterion functions model the documents as a graph and use clustering quality measures defined in the graph model. The ***hybrid*** criterion functions simultaneously optimize multiple individual criterion functions.

Internal Criterion Functions The first internal criterion function maximizes the sum of the average pairwise similarities between the documents assigned to each cluster, weighted according to the size of each cluster. Specifically, if we use the cosine function to measure the similarity between documents, then we want the clustering solution to optimize the following criterion function:
$$\text{maximize } \mathcal{I}_1 = \sum_{r=1}^{k} n_r \left(\frac{1}{n_r^2} \sum_{d_i, d_j \in S_r} \cos(d_i, d_j) \right) = \sum_{r=1}^{k} \frac{\|D_r\|^2}{n_r}. \quad (3)$$

The second criterion function is used by the popular vector-space variant of the K-means algorithm [6, 20, 7, 28, 17]. In this algorithm each cluster is represented by its centroid vector and the goal is to find the clustering solution that maximizes the similarity between each document and the centroid of the cluster that is assigned to. Specifically, if we use the cosine function to measure the similarity between a document and a centroid, then the criterion function

becomes the following:

$$\text{maximize } \mathcal{I}_2 = \sum_{r=1}^{k} \sum_{d_i \in S_r} \cos(d_i, C_r). = \sum_{r=1}^{k} \sum_{d_i \in S_r} \frac{{d_i}^t C_r}{\|C_r\|} = \sum_{r=1}^{k} \frac{{D_r}^t C_r}{\|C_r\|} = \sum_{r=1}^{k} \frac{{D_r}^t D_r}{\|D_r\|} = \sum_{r=1}^{k} \|D_r\|. \tag{4}$$

Comparing the $\mathcal{I}_2$ criterion function with $\mathcal{I}_1$ we can see that the essential difference between these criterion functions is that $\mathcal{I}_2$ scales the within-cluster similarity by the $\|D_r\|$ term as opposed to n_r term used by $\mathcal{I}_1$. The term $\|D_r\|$ is nothing more than the square-root of the pairwise similarity between all the document in S_r, and will tend to emphasize the importance of clusters (beyond the $\|D_r\|^2$ term) whose documents have smaller pairwise similarities compared to clusters with higher pair-wise similarities. Also note that if the similarity between a document and the centroid vector of its cluster is defined as just the dot-product of these vectors, then we will get back the $\mathcal{I}_1$ criterion function.

External Criterion Functions It is quite hard to define external criterion functions that lead to meaningful clustering solutions. For example, it may appear that an intuitive external function may be derived by requiring that the centroid vectors of the different clusters are as mutually orthogonal as possible, *i.e.*, they contain documents that share very few terms across the different clusters. However, for many problems this criterion function has trivial solutions that can be achieved by assigning to the first $k-1$ clusters a single document that shares very few terms with the rest, and then assigning the rest of the documents to the kth cluster. For this reason, the external function that we will discuss tries to separate the documents of each cluster from the entire collection, as opposed trying to separate the documents among the different clusters. This external criterion function was motivated by multiple discriminant analysis and is similar to minimizing the trace of the between-cluster scatter matrix [9, 30].

In particular, our external criterion function is defined as

$$\text{minimize } \sum_{r=1}^{k} n_r \cos(C_r, C), \tag{5}$$

where C is the centroid vector of the entire collection. From this equation we can see that we try to minimize the cosine between the centroid vector of each cluster to the centroid vector of the entire collection. By minimizing the cosine we essentially try to increase the angle between them as much as possible. Also note that the contribution of each cluster is weighted based on the cluster size, so that larger clusters will weight heavier in the overall clustering solution. Equation 5 can be re-written as

$$\sum_{r=1}^{k} n_r \cos(C_r, C) = \sum_{r=1}^{k} n_r \frac{{C_r}^t C}{\|C_r\|\|C\|} = \sum_{r=1}^{k} n_r \frac{{D_r}^t D}{\|D_r\|\|D\|} = \frac{1}{\|D\|} \left(\sum_{r=1}^{k} n_r \frac{{D_r}^t D}{\|D_r\|} \right),$$

where D is the composite vector of the entire document collection. Note that since $1/\|D\|$ is constant irrespective of the clustering solution the criterion function can be re-stated as:

$$\text{minimize } \mathcal{E}_1 = \sum_{r=1}^{k} n_r \frac{{D_r}^t D}{\|D_r\|}. \tag{6}$$

As we can see from Equation 6, even-though our initial motivation was to define an external criterion function, because we used the cosine function to measure the separation between the cluster and the entire collection, the criterion function does take into account the within-cluster similarity of the documents (due to the $\|D_r\|$ term). Thus, $\mathcal{E}_1$ is actually a hybrid criterion function that combines both external and internal characteristics of the clusters.

Hybrid Criterion Functions In our study, we will focus on two hybrid criterion function that are obtained by combining criterion $\mathcal{I}_1$ with $\mathcal{E}_1$, and $\mathcal{I}_2$ with $\mathcal{E}_1$, respectively. Formally, the first criterion function is

$$\text{maximize } \mathcal{H}_1 = \frac{\mathcal{I}_1}{\mathcal{E}_1} = \frac{\sum_{r=1}^{k} \|D_r\|^2/n_r}{\sum_{r=1}^{k} n_r {D_r}^t D/\|D_r\|}, \tag{7}$$

and the second is

$$\text{maximize } \mathcal{H}_2 = \frac{\mathcal{I}_2}{\mathcal{E}_1} = \frac{\sum_{r=1}^{k} \|D_r\|}{\sum_{r=1}^{k} n_r {D_r}^t D / \|D_r\|}. \tag{8}$$

Note that since $\mathcal{E}_1$ is minimized, both $\mathcal{H}_1$ and $\mathcal{H}_2$ need to be maximized as they are inversely related to $\mathcal{E}_1$.

Graph Based Criterion Functions An alternate way of viewing the relations between the documents is to use similarity graphs. Given a collection of n documents S, the similarity graph G_s is obtained by modeling each document as a vertex, and having an edge between each pair of vertices whose weight is equal to the similarity between the corresponding documents. Viewing the documents in this fashion, a number of internal, external, or combined criterion functions can be defined that measure the overall clustering quality. In our study we will investigate one such criterion function called MinMaxCut, that was proposed recently [8]. MinMaxCut falls under the category of criterion functions that combine both the internal and external views of the clustering process and is defined as [8]

$$\text{minimize } \sum_{r=1}^{k} \frac{\text{cut}(S_r, S - S_r)}{\sum_{d_i, d_j \in S_r} \text{sim}(d_i, d_j)},$$

where $\text{cut}(S_r, S - S_r)$ is the edge-cut between the vertices in S_r to the rest of the vertices in the graph $S - S_r$. The edge-cut between two sets of vertices A and B is defined to be the sum of the edges connecting vertices in A to vertices in B. The motivation behind this criterion function is that the clustering process can be viewed as that of partitioning the documents into groups by minimizing the edge-cut of each partition. However, for reasons similar to those discussed in Section 3.1, such an external criterion may have trivial solutions, and for this reason each edge-cut is scaled by the sum of the internal edges. As shown in [8], this scaling leads to better balanced clustering solutions.

If we use the cosine function to measure the similarity between the documents, and Equations 1 and 2, then the above criterion function can be re-written as

$$\sum_{r=1}^{k} \frac{\sum_{d_i \in S_r, d_j \in S - S_r} \cos(d_i, d_j)}{\sum_{d_i, d_j \in S_r} \cos(d_i, d_j)} = \sum_{r=1}^{k} \frac{{D_r}^t (D - D_r)}{\|D_r\|^2} = \left(\sum_{r=1}^{k} \frac{{D_r}^t D}{\|D_r\|^2} \right) - k,$$

and since k is constant, the criterion function can be simplified to

$$\text{minimize } \mathcal{G}_1 = \sum_{r=1}^{k} \frac{{D_r}^t D}{\|D_r\|^2}. \tag{9}$$

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 [28, 34]. The details of this algorithm are provided in the remaining of this section.

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 one of the other $k - 1$ clusters. 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 cluster 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 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* [9]. Since each move directly optimizes the particular criterion function, this refinement strategy always converges to a local minima. 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 minima, and the local minima 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

We experimented with two different methods for selecting which cluster to bisect next. 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 [28, 34]. 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 5.

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-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 for the greedy refinement algorithm is small (less than 20), and are to a large extend 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 pairs 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[27, 19, 16, 10, 11, 18]. 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 [27] 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

$$\text{sim}_{\text{single-link}}(S_i, S_j) = \max_{d_i \in S_i,\ d_j \in S_j} \{\cos(d_i, d_j)\}. \tag{10}$$

In contrast, the complete-link scheme [19] uses the minimum similarity between a pair of documents to measure the same similarity. That is,

$$\text{sim}_{\text{complete-link}}(S_i, S_j) = \min_{d_i \in S_i,\ d_j \in S_j} \{\cos(d_i, d_j)\}. \tag{11}$$

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 [16] (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,

$$\mathrm{sim}_{\mathrm{UPGMA}}(S_i, S_j) = \frac{1}{n_i n_j} \sum_{d_i \in S_i,\ d_j \in S_j} \cos(d_i, d_j) = \frac{{D_i}^t D_j}{n_i n_j}. \tag{12}$$

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 [9], 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 the 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 that 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 tends to be changed for all pairs of clusters. As a result, they cannot be pre-computed and inserted into a priority queue.

4.3 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 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, their 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 way of improving agglomerative clustering algorithms by eliminating this type of errors, is to use a partitional clustering algorithm to constrain the space over which agglomeration decisions are made, so that each document is only allowed 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. Then, each of these 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 $O(k((\frac{n}{k})^2 \log(\frac{n}{k})) + k^2 \log k)$, where k is the number of intermediate partitional 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 $O(n^{2/3} \log n)$.

5 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, which can be downloaded from *http://www.cs.umn.edu/~karypis/cluto*.

5.1 Document Collections

In our experiments, we used a total of twelve different datasets, whose general characteristics are summarized in Table 1. The smallest of these datasets contained 878 documents and the largest contained 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 [25]. Moreover, any term that occurs in fewer than two documents was eliminated.

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
la2	LA Times (TREC)	3075	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

Table 1: Summary of data sets used to evaluate the various clustering criterion functions.

The *fbis* dataset is from the Foreign Broadcast Information Service data of TREC-5 [31], 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 contained 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* and *la2* datasets were obtained from articles of the Los Angeles Times that was used in TREC-5 [31]. 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* are derived from TREC-5 [31], TREC-6 [31], and TREC-7 [31] 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 [21]. 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 [23, 12, 2, 3]. Each document corresponds to a web page listed in the subject hierarchy of Yahoo! [32]. 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*.

5.2 Experimental Methodology and Metrics

For each one of the different datasets we obtained hierarchical clustering solutions using the various partitional and agglomerative clustering algorithms described in Sections 3 and 4. The quality of a clustering solution was determined by analyzing the entire hierarchical tree that is produced by a particular clustering algorithm. This is often done by using a measure that takes into account the overall set of clusters that are represented in the hierarchical tree. One such measure is the ***FScore measure***, introduced by [20]. Given a particular class C_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 C_r, then the FScore of this class and cluster is defined to be

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

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

$$F(C_r) = \max_{S_i \in T} F(C_r, S_i).$$

The FScore of the entire clustering solution is then defined to be the sum of the individual class FScore weighted according to the class size.

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

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 exactly same 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.

5.3 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 algorithms and partitional algorithms. For agglomerative algorithms, nine selection schemes or criterion functions have been tested including the six criterion functions discussed in Section 3.1, and the three traditional 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 bisection 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. Overall, we evaluated 15 hierarchical clustering methods.

The FScore results for the hierarchical trees for the various datasets and methods are shown in Table 2, where each row corresponds to one method and each column corresponds to one dataset. The results in this table are provided primarily for completeness and in order to evaluate the various methods we actually summarized these results in two ways, one is by looking at the average performance of each method over the entire set of datasets, and the other is by comparing each pair of methods to see which method outperforms the other for most of the datasets.

The first way of summarizing the results is to average the FScore for each method over the twelve different datasets. However, since the hierarchical tree quality for different datasets is quite different, we felt that such simple averaging may distort the overall results. For this reason, we used averages of relative FScores as follows. For each dataset, we divided the FScore obtained by a particular method by the largest FScore obtained for that particular dataset over the 15 methods. These ratios represent the degree to which a particular method performed worse than the best method for that particular series of experiments. Note that for different datasets, the method that achieved the best hierarchical tree as measured by FScore may be different. These ratios are less sensitive to the actual FScore values. We will refer to these ratios as *relative FScores*. Since, higher FScore values are better, all these relative FScore values are less than one. Now, for each method we averaged these relative FScores over the various datasets. A method that has an ***average relative FScore*** close to 1.0 will indicate that this method did the best for most of the datasets. On the other hand, if the average relative FScore is low, then this method performed poorly.

	fbis	hitech	k1a	k1b	la1	la2	re0	re1	reviews	tr31	tr41	wap
$\mathcal{E}_1$	0.643	0.485	0.544	0.816	0.634	0.633	0.590	0.655	0.724	0.795	0.772	0.577
$\mathcal{G}_1$	0.624	0.471	0.603	0.844	0.612	0.644	0.581	0.617	0.654	0.796	0.754	0.618
$\mathcal{H}_1$	0.629	0.491	0.597	0.872	0.642	0.613	0.603	0.662	0.651	0.818	0.728	0.629
$\mathcal{H}_2$	0.637	0.468	0.579	0.858	0.620	0.697	0.585	0.660	0.727	0.789	0.729	0.580
$\mathcal{I}_1$	0.592	0.480	0.583	0.836	0.580	0.610	0.561	0.607	0.642	0.756	0.694	0.588
$\mathcal{I}_2$	0.639	0.480	0.605	0.896	0.648	0.681	0.587	0.684	0.689	0.844	0.779	0.618
UPGMA	0.673	0.499	0.646	0.892	0.654	0.709	0.584	0.695	0.750	0.816	0.826	0.640
slink	0.481	0.393	0.375	0.655	0.369	0.365	0.465	0.445	0.452	0.532	0.674	0.435
clink	0.609	0.382	0.552	0.764	0.364	0.449	0.495	0.508	0.513	0.804	0.758	0.569
p$\mathcal{E}_1$	0.623	0.577	0.670	0.891	0.721	0.787	0.618	0.758	0.870	0.858	0.743	0.694
p$\mathcal{G}_1$	0.668	0.512	0.697	0.872	0.758	0.725	0.639	0.721	0.818	0.892	0.783	0.687
p$\mathcal{H}_1$	0.686	0.545	0.691	0.889	0.761	0.742	0.628	0.699	0.746	0.877	0.811	0.672
p$\mathcal{H}_2$	0.641	0.581	0.693	0.902	0.749	0.739	0.632	0.723	0.859	0.873	0.800	0.690
p$\mathcal{I}_1$	0.702	0.481	0.666	0.876	0.646	0.634	0.626	0.675	0.762	0.769	0.753	0.679
p$\mathcal{I}_2$	0.681	0.575	0.682	0.882	0.801	0.766	0.633	0.712	0.821	0.893	0.833	0.714

Table 2: The FScores for the different datasets for the hierarchical clustering solutions obtained via various hierarchical clustering methods.

The results of the relative FScores for various hierarchical clustering methods are shown in Table 3. Again, each row of the table corresponds to one method, and each column of the table corresponds to one dataset. The average relative FScore values are shown in the last column labeled "Average". The entries that are boldfaced correspond to the methods that performed the best, and the entries that are underlined correspond to the methods that performed the best among agglomerative methods or partitional methods.

	fbis	hitech	k1a	k1b	la1	la2	re0	re1	reviews	tr31	tr41	wap	Average
$\mathcal{E}_1$	0.916	0.835	0.780	0.905	0.791	0.804	0.923	0.864	0.832	0.890	0.927	0.808	0.856
$\mathcal{G}_1$	0.889	0.811	0.865	0.936	0.764	0.818	0.909	0.814	0.752	0.891	0.905	0.866	0.852
$\mathcal{H}_1$	0.896	0.845	0.857	0.967	0.801	0.779	0.944	0.873	0.748	0.916	0.874	0.881	0.865
$\mathcal{H}_2$	0.907	0.805	0.831	0.951	0.774	0.886	0.915	0.871	0.836	0.883	0.875	0.812	0.862
$\mathcal{I}_1$	0.843	0.826	0.836	0.927	0.724	0.775	0.878	0.801	0.738	0.847	0.833	0.824	0.821
$\mathcal{I}_2$	0.910	0.826	0.868	0.993	0.809	0.865	0.919	0.902	0.792	0.945	0.935	0.866	0.886
UPGMA	0.959	0.859	0.927	0.989	0.817	0.901	0.914	0.917	0.862	0.914	0.992	0.896	<u>0.912</u>
slink	0.685	0.676	0.538	0.726	0.461	0.464	0.728	0.587	0.519	0.596	0.809	0.609	0.617
clink	0.868	0.657	0.792	0.847	0.454	0.571	0.775	0.670	0.590	0.900	0.910	0.797	0.736
p$\mathcal{E}_1$	0.887	0.993	0.961	0.988	0.900	1.000	0.967	1.000	1.000	0.961	0.892	0.972	0.960
p$\mathcal{G}_1$	0.952	0.881	1.000	0.967	0.946	0.921	1.000	0.951	0.940	0.999	0.940	0.962	0.955
p$\mathcal{H}_1$	0.977	0.938	0.991	0.986	0.950	0.943	0.983	0.922	0.858	0.982	0.974	0.941	0.954
p$\mathcal{H}_2$	0.913	1.000	0.994	1.000	0.935	0.939	0.989	0.954	0.987	0.978	0.960	0.966	0.968
p$\mathcal{I}_1$	1.000	0.828	0.956	0.971	0.806	0.806	0.980	0.890	0.876	0.861	0.904	0.951	0.902
p$\mathcal{I}_2$	0.970	0.990	0.979	0.978	1.000	0.973	0.991	0.939	0.944	1.000	1.000	1.000	<u>**0.980**</u>

Table 3: The relative FScores averaged over the different datasets for the hierarchical clustering solutions obtained via various hierarchical clustering methods.

A number of observations can be made by analyzing the results in Table 3. 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 the average, the p$\mathcal{I}_2$ method outperforms the other partitional methods and agglomerative methods by 2%–8% and 7%–37%, respectively. Second, the UPGMA method performs the best among agglomerative methods followed by the $\mathcal{I}_2$ method. The two methods together achieved the best hierarchical clustering solutions among agglomerative methods for all the datasets except re0. On the average, the UPGMA and $\mathcal{I}_2$ methods outperform the other agglomerative methods by 5%–30% and 2%–27%, respectively. 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%. The p$\mathcal{I}_1$ method performs a little bit worse than the UPGMA method and better than the rest of the agglomerative methods. Fourth, single-link, complete-link and $\mathcal{I}_1$ performed poorly among agglomerative methods and p$\mathcal{I}_1$ performed the worst among partitional methods. Finally, on the average, $\mathcal{H}_1$ 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.

When the relative performance of different methods is close, the average relative FScores will be quite similar.

	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$	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	4	6	9	4	1	12	10	2	0	0	1	3	0
$\mathcal{G}_1$	5	0	4	5	11	0	0	12	10	2	0	0	0	3	0
$\mathcal{H}_1$	8	8	0	8	12	3	2	12	11	1	0	0	0	2	0
$\mathcal{H}_2$	6	7	4	0	9	2	1	12	10	1	0	0	0	2	0
$\mathcal{I}_1$	3	1	0	3	0	0	0	12	9	0	0	0	0	0	0
$\mathcal{I}_2$	8	11	9	10	11	0	3	12	12	3	1	1	0	6	1
UPGMA	11	12	10	11	12	9	0	12	12	3	3	3	2	7	1
slink	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
clink	2	2	1	2	3	0	0	10	0	1	0	0	0	2	0
p$\mathcal{E}_1$	10	10	11	11	12	9	9	12	11	0	6	6	4	9	5
p$\mathcal{G}_1$	12	12	11	12	12	11	9	12	12	6	0	6	5	10	3
p$\mathcal{H}_1$	12	12	12	12	12	11	9	12	12	6	6	0	5	9	3
p$\mathcal{H}_2$	11	12	12	12	12	12	10	12	12	8	7	7	0	11	5
p$\mathcal{I}_1$	9	9	10	10	12	6	5	12	10	3	2	3	1	0	1
p$\mathcal{I}_2$	12	12	12	12	12	11	11	12	12	7	9	9	7	11	0

Table 4: Dominance matrix for various hierarchical clustering methods.

Hence, to make the comparisons of these methods easier, our second way of summarizing the results is to create a dominance matrix for the various methods. As shown in Table 4, the dominance matrix is a 15 by 15 matrix, where each row or column corresponds to one method and the value in each entry is the number of datasets for which the method corresponding to the row outperforms the method corresponding to 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 twelve datasets, the $\mathcal{I}_2$ method outperforms the $\mathcal{E}_1$ method. The values that are close to twelve indicate that the row method outperforms the column method.

Similar 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 twelve except two entries in the row of p$\mathcal{I}_1$, which means 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 $\mathcal{I}_2$ and $\mathcal{H}_1$, and slink, clink and $\mathcal{I}_1$ are the worst set of agglomerative methods. 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$\mathcal{H}_2$, and p$\mathcal{I}_1$ performed worse than the other partitional methods for most of the datasets.

5.4 Constrained Agglomerative Trees

Our second set of experiments was focused on evaluating the constrained agglomerative clustering methods. These results were obtained by using the different criterion functions to find intermediate partitional clusters, and then using UPGMA as the agglomerative scheme to construct the final hierarchical solutions as described in Section 4.3. UPGMA was selected because it performed the best among the various agglomerative schemes.

The results of the constrained agglomerative clustering methods are shown in Table 5. Each dataset is shown in a different subtable. There are six experiments performed for each dataset and each of them corresponds to a row. The row labeled "UPGMA" contains the FScores for the hierarchical clustering solutions generated by the UPGMA method with one intermediate cluster, which are the same for all the criterion functions. The rows labeled "10", "20", "$n/40$" and "$n/20$" contain the FScores obtained by the constrained agglomerative methods using 10, 20, $n/40$ and $n/20$ partitional clusters to constrain the solution, where n is the total number of documents in each dataset. The row labeled "rb" contains the FScores for the hierarchical clustering solutions obtained by repeated bisection algorithms with various criterion functions. The entries that are boldfaced correspond to the method that performed the best for a particular criterion function, whereas the entries that are underlined correspond to the best hierarchical clustering solution obtained for each dataset.

A number of observations can be made by analyzing the results in Table 5. First, for all the datasets except tr41, the constrained agglomerative methods improved the hierarchical solutions obtained by agglomerative methods alone, no matter what partitional clustering algorithm is used to obtain intermediate clusters. The improvement can be achieved even with small number of intermediate clusters. Second, for many cases, the constrained agglomerative methods performed even better than the corresponding partitional methods. Finally, the partitional clustering methods that improved the agglomerative hierarchical results the most are the same partitional clustering methods that performed

fbis						
Method	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$
UPGMA	0.673	0.673	0.673	0.673	0.673	0.673
10	0.656	0.659	0.692	0.628	0.709	0.669
20	0.658	0.681	**0.707**	0.681	0.718	0.687
$n/40$	0.683	0.685	0.693	0.691	0.720	**0.704**
$n/20$	**0.686**	**0.699**	0.703	**0.708**	**0.721**	0.691
rb	0.623	0.668	0.686	0.641	0.702	0.681

hitech						
Method	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$
UPGMA	0.499	0.499	0.499	0.499	0.499	0.499
10	**0.595**	0.535	0.568	0.587	0.490	**0.583**
20	0.590	**0.556**	**0.573**	**0.609**	0.517	**0.583**
$n/40$	0.554	0.536	0.563	0.578	**0.539**	0.564
$n/20$	0.545	0.535	0.543	0.562	0.529	0.561
rb	0.577	0.512	0.545	0.581	0.481	0.575

k1a						
Method	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$
UPGMA	0.646	0.646	0.646	0.646	0.646	0.646
10	0.680	0.661	0.653	0.691	0.618	0.677
20	**0.696**	0.673	0.688	0.708	0.633	0.699
$n/40$	0.694	0.680	**0.721**	**0.724**	0.649	0.700
$n/20$	0.692	**0.683**	0.710	0.714	**0.669**	**0.722**
rb	0.670	0.697	0.691	0.693	0.666	0.682

k1b						
Method	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$
UPGMA	0.892	0.892	0.892	0.892	0.892	0.892
10	0.911	**0.898**	0.910	0.926	0.904	**0.908**
20	0.908	0.891	0.908	0.925	**0.907**	0.903
$n/40$	**0.918**	0.896	**0.915**	**0.936**	0.893	0.903
$n/20$	0.897	0.896	0.907	0.928	0.876	0.899
rb	0.891	0.872	0.889	0.902	0.876	0.882

la1						
Method	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$
UPGMA	0.654	0.654	0.654	0.654	0.654	0.654
10	0.736	0.704	0.707	0.743	0.629	**0.806**
20	**0.747**	0.712	0.709	0.728	0.665	0.798
$n/40$	0.737	**0.763**	0.696	0.729	**0.699**	0.783
$n/20$	0.722	0.730	0.678	0.737	0.695	0.768
rb	0.721	0.758	**0.761**	**0.749**	0.646	0.801

la2						
Method	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$
UPGMA	0.709	0.709	0.709	0.709	0.709	0.709
10	0.712	0.719	0.714	0.740	0.672	0.759
20	**0.794**	0.735	0.731	0.783	0.741	0.765
$n/40$	0.792	**0.772**	0.725	0.789	0.738	0.787
$n/20$	0.763	0.737	0.738	**0.792**	**0.764**	**0.794**
rb	0.787	0.725	**0.742**	0.739	0.634	0.766

re0						
Method	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$
UPGMA	0.584	0.584	0.584	0.584	0.584	0.584
10	0.622	0.632	0.624	0.650	0.611	0.629
20	0.631	0.639	0.616	0.646	0.614	0.632
$n/40$	**0.639**	0.641	0.626	**0.657**	0.586	0.635
$n/20$	0.633	**0.642**	0.626	0.645	**0.634**	**0.645**
rb	0.618	0.639	**0.628**	0.632	0.626	0.633

re1						
Method	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$
UPGMA	0.695	0.695	0.695	0.695	0.695	0.695
10	0.713	**0.741**	0.730	0.723	**0.714**	0.731
20	0.719	0.735	0.728	**0.728**	0.708	0.724
$n/40$	0.713	0.723	0.753	0.715	**0.714**	0.719
$n/20$	0.735	0.714	**0.754**	0.708	0.702	**0.739**
rb	**0.758**	0.721	0.699	0.723	0.675	0.712

tr31						
Method	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$
UPGMA	0.816	0.816	0.816	0.816	0.816	0.816
10	0.862	0.893	**0.897**	0.878	0.828	**0.897**
20	0.882	**0.894**	0.895	0.866	0.839	0.896
$n/40$	**0.896**	**0.894**	0.891	**0.881**	0.847	0.853
$n/20$	0.839	**0.894**	0.894	0.843	**0.874**	0.853
rb	0.858	0.892	0.877	0.873	0.769	0.893

tr41						
Method	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$
UPGMA	**0.826**	**0.826**	**0.826**	**0.826**	0.826	0.826
10	0.770	0.797	0.809	0.789	**0.863**	0.807
20	0.821	0.811	0.799	0.815	0.858	0.848
$n/40$	0.805	0.824	0.813	0.821	0.842	**0.849**
$n/20$	0.818	0.782	0.793	0.802	0.802	0.839
rb	0.743	0.783	0.811	0.800	0.753	0.833

reviews						
Method	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$
UPGMA	0.750	0.750	0.750	0.750	0.750	0.750
10	0.870	0.854	0.844	0.866	0.817	0.851
20	0.871	0.847	0.846	**0.867**	0.815	0.850
$n/40$	**0.873**	**0.855**	0.847	0.859	0.824	**0.852**
$n/20$	0.862	0.852	**0.849**	0.858	**0.828**	**0.852**
rb	0.870	0.818	0.746	0.859	0.762	0.821

wap						
Method	$\mathcal{E}_1$	$\mathcal{G}_1$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{I}_1$	$\mathcal{I}_2$
UPGMA	0.640	0.640	0.640	0.640	0.640	0.640
10	0.696	0.685	0.684	0.679	0.639	0.658
20	0.677	**0.705**	0.694	0.672	0.640	0.672
$n/40$	0.693	0.700	**0.700**	0.686	0.656	0.696
$n/20$	**0.708**	0.703	0.689	**0.709**	0.672	0.700
rb	0.694	0.687	0.672	0.690	**0.679**	**0.714**

Table 5: Comparison of UPGMA, constrained agglomerative methods with 10, 20, $n/40$ and $n/20$ intermediate partitional clusters, and repeated bisection methods with various criterion functions.

the best in terms of generating the whole hierarchical trees.

6 Discussion

The most important observation from the experimental results is that partitional methods performed better than agglomerative methods. As discussed in Section 4.3, one of the limitations of agglomerative methods is that errors may be introduced during the initial merging decisions, especially for the cases in which there are a large number of equally good merging alternatives for each cluster. Without a high-level view of the overall clustering solution, it is hard for agglomerative methods to make the right decision in such cases. Since the errors will be carried through and may be multiplied as the agglomeration progresses, the resulting hierarchical trees suffer from those early stage errors. This observation is also supported from the experimental results with the constrained agglomerative algorithms. We can see in this case that once we constrain the space over which agglomeration decisions are made, even with small number of intermediate clusters, some early stage errors can be eliminated. As a result, the constrained agglomerative algorithms improved the hierarchical solutions obtained by agglomerative methods alone. Since agglomerative methods can do a better job of grouping together documents that form small and reasonably cohesive clusters than partitional methods, the resulting hierarchical solutions by the constrained agglomerative methods are also better than partitional methods alone for many cases.

Another surprising observation from the experimental results 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.

By looking at the FScore values for each individual class, we found that for most of the classes $\mathcal{I}_1$ can produce clusters with similar quality as UPGMA. However, $\mathcal{I}_1$ performed poorly for a few large classes. For those classes, $\mathcal{I}_1$ 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 objective function after merging two clusters S_i and S_j to obtain cluster S_r is given by,

$$\begin{aligned}\Delta\mathcal{I}_1 &= \frac{\|D_r\|^2}{n_r} - \frac{\|D_i\|^2}{n_i} - \frac{\|D_j\|^2}{n_j} = n_r\mu_r - n_i\mu_i - n_j\mu_j \\ &= (n_i+n_j)\frac{n_i^2\mu_i + n_j^2\mu_j + 2n_in_j\xi_{ij}}{(n_i+n_j)^2} - n_i\mu_i - n_j\mu_j = \frac{n_in_j}{n_i+n_j}(2\xi_{ij} - \mu_i - \mu_j). \qquad (13)\end{aligned}$$

From Equation 13, 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.

7 Concluding Remarks

In the 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 produced better hierarchical solutions than agglomerative methods, and that the constrained agglomerative methods improved the clustering solutions obtained by agglomerative or partitional methods alone. These results suggest that the poor performance of agglomerative methods may be attributed to the merging errors they make during early stages, which can be eliminated to some extent by introducing partitional constrains.

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] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] J. Han, M. Kamber, and A. K. H. Tung. Spatial clustering methods in data mining: A survey. In H. Miller and J. Han, editors, *Geographic Data Mining and Knowledge Discovery*. Taylor and Francis, 2001.

[15] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

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

[17] G. Karypis and E.H. Han. Concept indexing: A fast dimensionality reduction algorithm with applications to document retrieval & categorization. Technical Report TR-00-016, Department of Computer Science, University of Minnesota, Minneapolis, 2000. Available on the WWW at URL *http://www.cs.umn.edu/~karypis*.

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

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

[20] 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.

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

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

[23] 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.

[24] 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.

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

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

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

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

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

[30] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 1999.

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

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

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

[34] Ying Zhao and George Karypis. Criterion functions for document clustering: Experiments and analysis. Technical Report TR #01–40, Department of Computer Science, University of Minnesota, Minneapolis, MN, 2001. Available on the WWW at *http://cs.umn.edu/~karypis/publications*.