# Comparison of Agglomerative and Partitional Document Clustering Algorithms*

Ying Zhao and George Karypis

Department of Computer Science, University of Minnesota, Minneapolis, MN 55455

{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, and in greatly improving the retrieval performance either via cluster-driven dimensionality reduction, term-weighting, or query expansion. This ever-increasing importance of document clustering and the expanded range of its applications led to the development of a number of novel algorithms and new clustering criterion functions, especially in the context of partitional clustering.

The focus of this paper is to experimentally evaluate the performance of seven different global criterion functions in the context of agglomerative clustering algorithms and compare the clustering results of agglomerative algorithms and partitional algorithms for each one of the criterion functions. Our experimental evaluation shows that for every criterion function, partitional algorithms always lead to better clustering results 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.

## 1 Introduction

The topic of clustering has been extensively studied in many scientific disciplines and over the years a variety of different algorithms have been developed. Two recent surveys on the topics [12, 11] offer a comprehensive summary of the different applications and algorithms. These algorithms can be categorized based on their underlying methodology, as either *agglomerative* [22, 16, 8, 9, 15] or *partitional* approaches [18, 13, 19, 3, 27, 10, 24, 2, 5].

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 [4, 17, 1, 23]. However, there was the common belief that in terms of clustering quality, partitional algorithms are actually inferior and less effective than their agglomerative counterparts. This belief was 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, in the context of document retrieval, the hierarchical algorithms seems to perform better than the partitional algorithms for retrieving relevant documents [25]. Similarly, Larsen [17] also observed that group average greedy agglomerative clustering outperformed various partitional clustering algorithms in document data sets from TREC and Reuters. However, most of the previous comparisons did not address the effect of a key characteristic of both partitional clustering algorithms and agglomerative clustering algorithms, which is the criterion function whose optimization drives the entire clustering process. A recent study [26] on the suitability of different criterion functions to the problem of partitionally clustering document datasets showed that different criterion functions do lead to substantially different results. An extensive study of the various criterion functions in the context of agglomerative clustering algorithms is needed in order to conduct a more comprehensive comparison.

The focus of this paper is to perform such a study on the suitability of different criterion functions in the context of agglomerative document clustering, and to compare agglomerative and partitional algorithms from the perspective of criterion functions. In particular, we evaluate a total of seven different global criterion functions that measure various

*This work was supported by NSF CCR-9972519, EIA-9986042, ACI-9982274, 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**

aspects of intra-cluster similarity, inter-cluster dissimilarity, and their combinations. These criterion functions utilize different views of the underlying collection, by either modeling the documents as vectors in a high dimensional space, or by modeling the collection as a graph. We also compare the hierarchical clustering results obtained from repeated bisections and agglomerative clustering under the same criterion function. In the former case, the criterion function is used in cluster refinement, whereas the later case, the criterion function determines the next pair of clusters to be merged. The commonly used method group average (UPGMA) [13] is among the seven criterion functions.

We experimentally evaluated the performance of these criterion functions using partitional and agglomerative clustering algorithms in twelve different data sets obtained from various sources. Our experiments showed that in the context of agglomerative document clustering, different criterion functions also lead to substantially different results. However, the relative performance of various criterion functions in agglomerative clustering algorithms does differ from the relative performance in partitional clustering algorithms as we found in [26].

Our experimental results also showed that partitional algorithms always generate better hierarchical clustering solutions by repeated bisection than agglomerative algorithms for all the criterion functions. The observed superiority of partitional algorithms 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 better clustering performance.

# 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 [21] 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 $i$th term in the document and $df_i$ is the number of documents that contain the $i$th 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 Document Clustering

In this section, we first discuss the various criterion functions that we evaluated in the context of agglomerative clustering and then describe the details of the agglomerative and partitional clustering algorithms that we used in our experiments.

## 3.1 Clustering Criterion Functions

The clustering criterion functions introduced in this section can be classified into four groups: internal, external, hybrid and graph-based. The ***internal*** criterion functions focuses 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. Most of these criterion functions have been proposed in the context of partitional clustering algorithms. However, they can also be used for agglomerative clustering. Table 1 lists all the seven criterion functions that we will study. In these formulas, $D$ is the composite vector of the entire document collection, and $C_r$ and $D_r$ are the centroid and composite vectors of the $S_r$ cluster, respectively.

| Criterion Function | Category | Optimazition Function |
|---|---|---|
| $\mathcal{I}_1$ | Internal | maximize $\sum_{r=1}^{k} n_r \left( \frac{1}{n_r^2} \sum_{d_i, d_j \in S_r} \cos(d_i, d_j) \right) = \sum_{r=1}^{k} \frac{\|D_r\|^2}{n_r}$. (1) |
| $\mathcal{I}_2$ | Internal | maximize $\sum_{r=1}^{k} \sum_{d_i \in S_r} \cos(d_i, C_r) = \sum_{r=1}^{k} \sum_{d_i \in S_r} \frac{{d_i}^t C_r}{\|C_r\|} = \sum_{r=1}^{k} \|D_r\|$. (2) |
| $\mathcal{E}_1$ | External | minimize $\sum_{r=1}^{k} n_r \frac{{D_r}^t D}{\|D_r\|}$. (3) |
| $\mathcal{G}_1$ | Graph-Based, Hybrid | minimize $\sum_{r=1}^{k} \frac{{D_r}^t D}{\|D_r\|^2}$. (4) |
| $\mathcal{G}'_1$ | Graph-Based, Hybrid | minimize $\sum_{r=1}^{k} n_r^2 \frac{{D_r}^t D}{\|D_r\|^2}$. (5) |
| $\mathcal{H}_1$ | Hybrid | maximize $\frac{\mathcal{I}_1}{\mathcal{E}_1} = \frac{\sum_{r=1}^{k} \|D_r\|^2 / n_r}{\sum_{r=1}^{k} n_r {D_r}^t D / \|D_r\|}$, (6) |
| $\mathcal{H}_2$ | Hybrid | maximize $\frac{\mathcal{I}_2}{\mathcal{E}_1} = \frac{\sum_{r=1}^{k} \|D_r\|}{\sum_{r=1}^{k} n_r {D_r}^t D / \|D_r\|}$. (7) |

**Table 1**: Summary of various clustering criterion functions.

The $\mathcal{I}_1$ criterion function (Equation 1) maximizes the sum of the average pair-wise cosine similarities between the documents assigned to each cluster, weighted according to the size of each cluster, and is identical to the group-average heuristic used in agglomerative clustering [4]. The $\mathcal{I}_2$ criterion function (Equation 2) is used by the popular vector-space variant of the $K$-means algorithm [4, 17, 6, 23, 14]. The goal is to find clustering solutions that maximize the cosine similarity between each document and the centroid of the cluster that is assigned to. The $\mathcal{E}_1$ criterion function (Equation 3) tries to separate the documents of each cluster from the entire collection. In particular, it minimizes the cosine between the centroid vector of each cluster to the centroid vector of the entire collection. The contribution of each cluster is weighted based on the cluster size. The $\mathcal{G}_1$ criterion function (Equation 4) is *graph based* and is derived from the recently introduced MinMaxCut [5] objective, that simultaneously minimizes the *cut* between the cluster and every other document in the collection and maximizes the self-similarity within the cluster itself. It can be shown that $\mathcal{G}_1$ combines each cluster quality weighted proportional to the inverse of the size of the cluster [26]. As a variant to the $\mathcal{G}_1$ function, the $\mathcal{G}'_1$ criterion function (Equation 5) weights the quality measure of a cluster by its size to ensure that large clusters contribute more to the overall function value. Finally, the two *hybrid* criterion functions $\mathcal{H}_1$ (Equation 6) and $\mathcal{H}_2$ (Equation 7) were obtained by combining $\mathcal{I}_1$ with $\mathcal{E}_1$, and $\mathcal{I}_2$ with $\mathcal{E}_1$, respectively. A more detailed description and motivation of all the criterion functions can be found in [26].

## 3.2 Hierarchical Agglomerative Clustering Algorithms

Hierarchical agglomerative algorithms find the clusters by initially assigning each object to its own cluster and then repeatedly merging pairs of clusters until a certain stopping criterion is met. The various clustering criterion functions that we are considering in this paper can be used to determine the pairs of clusters to be merged at each step in the following way.

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. Depending on the desired solution, this process continues until either there are only $k$ cluster left, or when the entire agglomerative tree has been obtained.

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 upper bounded by $O(n^2m)$, where $n$ is the number of documents and $m$ is the number of terms. However, for most realistic document datasets, the average number of terms in each document is much smaller than $m$ and often ranges within 100-300 terms, irrespective of $m$. For this reason, the pairwise similarity between all the documents can be computed in $O(n^2)$ time by using appropriate sparse data structures.

The second step is the repeated selection of 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 $l$th 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 the $\mathcal{I}_1$, $\mathcal{I}_2$, $\mathcal{E}_1$, $\mathcal{G}_1$, and $\mathcal{G}'_1$ criterion functions because 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 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 $l$th 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.

## 3.3 Partitional Clustering Algorithms

Partitional clustering algorithms compute a $k$-way clustering of a set of documents either directly or via a sequence of repeated bisections. A direct $k$-way clustering is commonly computed as follows. Initially, a set of $k$ documents is selected from the collection to act as the *seeds* of the $k$ clusters. Then, for each document, its similarity to these $k$ seeds is computed, and it is assigned to the cluster corresponding to its most similar seed. This forms the initial $k$-way clustering. This clustering is then repeatedly refined so that it optimizes the desired clustering criterion function. A $k$-way partitioning via repeated bisections is obtained by recursively applying the above algorithm to compute 2-way clustering (*i.e.*, bisections). Initially, the documents are partitioned into two clusters, then one of these clusters is selected and is further bisected, and so on. This process continues $k-1$ times, leading to $k$ clusters. Each of these bisections is performed so that the resulting two-way clustering solution optimizes the particular criterion function. However, the overall $k$-way clustering solution will not necessarily be at a local minima with respect to the criterion function. The key step in this algorithm is the method used to select which cluster to bisect next. In all of our experiments, we chose to select the largest cluster, as this approach lead to reasonably good and balanced clustering solutions [23]. Extensive experiments presented in [26], show that the clustering solutions obtained via repeated bisections are comparable or better than those produced via direct clustering. Furthermore, their computational requirements are much smaller, as they have to solve a simpler optimization problem at each step. For this reason, in all of our experiments we use this approach to compute partitional clustering solutions.

A key step in the above class of partitional algorithms is the method used to refine the initial clustering solution. 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*

[7]. 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 algorithms used during the refinement phase are greedy in nature, they are not guaranteed to converge to a global minima, and the local minima solution they obtain depends on the particular set of seed documents that were selected to obtain 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.

One of the differences between partitional and agglomerative clustering algorithms is the fact that the former do not generate an agglomerative tree. Agglomerative trees are very useful as they contain information on how the different documents are related with each other, at different levels of granularity. One way of inducing an agglomerative tree from a partitional clustering solution is to do it in such a way so that it preserves the already computed $k$-way clustering. This can be done in two steps. First, we build an agglomerative tree for the documents belonging to each one of the clusters, and then we combine these trees by building an agglomerative tree, whose leaves are the partitionally discovered clusters. This approach ensures that the $k$-way clustering solution induced by the overall tree is identical to the $k$-way clustering solution computed by the partitional algorithm. Both of these trees are constructed so that they optimize the same criterion function that was used to derived the partitional clustering solution.

# 4 Experimental Results

We experimentally evaluated the performance of the different clustering criterion functions, in the context of hierarchical agglomerative clustering, on 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.

## 4.1 Document Collections

In our experiments, we used a total of twelve different datasets, whose general characteristics are summarized in Table 2. 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 [20]. Moreover, any term that occurs in fewer than two documents was eliminated. Due to space constraints, we did not include a detailed description of these data sets, but it can be found in [26].

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

## 4.2 Experimental Methodology and Metrics

For each one of the different datasets we obtained 10- and 20-way clustering solutions that optimized the various criterion functions, in the context of hierarchical and partitional agglomerative clustering. We computed both the $k$-way clustering and the entire hierarchical agglomerative tree that was derived from each one of the criterion functions.

The quality of a clustering solution was measured using two different methods that look at the class labels of the documents assigned to each cluster. The first method uses the widely used *entropy* metric that looks at how the various classes of documents are distributed within each one of the $k$ clusters. The second method, determines the quality of a clustering solution by analyzing the entire agglomerative tree that is produced by the particular clustering criterion function.

**Entropy Measure** Given a particular cluster $S_r$ of size $n_r$, the entropy of this cluster 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 $i$th class that were assigned to the $r$th cluster. The entropy of the entire clustering solution is then defined to be the sum of the individual cluster entropies weighted according to the cluster size. That is,

$$Entropy = \sum_{r=1}^{k} \frac{n_r}{n} E(S_r).$$

A perfect clustering solution will be the one that leads to clusters that contain documents from only a single class, in which case the entropy will be zero. In general, the smaller the entropy values, the better the clustering solution is.

**FScore Measure** One of the limitations of evaluating the quality of a clustering solution produced by agglomerative methods is that it is very sensitive on the choice of $k$, and the derived clustering solution may appear unnecessarily poor due to the presence of a few outlier documents. For this reason, a better approach for comparing the clustering solutions produced by agglomerative methods is to compare the overall set of clusters that are represented in the tree they produce. One such measure is FScore measure, introduced by [17]. 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 $n_{ri}/n_i$. 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.

## 4.3 Evaluation of the Criterion Functions for Agglomerative Clustering

Our first set of experiments was focused on evaluating the quality of the clustering solutions produced by the various criterion functions when they were used to guide the agglomeration process.

The entropy results for the various datasets and criterion functions for 10- and 20-way clustering solutions are

shown in Table 3. The results in this table are provided primarily for completeness and in order to evaluate the various criterion functions we actually summarized these results by looking at the average performance of each criterion function over the entire set of datasets.

| | 10-way Clustering | | | | | | | 20-way Clustering | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{E}_1$ | $\mathcal{G}_1$ | $\mathcal{G}'_1$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{E}_1$ | $\mathcal{G}_1$ | $\mathcal{G}'_1$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ |
| tr31 | 0.270 | 0.226 | 0.313 | 0.269 | 0.317 | 0.293 | 0.284 | 0.230 | 0.196 | 0.228 | 0.187 | 0.241 | 0.227 | 0.224 |
| tr41 | 0.472 | 0.262 | 0.320 | 0.376 | 0.321 | 0.329 | 0.327 | 0.223 | 0.187 | 0.226 | 0.239 | 0.222 | 0.229 | 0.220 |
| re0 | 0.488 | 0.455 | 0.492 | 0.489 | 0.488 | 0.470 | 0.478 | 0.434 | 0.405 | 0.425 | 0.426 | 0.430 | 0.382 | 0.414 |
| re1 | 0.548 | 0.470 | 0.509 | 0.513 | 0.516 | 0.489 | 0.494 | 0.463 | 0.383 | 0.406 | 0.434 | 0.421 | 0.388 | 0.400 |
| k1a | 0.567 | 0.472 | 0.546 | 0.484 | 0.537 | 0.486 | 0.506 | 0.462 | 0.418 | 0.481 | 0.441 | 0.485 | 0.399 | 0.448 |
| k1b | 0.276 | 0.180 | 0.280 | 0.174 | 0.272 | 0.173 | 0.235 | 0.211 | 0.163 | 0.241 | 0.158 | 0.237 | 0.152 | 0.203 |
| wap | 0.520 | 0.448 | 0.528 | 0.485 | 0.537 | 0.474 | 0.503 | 0.435 | 0.397 | 0.455 | 0.446 | 0.484 | 0.406 | 0.426 |
| fbis | 0.483 | 0.458 | 0.448 | 0.490 | 0.457 | 0.456 | 0.446 | 0.418 | 0.396 | 0.375 | 0.401 | 0.391 | 0.395 | 0.378 |
| hitech | 0.734 | 0.726 | 0.731 | 0.714 | 0.745 | 0.720 | 0.710 | 0.675 | 0.673 | 0.688 | 0.652 | 0.691 | 0.664 | 0.672 |
| la1 | 0.636 | 0.580 | 0.570 | 0.580 | 0.654 | 0.561 | 0.611 | 0.590 | 0.530 | 0.541 | 0.542 | 0.603 | 0.531 | 0.561 |
| la2 | 0.673 | 0.540 | 0.594 | 0.535 | 0.607 | 0.561 | 0.531 | 0.513 | 0.515 | 0.562 | 0.504 | 0.562 | 0.536 | 0.512 |
| reviews | 0.606 | 0.442 | 0.460 | 0.506 | 0.482 | 0.436 | 0.402 | 0.408 | 0.374 | 0.431 | 0.387 | 0.407 | 0.383 | 0.377 |

**Table 3**: The entropy values for the various datasets and criterion functions for the clustering solutions obtained via hierarchical agglomerative clustering.

One way of summarizing the results is to average the entropies for each criterion function over the twelve different datasets. However, since the clustering quality for different datasets is quite different and since the quality tends to improve as we increase the number of clusters, we felt that such simple averaging may distort the overall results. For this reason, our summarization is based on averaging relative entropies, as follows. For each dataset and value of $k$, we divided the entropy obtained by a particular criterion function by the smallest entropy obtained for that particular dataset and value of $k$ over the different criterion functions. These ratios represent the degree to which a particular criterion function performed worse than the best criterion function for that particular series of experiments. Note that for different datasets and values of $k$, the criterion function that achieved the best solution as measured by entropy may be different. These ratios are less sensitive to the actual entropy values and the particular value of $k$. We will refer to these ratios as *relative entropies*. Now, for each criterion function and value of $k$ we averaged these relative entropies over the various datasets. A criterion function that has an ***average relative entropy*** close to 1.0 will indicate that this function did the best for most of the datasets. On the other hand, if the average relative entropy is high, then this criterion function performed poorly.

The values for the relative entropies for the 10- and 20-way clustering solutions are shown in Table 4, and the average relative entropy is shown in the row labeled "Average". The entries that are boldfaced correspond to the criterion functions that performed the best.

| | 10-way Clustering | | | | | | | 20-way Clustering | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{E}_1$ | $\mathcal{G}_1$ | $\mathcal{G}'_1$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{E}_1$ | $\mathcal{G}_1$ | $\mathcal{G}'_1$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ |
| tr31 | 1.195 | 1.000 | 1.385 | 1.190 | 1.403 | 1.296 | 1.257 | 1.230 | 1.048 | 1.219 | 1.000 | 1.289 | 1.214 | 1.198 |
| tr41 | 1.802 | 1.000 | 1.221 | 1.435 | 1.225 | 1.256 | 1.248 | 1.193 | 1.000 | 1.209 | 1.278 | 1.187 | 1.225 | 1.176 |
| re0 | 1.073 | 1.000 | 1.081 | 1.075 | 1.073 | 1.033 | 1.051 | 1.136 | 1.060 | 1.113 | 1.115 | 1.126 | 1.000 | 1.084 |
| re1 | 1.166 | 1.000 | 1.083 | 1.091 | 1.098 | 1.040 | 1.051 | 1.209 | 1.000 | 1.060 | 1.133 | 1.099 | 1.013 | 1.044 |
| k1a | 1.201 | 1.000 | 1.157 | 1.025 | 1.138 | 1.030 | 1.072 | 1.158 | 1.048 | 1.206 | 1.105 | 1.216 | 1.000 | 1.123 |
| k1b | 1.595 | 1.040 | 1.618 | 1.006 | 1.572 | 1.000 | 1.358 | 1.388 | 1.072 | 1.586 | 1.039 | 1.559 | 1.000 | 1.336 |
| wap | 1.161 | 1.000 | 1.179 | 1.083 | 1.199 | 1.058 | 1.123 | 1.096 | 1.000 | 1.146 | 1.123 | 1.219 | 1.023 | 1.073 |
| fbis | 1.083 | 1.027 | 1.004 | 1.099 | 1.025 | 1.022 | 1.000 | 1.115 | 1.056 | 1.000 | 1.069 | 1.043 | 1.053 | 1.008 |
| hitech | 1.034 | 1.023 | 1.030 | 1.006 | 1.049 | 1.014 | 1.000 | 1.035 | 1.032 | 1.055 | 1.000 | 1.060 | 1.018 | 1.031 |
| la1 | 1.134 | 1.034 | 1.016 | 1.034 | 1.166 | 1.000 | 1.089 | 1.113 | 1.000 | 1.021 | 1.023 | 1.138 | 1.002 | 1.058 |
| la2 | 1.267 | 1.017 | 1.119 | 1.008 | 1.143 | 1.056 | 1.000 | 1.018 | 1.022 | 1.115 | 1.000 | 1.115 | 1.063 | 1.016 |
| reviews | 1.507 | 1.100 | 1.144 | 1.259 | 1.199 | 1.085 | 1.000 | 1.091 | 1.000 | 1.152 | 1.035 | 1.088 | 1.024 | 1.008 |
| Average | 1.268 | **1.020** | 1.170 | 1.109 | 1.191 | 1.074 | 1.104 | 1.148 | **1.028** | 1.157 | 1.077 | 1.178 | 1.053 | 1.096 |

**Table 4**: Relative entropies of the different datasets for different criterion functions for the clustering solutions obtained via hierarchical agglomerative clustering. Boldfaced entries represent the best performing scheme.

A number of observations can be made by analyzing the results in Table 4. First, the $\mathcal{I}_2$ criterion function leads to the best solutions irrespective of the number of clusters for most of the data sets. Over the entire set of experiments, this method is either the best or always within 10% of the best solution. On the average, the $\mathcal{I}_2$ criterion function

outperforms the other criterion functions by 5%–27% and 3%–13% for 10- and 20-way clustering, respectively. Second, the $\mathcal{H}_1$ criterion function performs the next best and overall is within 5% and 3% of the best solution for 10- and 20-way clustering, respectively. Third, the $\mathcal{H}_2$ criterion function performs quite well when the number of clusters is relatively small. It outperforms the $\mathcal{I}_2$ criterion function in several data sets—a trend also observed in the case of partitional clustering algorithms [26]. Fourth, the $\mathcal{I}_1$, $\mathcal{E}_1$, and the $\mathcal{G}'_1$ criterion functions lead to clustering solutions that are consistently worse than the solutions obtained using the other criterion functions regardless of the number of clusters. Finally, the $\mathcal{G}_1$ criterion function always performs somewhere in the middle of the road. Also note that the relative performance of the various criterion functions remains relatively the same for the 10- and 20-way clustering solutions.

The FScore-based comparison of the trees produced by the various criterion functions for the different datasets is shown in Table 5. The sub-table labeled "FScore" shows the actual FScore values of the trees produced by the different criterion functions, whereas the sub-table labeled "Relative FScore" shows the ratio of the FScore values for a particular criterion function relative to the best FScore value achieved for that particular data set. Since, higher FScore values are better, all these relative FScore values are less than one, and the scheme that has a relative FScore value of one, is the best. Finally, the row labeled "Average" shows the *average relative FScore value* over the different data sets. A criterion function that has an average relative FScore value close to 1.0 will indicate that this function did the best for most of the datasets. On the other hand, if the average relative FScore is small, then this criterion function performed poorly.

**FScore**

| Name | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{E}_1$ | $\mathcal{G}_1$ | $\mathcal{G}'_1$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ |
|---|---|---|---|---|---|---|---|
| tr31 | 0.756 | 0.844 | 0.795 | 0.796 | 0.762 | 0.818 | 0.789 |
| tr41 | 0.694 | 0.779 | 0.772 | 0.754 | 0.769 | 0.728 | 0.729 |
| re0 | 0.561 | 0.576 | 0.575 | 0.581 | 0.571 | 0.603 | 0.585 |
| re1 | 0.607 | 0.684 | 0.651 | 0.617 | 0.632 | 0.662 | 0.660 |
| k1a | 0.583 | 0.605 | 0.544 | 0.603 | 0.538 | 0.601 | 0.579 |
| k1b | 0.836 | 0.896 | 0.816 | 0.844 | 0.812 | 0.889 | 0.858 |
| wap | 0.588 | 0.618 | 0.577 | 0.618 | 0.576 | 0.607 | 0.580 |
| fbis | 0.592 | 0.639 | 0.643 | 0.624 | 0.617 | 0.629 | 0.637 |
| hitech | 0.480 | 0.480 | 0.489 | 0.471 | 0.489 | 0.476 | 0.468 |
| la1 | 0.580 | 0.648 | 0.634 | 0.612 | 0.607 | 0.635 | 0.620 |
| la2. | 0.610 | 0.681 | 0.633 | 0.644 | 0.610 | 0.613 | 0.697 |
| reviews | 0.642 | 0.689 | 0.690 | 0.654 | 0.734 | 0.648 | 0.727 |

**Relative FScore**

| Name | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{E}_1$ | $\mathcal{G}_1$ | $\mathcal{G}'_1$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ |
|---|---|---|---|---|---|---|---|
| tr31 | 0.896 | 1.000 | 0.942 | 0.943 | 0.903 | 0.969 | 0.935 |
| tr41 | 0.891 | 1.000 | 0.991 | 0.968 | 0.987 | 0.935 | 0.936 |
| re0 | 0.930 | 0.955 | 0.954 | 0.964 | 0.947 | 1.000 | 0.970 |
| re1 | 0.887 | 1.000 | 0.952 | 0.902 | 0.924 | 0.968 | 0.965 |
| k1a | 0.964 | 1.000 | 0.899 | 0.997 | 0.889 | 0.993 | 0.957 |
| k1b | 0.933 | 1.000 | 0.911 | 0.942 | 0.906 | 0.992 | 0.958 |
| wap | 0.951 | 1.000 | 0.934 | 1.000 | 0.932 | 0.982 | 0.939 |
| fbis | 0.921 | 0.994 | 1.000 | 0.970 | 0.960 | 0.978 | 0.991 |
| hitech | 0.982 | 0.982 | 1.000 | 0.963 | 1.000 | 0.973 | 0.957 |
| la1 | 0.895 | 1.000 | 0.978 | 0.944 | 0.937 | 0.980 | 0.957 |
| la2 | 0.875 | 0.977 | 0.908 | 0.924 | 0.875 | 0.879 | 1.000 |
| reviews | 0.875 | 0.939 | 0.940 | 0.891 | 1.000 | 0.883 | 0.990 |
| Average | 0.917 | **0.987** | 0.951 | 0.951 | 0.938 | 0.961 | 0.963 |

**Table 5**: The actual and average FScore values for the various datasets and criterion functions for the tree obtained via hierarchical agglomerative clustering.

Comparing the FScore-based results with those based on entropy, we can see that the relative performance of most of the criterion functions remains the same. $\mathcal{I}_2$ still leads to the best solutions whereas $\mathcal{I}_1$ leads to the worst. Over the entire set of experiments, $\mathcal{I}_2$ is either the best or always within 6% of the best solution. On the average, the $\mathcal{I}_2$ criterion function outperforms the other criterion functions with a 3%–8% lead. On the other hand, $\mathcal{I}_1$ is always 2%–11% worse then the best results. The remaining five criterion functions perform similarly with each other, with $\mathcal{H}_1$ and $\mathcal{H}_2$ doing somewhat better than the rest (within 3% of the best solution). The key difference between the FScore- and entropy-based results is that the $\mathcal{E}_1$, $\mathcal{G}'_1$ and $\mathcal{H}_2$ criterion functions appear to have a better relative performance when measured using FScore instead of entropy. Since FScore measures the quality of the entire tree and not any fixed number of clusters, one explanation of the difference between FScore and entropy may be that these criterion functions generate reasonably good agglomerative solutions, but they are sensitive on outlier objects, that tends to produce somewhat worse fixed *k* clustering solutions.

Comparing the observations made from Table 5 and Table 4 with the corresponding findings for partitional clustering algorithms in our previous study [26], they do share several common trends. The $\mathcal{I}_2$ criterion function performs the best, the $\mathcal{I}_1$ criterion function performs the worst and $\mathcal{G}'_1$ is in the middle. However, the $\mathcal{E}_1$ criterion function does not work well in agglomerative algorithms, and $\mathcal{I}_2$ outperforms every other criterion function, whereas in partitional algorithms, $\mathcal{H}_1$ and $\mathcal{H}_2$ can produce competitive solutions.

## 4.4 Comparison of Agglomerative vs. Partitional Clustering Solutions

Our second set of experiments was focused on comparing the clustering solutions produced by the agglomerative algorithms to those produced by the partitional algorithm based on repeated bisections (described in Section 3.3).

Table 6 shows the entropy of the clustering solutions produced by the various criterion functions for 10- and 20-way partitional clustering. Recall from Section 3.3, we computed 10 partitional clustering solutions with various sets of initial seed documents for each dataset and criterion function, and we chose the clustering solution that achieves the best value of the corresponding criterion function used in clustering among the 10 trails. Table 7 shows the entropies of the same clustering solutions relative to those achieved by the corresponding agglomerative approaches. Since, smaller entropy values are better, any ratios greater than one indicate that the clustering solution produced by the agglomerative approach is better, whereas any ratios smaller than one indicate that partitional does better. Also, the row labeled "Average" shows the average of these relative entropies over all the different data sets.

| | 10-way Clustering | | | | | | | 20-way Clustering | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{E}_1$ | $\mathcal{G}_1$ | $\mathcal{G}'_1$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{E}_1$ | $\mathcal{G}_1$ | $\mathcal{G}'_1$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ |
| tr31 | 0.320 | 0.163 | 0.211 | 0.182 | 0.287 | 0.192 | 0.182 | 0.221 | 0.122 | 0.147 | 0.143 | 0.188 | 0.159 | 0.163 |
| tr41 | 0.311 | 0.261 | 0.296 | 0.282 | 0.322 | 0.267 | 0.277 | 0.200 | 0.152 | 0.188 | 0.180 | 0.193 | 0.174 | 0.173 |
| re0 | 0.415 | 0.412 | 0.411 | 0.396 | 0.410 | 0.389 | 0.402 | 0.319 | 0.328 | 0.350 | 0.304 | 0.344 | 0.348 | 0.331 |
| re1 | 0.433 | 0.414 | 0.397 | 0.406 | 0.414 | 0.410 | 0.399 | 0.344 | 0.311 | 0.323 | 0.315 | 0.337 | 0.335 | 0.312 |
| k1a | 0.424 | 0.424 | 0.404 | 0.410 | 0.412 | 0.419 | 0.401 | 0.338 | 0.316 | 0.319 | 0.322 | 0.327 | 0.290 | 0.324 |
| k1b | 0.187 | 0.172 | 0.152 | 0.156 | 0.166 | 0.183 | 0.133 | 0.131 | 0.116 | 0.120 | 0.131 | 0.119 | 0.103 | 0.116 |
| wap | 0.431 | 0.422 | 0.395 | 0.418 | 0.412 | 0.411 | 0.401 | 0.341 | 0.314 | 0.330 | 0.316 | 0.338 | 0.307 | 0.323 |
| fbis | 0.400 | 0.394 | 0.446 | 0.408 | 0.408 | 0.400 | 0.440 | 0.336 | 0.344 | 0.373 | 0.330 | 0.343 | 0.334 | 0.360 |
| hitech | 0.666 | 0.583 | 0.575 | 0.602 | 0.589 | 0.593 | 0.571 | 0.624 | 0.531 | 0.541 | 0.553 | 0.541 | 0.551 | 0.535 |
| la1 | 0.477 | 0.362 | 0.395 | 0.418 | 0.396 | 0.421 | 0.384 | 0.421 | 0.346 | 0.370 | 0.390 | 0.371 | 0.385 | 0.372 |
| la2 | 0.403 | 0.351 | 0.401 | 0.399 | 0.393 | 0.375 | 0.366 | 0.357 | 0.324 | 0.349 | 0.367 | 0.344 | 0.348 | 0.325 |
| reviews | 0.359 | 0.299 | 0.232 | 0.288 | 0.283 | 0.316 | 0.254 | 0.281 | 0.249 | 0.199 | 0.230 | 0.245 | 0.278 | 0.209 |

**Table 6**: The entropy values for the various datasets and criterion functions for the clustering solutions obtained via partitional clustering.

| | 10-way Clustering | | | | | | | 20-way Clustering | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{E}_1$ | $\mathcal{G}_1$ | $\mathcal{G}'_1$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{E}_1$ | $\mathcal{G}_1$ | $\mathcal{G}'_1$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ |
| tr31 | 1.185 | 0.721 | 0.674 | 0.677 | 0.905 | 0.655 | 0.641 | 0.961 | 0.622 | 0.645 | 0.765 | 0.780 | 0.700 | 0.728 |
| tr41 | 0.659 | 0.996 | 0.925 | 0.750 | 1.003 | 0.812 | 0.847 | 0.897 | 0.813 | 0.832 | 0.753 | 0.869 | 0.760 | 0.786 |
| re0 | 0.850 | 0.905 | 0.835 | 0.810 | 0.840 | 0.828 | 0.841 | 0.735 | 0.810 | 0.824 | 0.714 | 0.800 | 0.911 | 0.800 |
| re1 | 0.790 | 0.881 | 0.780 | 0.791 | 0.802 | 0.838 | 0.808 | 0.743 | 0.812 | 0.796 | 0.726 | 0.800 | 0.863 | 0.780 |
| k1a | 0.748 | 0.898 | 0.740 | 0.847 | 0.767 | 0.862 | 0.792 | 0.732 | 0.756 | 0.663 | 0.730 | 0.674 | 0.727 | 0.723 |
| k1b | 0.678 | 0.956 | 0.543 | 0.897 | 0.610 | 1.058 | 0.566 | 0.621 | 0.712 | 0.498 | 0.829 | 0.502 | 0.678 | 0.571 |
| wap | 0.829 | 0.942 | 0.748 | 0.862 | 0.767 | 0.867 | 0.797 | 0.784 | 0.791 | 0.725 | 0.709 | 0.698 | 0.756 | 0.758 |
| fbis | 0.828 | 0.860 | 0.996 | 0.833 | 0.893 | 0.877 | 0.987 | 0.804 | 0.869 | 0.995 | 0.823 | 0.877 | 0.846 | 0.952 |
| hitech | 0.907 | 0.803 | 0.787 | 0.843 | 0.791 | 0.824 | 0.804 | 0.924 | 0.789 | 0.786 | 0.848 | 0.783 | 0.830 | 0.796 |
| la1 | 0.750 | 0.624 | 0.693 | 0.721 | 0.606 | 0.750 | 0.628 | 0.714 | 0.653 | 0.684 | 0.720 | 0.615 | 0.725 | 0.663 |
| la2 | 0.599 | 0.650 | 0.675 | 0.746 | 0.647 | 0.668 | 0.689 | 0.696 | 0.629 | 0.621 | 0.728 | 0.612 | 0.649 | 0.635 |
| reviews | 0.592 | 0.676 | 0.504 | 0.569 | 0.587 | 0.725 | 0.632 | 0.689 | 0.666 | 0.462 | 0.594 | 0.602 | 0.726 | 0.554 |
| Average | 0.785 | 0.826 | 0.742 | 0.779 | 0.768 | 0.814 | 0.753 | 0.775 | 0.743 | 0.711 | 0.745 | 0.718 | 0.764 | 0.729 |

**Table 7**: The entropy values for the various datasets and criterion functions for the clustering solutions obtained via partitional clustering relative to the entropy values achieved via agglomerative clustering.

A number of observations can be made by analyzing the results in Table 7. First, the clustering solutions produced by the partitional approach are consistently better then those produced by the agglomerative approach for all the criterion functions. The partitional results are 18%–29% better than the corresponding agglomerative results. Second, the improvements vary among the criterion functions. For those criterion functions that perform well in the partitional approach, but poorly in the agglomerative approach, the improvements are greater. For example, $\mathcal{E}_1$, $\mathcal{G}'_1$, and $\mathcal{H}_2$ improve by 24%–26% and 28%–29% for 10- and 20-way clustering, respectively. On the other head, the improvements are lower for those criterion functions that have similar relative performance in both agglomerative and partitional approaches. For example, $\mathcal{I}_2$ and $\mathcal{H}_1$ improve by 18%–19% and 25%–26% for 10-and 20-way clustering, respectively.

Table 8 shows the FScore of the agglomerative trees that are produced based on the 10- and 20-way partitional clustering solutions. These trees were obtained using the approach described in Section 3.3. Again, these results are primarily provided for completeness and our discussion will focus on comparing these clustering solutions to those

produced by the agglomerative algorithms. Table 9 shows the FScore values achieved by the partitional approach relative to those achieved by the corresponding agglomerative approaches. Since, larger FScore values are better, any ratios smaller than one indicate that the clustering solution produced by the agglomerative approach is better, whereas any ratios greater than one indicate that partitional does better. Also, the row labeled "Average" shows the average of these relative FScore values over all the different data sets.

| | 10-way Clustering | | | | | | | 20-way Clustering | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{E}_1$ | $\mathcal{G}_1$ | $\mathcal{G}'_1$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{E}_1$ | $\mathcal{G}_1$ | $\mathcal{G}'_1$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ |
| tr31 | 0.770 | 0.896 | 0.856 | 0.687 | 0.756 | 0.859 | 0.873 | 0.727 | 0.883 | 0.875 | 0.797 | 0.801 | 0.800 | 0.818 |
| tr41 | 0.755 | 0.823 | 0.741 | 0.756 | 0.739 | 0.786 | 0.803 | 0.705 | 0.818 | 0.742 | 0.733 | 0.671 | 0.790 | 0.745 |
| re0 | 0.603 | 0.623 | 0.612 | 0.624 | 0.668 | 0.618 | 0.608 | 0.614 | 0.621 | 0.618 | 0.612 | 0.615 | 0.627 | 0.634 |
| re1 | 0.631 | 0.731 | 0.719 | 0.707 | 0.693 | 0.710 | 0.730 | 0.625 | 0.739 | 0.725 | 0.682 | 0.695 | 0.717 | 0.724 |
| k1a | 0.647 | 0.655 | 0.660 | 0.653 | 0.636 | 0.682 | 0.672 | 0.678 | 0.669 | 0.687 | 0.661 | 0.679 | 0.712 | 0.687 |
| k1b | 0.873 | 0.873 | 0.894 | 0.868 | 0.858 | 0.885 | 0.920 | 0.893 | 0.896 | 0.892 | 0.867 | 0.867 | 0.915 | 0.870 |
| wap | 0.666 | 0.663 | 0.675 | 0.650 | 0.652 | 0.658 | 0.674 | 0.676 | 0.689 | 0.677 | 0.680 | 0.676 | 0.694 | 0.682 |
| fbis | 0.690 | 0.670 | 0.654 | 0.610 | 0.668 | 0.679 | 0.616 | 0.691 | 0.683 | 0.653 | 0.631 | 0.683 | 0.687 | 0.647 |
| hitech | 0.473 | 0.556 | 0.571 | 0.522 | 0.553 | 0.545 | 0.580 | 0.473 | 0.566 | 0.570 | 0.541 | 0.556 | 0.547 | 0.529 |
| la1 | 0.642 | 0.804 | 0.718 | 0.681 | 0.719 | 0.721 | 0.725 | 0.642 | 0.804 | 0.719 | 0.611 | 0.719 | 0.751 | 0.725 |
| la2 | 0.638 | 0.774 | 0.772 | 0.642 | 0.749 | 0.717 | 0.781 | 0.711 | 0.777 | 0.784 | 0.647 | 0.765 | 0.717 | 0.764 |
| reviews | 0.684 | 0.822 | 0.866 | 0.790 | 0.838 | 0.753 | 0.762 | 0.675 | 0.847 | 0.876 | 0.743 | 0.823 | 0.777 | 0.763 |

**Table 8**: The FScore values for the various datasets and criterion functions for the clustering solutions obtained via partitional clustering.

| | 10-way Clustering | | | | | | | 20-way Clustering | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{E}_1$ | $\mathcal{G}_1$ | $\mathcal{G}'_1$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{E}_1$ | $\mathcal{G}_1$ | $\mathcal{G}'_1$ | $\mathcal{H}_1$ | $\mathcal{H}_2$ |
| tr31 | 1.019 | 1.062 | 1.077 | 0.863 | 0.992 | 1.050 | 1.106 | 0.962 | 1.046 | 1.101 | 1.001 | 1.051 | 0.978 | 1.037 |
| tr41 | 1.088 | 1.056 | 0.960 | 1.003 | 0.961 | 1.080 | 1.102 | 1.016 | 1.050 | 0.961 | 0.972 | 0.873 | 1.085 | 1.022 |
| re0 | 1.075 | 1.082 | 1.064 | 1.074 | 1.170 | 1.025 | 1.039 | 1.094 | 1.078 | 1.075 | 1.053 | 1.077 | 1.040 | 1.084 |
| re1 | 1.040 | 1.069 | 1.104 | 1.146 | 1.097 | 1.073 | 1.106 | 1.030 | 1.080 | 1.114 | 1.105 | 1.100 | 1.083 | 1.097 |
| k1a | 1.110 | 1.083 | 1.213 | 1.083 | 1.182 | 1.135 | 1.161 | 1.163 | 1.106 | 1.263 | 1.096 | 1.262 | 1.185 | 1.187 |
| k1b | 1.044 | 0.974 | 1.096 | 1.028 | 1.057 | 0.996 | 1.072 | 1.068 | 1.000 | 1.093 | 1.027 | 1.068 | 1.029 | 1.014 |
| wap | 1.133 | 1.073 | 1.170 | 1.052 | 1.132 | 1.084 | 1.162 | 1.150 | 1.115 | 1.173 | 1.100 | 1.174 | 1.143 | 1.176 |
| fbis | 1.166 | 1.049 | 1.017 | 0.978 | 1.083 | 1.079 | 0.967 | 1.167 | 1.069 | 1.016 | 1.011 | 1.107 | 1.092 | 1.016 |
| hitech | 0.985 | 1.158 | 1.168 | 1.108 | 1.131 | 1.145 | 1.239 | 0.985 | 1.179 | 1.166 | 1.149 | 1.137 | 1.149 | 1.130 |
| la1 | 1.107 | 1.241 | 1.132 | 1.113 | 1.185 | 1.135 | 1.169 | 1.107 | 1.241 | 1.134 | 0.998 | 1.185 | 1.183 | 1.169 |
| la2 | 1.046 | 1.137 | 1.220 | 0.997 | 1.228 | 1.170 | 1.121 | 1.166 | 1.141 | 1.239 | 1.005 | 1.254 | 1.170 | 1.096 |
| reviews | 1.065 | 1.193 | 1.255 | 1.208 | 1.142 | 1.162 | 1.048 | 1.051 | 1.229 | 1.270 | 1.136 | 1.121 | 1.199 | 1.050 |
| Average | 1.073 | 1.098 | 1.123 | 1.054 | 1.113 | 1.094 | 1.108 | 1.080 | 1.111 | 1.134 | 1.055 | 1.117 | 1.111 | 1.090 |

**Table 9**: The FScore values for the various datasets and criterion functions for the clustering solutions obtained via partitional clustering relative to the FScore values achieved via agglomerative clustering.

Comparing these FScore values, we can see that the clustering solutions produced by the agglomerative approach are consistently worse than those produced by the partitional approach for all the criterion functions. The FScore values of the trees produced by partitional algorithms are 5%–13% better then those by agglomerative algorithms for various criterion functions. Also, the improvements vary among the criterion functions. Overall, the $\mathcal{E}_1$ criterion function has the greatest improvement of 12%–13% of the FScore values, whereas the $\mathcal{G}_1$ criterion function has the least improvement of 5%. Similar improvements in the range of 7%–11% are observed by the other criterion functions. Finally, the trees induced by 10- and 20-way partitional clustering have comparable FScore values. Moreover, we performed a detailed study to see the effect that $k$ has in the overall quality of the induced tree, and we found that there is a wide range of values that $k$ can take, and still achieve high quality induced trees. Due to space constraints, the results of this study have been omitted.

# 5 Concluding Remarks

In the paper we experimentally evaluated seven different criterion functions for clustering large document datasets using the agglomerative approach and compared the clustering results obtained via agglomerative algorithms with those obtained via partitional algorithms for each one of the clustering criterion functions. Our experimental results showed that in the context of agglomerative clustering, the group average criterion function ($\mathcal{I}_1$) performs the worst,

and the traditional criterion function used by the vector-space $K$-means ($\mathcal{I}_2$) leads to the best solutions.

The experimental results also showed that for every criterion function, the hierarchical trees produced by partitional algorithms are always better than those produced by agglomerative algorithms, and the various criterion functions lead to different improvements.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

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

[14] 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*.

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

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

[17] 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.

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

[19] 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.

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

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

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

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

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

[25] P. Willett. Recent trends in hierarchic document clustering: acritical review. In *Information Processing and Management, 24(5):577-597*, 1988.

[26] Y. Zhao, G. Karypis. Criterion Functions for Document Clustering: Experiments and Analysis. Technical Report #01-34, University of Minnesota, MN 2001.

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