Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 00-034

A Comparison of Document Clustering Techniques

Michael Steinbach, George Karypis, and Vipin Kumar

May 23, 2000

A Comparison of Document Clustering Techniques

Michael Steinbach George Karypis Vipin Kumar

Department of Computer Science and Egineering,
University of Minnesota
{steinbac, karypis, kumar@cs.umn.edu}

Abstract

This paper presents the results of an experimental study of some common document clustering techniques. In particular, we compare the two main approaches to document clustering, agglomerative hierarchical clustering and K-means. (For K-means we used a "standard" K-means algorithm and a variant of K-means, "bisecting" K-means.) Hierarchical clustering is often portrayed as the better quality clustering approach, but is limited because of its quadratic time complexity. In contrast, K-means and its variants have a time complexity which is linear in the number of documents, but are thought to produce inferior clusters. Sometimes K-means and agglomerative hierarchical approaches are combined so as to "get the best of both worlds." However, our results indicate that the bisecting K-means technique is better than the standard K-means approach and as good or better than the hierarchical approaches that we tested for a variety of cluster evaluation metrics. We propose an explanation for these results that is based on an analysis of the specifics of the clustering algorithms and the nature of document data.

1 Background and Motivation

Document clustering has been investigated for use in a number of different areas of text mining and information retrieval. Initially, document clustering was investigated for improving the precision or recall in information retrieval systems [Rij79, Kow97] and as an efficient way of finding the nearest neighbors of a document [BL85]. More recently, clustering has been proposed for use in browsing a collection of documents [CKPT92] or in organizing the results returned by a search engine in response to a user's query [ZEMK97]. Document clustering has also been used to automatically generate hierarchical clusters of documents [KS97]. (The automatic generation of a taxonomy of Web documents like that provided by Yahoo! (www.yahoo.com) is often cited as a goal.) A somewhat different approach [AGY99] finds the natural clusters in an already existing document taxonomy (Yahoo!), and then uses these clusters to produce an effective document classifier for new documents.

Agglomerative hierarchical clustering and K-means are two clustering techniques that are commonly used for document clustering. Agglomerative hierarchical clustering is often portrayed as "better" than K-means, although slower. A widely known study, discussed in [DJ88], indicated that agglomerative hierarchical clustering is superior to K-means, although we stress that these results were with non-document data. In the document domain, Scatter/Gather [CKPT92], a document browsing system based on clustering, uses a hybrid approach involving both K-means and agglomerative hierarchical clustering. K-means is used because of its efficiency and agglomerative hierarchical clustering is used because of its quality. Recent work to generate document hierarchies [LA99] uses some of the clustering techniques from [CKPT92] and presents a result that indicates that agglomerative hierarchical clustering is better than K-means, although this result is just for a single data set and is not one of the major results of the paper.

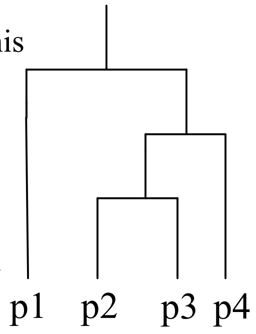
Initially we also believed that agglomerative hierarchical clustering was superior to K-means clustering, especially for building document hierarchies, and we sought to find new and better hierarchical clustering algorithms. However, during the course of our experiments we discovered that a simple and efficient variant of K-means, "bisecting" K-means, can produce clusters of documents that are better than those produced by "regular" K-means and as good or better than those produced by agglomerative hierarchical clustering techniques. We have also been able to find what we think is a reasonable explanation for this behavior.

The basic outline of this paper is as follows. Section 2 provides a brief review of agglomerative and hierarchical clustering techniques, while Section 3 reviews the vector space model for documents, particularly the aspects necessary to understand document clustering. Section 4 presents some measures of cluster quality that will be used as the basis for our comparison of different document clustering techniques and Section 5 gives some additional details about the K-means and bisecting K-means algorithms. Section 6 briefly describes the data sets used in our experiments, while sections 7 and 8 present our experimental results. More specifically, Section 7 compares three agglomerative hierarchical techniques, while Section 8 compares the best hierarchical technique to K-means and bisecting K-means. Section 9 presents our explanation for these results and Section 10 is a summary of our results.

2 Clustering Techniques

In this section we provide a brief overview of hierarchical and partitional (K-means) clustering techniques [DJ88, KR90]

Hierarchical techniques produce a nested sequence of partitions, with a single, all-inclusive cluster at the top and singleton clusters of individual points at the bottom. Each intermediate level can be viewed as combining two clusters from the next lower level (or splitting a cluster from the next higher level). The result of a hierarchical clustering algorithm can be graphically displayed as tree, called a dendogram. This tree graphically displays the merging process and the intermediate clusters. The dendogram at the right shows how four points can be merged into a single cluster. For document clustering, this dendogram provides a taxonomy, or hierarchical index.



There are two basic approaches to generating a hierarchical clustering:

a) **Agglomerative**: Start with the points as individual clusters and, at each step, merge the most similar or closest pair of clusters. This requires a definition of cluster similarity or distance.

b) **Divisive**: Start with one, all-inclusive cluster and, at each step, split a cluster until only singleton clusters of individual points remain. In this case, we need to decide, at each step, which cluster to split and how to perform the split.

Agglomerative techniques are more common, and these are the techniques that we will compare to K-means and its variants. (See [EW89] for a comparison of agglomerative hierarchical clustering methods used for document retrieval.) We summarize the traditional agglomerative hierarchical clustering procedure as follows:

Simple Agglomerative Clustering Algorithm

1. Compute the similarity between all pairs of clusters, i.e., calculate a similarity matrix whose ij^{th} entry gives the similarity between the i^{th} and j^{th} clusters.
2. Merge the most similar (closest) two clusters.
3. Update the similarity matrix to reflect the pairwise similarity between the new cluster and the original clusters.
4. Repeat steps 2 and 3 until only a single cluster remains.

In contrast to hierarchical techniques, partitional clustering techniques create a one-level (un-nested) partitioning of the data points. If *K* is the desired number of clusters, then partitional approaches typically find all *K* clusters at once. Contrast this with traditional hierarchical schemes, which bisect a cluster to get two clusters or merge two clusters to get one. Of course, a hierarchical approach can be used to generate a flat partition of *K* clusters, and likewise, the repeated application of a partitional scheme can provide a hierarchical clustering. The bisecting K-means algorithm that we present later is such an approach.

There are a number of partitional techniques, but we shall only describe the K-means algorithm which is widely used in document clustering. K-means is based on the idea that a center point can represent a cluster. In particular, for K-means we use the notion of a centroid, which is the mean or median point of a group of points. Note that a centroid almost never corresponds to an actual data point.

The basic K-means clustering technique is presented below. We elaborate on various issues in the following sections.

Basic K-means Algorithm for finding *K* clusters.

1. Select *K* points as the initial centroids.
2. Assign all points to the closest centroid.
3. Recompute the centroid of each cluster.
4. Repeat steps 2 and 3 until the centroids don't change.

3 The Vector Space Model and Document Clustering

Many issues specific to documents are discussed more fully in information retrieval texts [Rij79, Kow97]. We briefly review a few essential topics to provide a sufficient background for understanding document clustering.

For our clustering algorithms documents are represented using the vector-space model. In this model, each document, *d*, is considered to be a vector, ***d***, in the term-space (set of document "words"). In its simplest form, each document is represented by the (*TF*) vector,

$$\mathbf{d}_{tf} = (tf_1, tf_2, \ldots, tf_n),$$

where tf_i is the frequency of the i^{th} term in the document. (Normally very common words are stripped out completely and different forms of a word are reduced to one canonical form.) In addition, we use the version of this model that weights each term based on its *inverse document frequency* (IDF) in the document collection. (This discounts frequent words with little discriminating power.) Finally, in order to account for documents of different lengths, each document vector is normalized so that it is of unit length.

The similarity between two documents must be measured in some way if a clustering algorithm is to be used. There are a number of possible measures for computing the similarity between documents, but the most common one is the cosine measure, which is defined as

$$cosine(\boldsymbol{d_1}, \boldsymbol{d_2}) = (\boldsymbol{d_1} \bullet \boldsymbol{d_2}) / \|\boldsymbol{d_1}\| \, \|\boldsymbol{d_2}\| ,$$

where $\bullet$ indicates the vector dot product and $\|\boldsymbol{d}\|$ is the length of vector $\boldsymbol{d}$.

Given a set, *S,* of documents and their corresponding vector representations, we define the ***centroid*** vector $\boldsymbol{c}$ to be

$$\boldsymbol{c} = \frac{1}{|S|} \sum_{d \in S} \boldsymbol{d}$$

which is nothing more than the vector obtained by averaging the weights of the various terms present in the documents of *S.* Analogously to documents, the similarity between two centroid vectors and between a document and a centroid vector are computed using the cosine measure, i.e.,

$$cosine(\boldsymbol{d}, \boldsymbol{c}) = (\boldsymbol{d} \bullet \boldsymbol{c}) / \|\boldsymbol{d}\| \, \|\boldsymbol{c}\| = (\boldsymbol{d} \bullet \boldsymbol{c}) / \|\boldsymbol{c}\|$$

$$cosine(\boldsymbol{c_1}, \boldsymbol{c_2}) = (\boldsymbol{c_1} \bullet \boldsymbol{c_2}) / \|\boldsymbol{c_1}\| \, \|\boldsymbol{c_2}\|$$

Note that even though the document vectors are of length one, the centroid vectors will not necessarily be of unit length. (We use these two definitions in defining two of our agglomerative hierarchical techniques in Section 7, the “intra-cluster similarity and “centroid similarity” techniques, respectively.)

For K-means clustering, the cosine measure is used to compute which document centroid is closest to a given document. While a median is sometimes used as the centroid for K-means clustering, we follow the common practice of using the mean. The mean is easier to calculate than the median and has a number of nice mathematical properties.

For example, calculating the dot product between a document and a cluster centroid is equivalent to calculating the average similarity between that document and all the documents that comprise the cluster the centroid represents. (This observation is the basis of the “intra-cluster similarity” agglomerative hierarchical clustering technique in section 7.) Mathematically,

$$\boldsymbol{d_1} \bullet \boldsymbol{c} = \frac{1}{|S|} \sum_{d \in S} \boldsymbol{d_1} \bullet \boldsymbol{d} = \frac{1}{|S|} \sum_{d \in S} cosine(\boldsymbol{d_1}, \boldsymbol{d})$$

Also the square of the length of the centroid vector is just the average pairwise similarity between all points in the cluster. (This includes the similarity of each point with itself, which is just 1.) In the following section, we will use this average pairwise similarity as the basis for one of the measures for quantifying the goodness of a clustering algorithm.

$$\frac{1}{|S|^2}\sum_{\substack{d\in S\\ d'\in S}} cosine(\boldsymbol{d'},\boldsymbol{d}) = \frac{1}{|S|}\sum_{d\in S}\boldsymbol{d} \bullet \frac{1}{|S|}\sum_{d\in S}\boldsymbol{d} = \boldsymbol{c}\bullet\boldsymbol{c} = \|\boldsymbol{c}\|^2$$

4 Evaluation of Cluster Quality

For clustering, two measures of cluster "goodness" or quality are used. One type of measure allows us to compare different sets of clusters without reference to external knowledge and is called an *internal quality* measure. As mentioned in the previous section, we will use a measure of "overall similarity" based on the pairwise similarity of documents in a cluster. The other type of measures lets us evaluate how well the clustering is working by comparing the groups produced by the clustering techniques to known classes. This type of measure is called an *external quality* measure. One external measure is entropy [Sha48], which provides a measure of "goodness" for un-nested clusters or for the clusters at one level of a hierarchical clustering. Another external measure is the F-measure, which, as we use it here, is more oriented toward measuring the effectiveness of a hierarchical clustering. The F measure has a long history, but was recently extended to document hierarchies in [LA99].

There are many different quality measures and the performance and relative ranking of different clustering algorithms can vary substantially depending on which measure is used. However, if one clustering algorithm performs better than other clustering algorithms on many of these measures, then we can have some confidence that it is truly the best clustering algorithm for the situation being evaluated. As we shall see in the results sections, the bisecting K-means algorithm has the best performance for the three quality measures that we are about to describe.

4.1 Entropy

We use entropy as a measure of quality of the clusters (with the caveat that the best entropy is obtained when each cluster contains exactly one data point). Let *CS* be a clustering solution. For each cluster, the class distribution of the data is calculated first, i.e., for cluster *j* we compute p_{ij}, the "probability" that a member of cluster *j* belongs to class *i*. Then using this class distribution, the entropy of each cluster *j* is calculated using the standard formula

$E_j = -\sum_i p_{ij} \log(p_{ij})$, where the sum is taken over all classes. The total entropy for a set of clusters is calculated as the sum of the entropies of each cluster weighted by the size of each cluster: $E_{CS} = \sum_{j=1}^{m} \frac{n_j * E_j}{n}$, where n_j is the size of cluster *j*, *m* is the number of clusters, and *n* is the total number of data points.

4.2 F measure

The second external quality measure is the F measure [LA99], a measure that combines the precision and recall ideas from information retrieval [Rij79, Kow97]. We treat each cluster as if it were the result of a query and each class as if it were the desired set of documents for a query. We then calculate the recall and precision of that cluster for each given class. More specifically, for cluster *j* and class *i*

Recall(i, j) = n_{ij} / n_i

Precision(i, j) = n_{ij} / n_j

where n_{ij} is the number of members of class *i* in cluster *j*, n_j is the number of members of cluster *j* and n_i is the number of members of class *i*.

The F measure of cluster *j* and class *i* is then given by

F(i, j) = (2 * Recall(i, j) * Precision(i, j)) / ((Precision(i, j) + Recall(i, j))

For an entire hierarchical clustering the F measure of any class is the maximum value it attains at any node in the tree and an overall value for the F measure is computed by taking the weighted average of all values for the F measure as given by the following.

$$F = \sum_i \frac{n_i}{n} \max\{F(i, j)\}$$

where the max is taken over all clusters at all levels, and *n* is the number of documents.

4.3 Overall Similarity

In the absence of any external information, such as class labels, the cohesiveness of clusters can be used as a measure of cluster similarity. One method for computing the cluster cohesiveness is to use the weighted similarity of the internal cluster similarity,

$\frac{1}{|S|^2}\sum_{\substack{d \in S \\ d' \in S}} cosine(\boldsymbol{d'}, \boldsymbol{d})$.. Recall that in Section 3 it was shown that this is just the squared length of the cluster centroid, $\|\boldsymbol{c}\|^2$.

5 Details of K-means and Bisecting K-means

In this section we discuss general issues related to the K-means clustering algorithm and introduce the bisecting K-means algorithm.

There are many ways to enhance the basic K-means algorithm given in section 2, e.g., see [CKPT92, BF98, and LA99]. But to keep things simple, we chose a very simple and efficient implementation of the K-means algorithm. For instance, we select our initial centroids by randomly choosing *K* documents.

However, we did choose to update centroids incrementally, i.e., as each point is assigned to a cluster, rather than at the end of an assignment pass as is indicated in the K-means algorithm in section 2. Our reason is that we noticed that incremental updates were more effective, i.e., produced results with better overall similarity and lower entropy. The incremental version of K-means is also advocated in [LA99].

For what follows we will use a bisecting K-means algorithm as our primary clustering algorithm. This algorithm starts with a single cluster of all the documents and works in the following way:

Basic Bisecting K-means Algorithm for finding *K* clusters.

1. Pick a cluster to split.
2. Find 2 sub-clusters using the basic K-means algorithm. (Bisecting step)
3. Repeat step 2, the bisecting step, for ITER times and take the split that produces the clustering with the highest overall similarity.
4. Repeat steps 1, 2 and 3 until the desired number of clusters is reached.

There are a number of different ways to choose which cluster is split. For example, we can choose the largest cluster at each step, the one with the least overall similarity, or use a criterion based on both size and overall similarity. We did numerous runs and determined that the differences between methods were small. In the rest of this paper we split the largest remaining cluster.

Note that the bisecting K-means algorithm can produce either an un-nested (flat) clustering or a hierarchical clustering. For un-nested clusters we will often "refine" the clusters using the basic K-means algorithms, but we do not refine the nested clusters. We will provide more details later.

Strictly speaking, the bisecting K-means algorithm is a divisive hierarchical clustering algorithm, but, to avoid confusion, when we speak of hierarchical clustering algorithms we shall mean agglomerative hierarchical algorithms of the sort traditionally used to cluster documents.

Finally, note that bisecting K-means has a time complexity which is linear in the number of documents. If the number of clusters is large and if refinement is not used, then bisecting K-means is even more efficient than the regular K-means algorithm (In this case, there is no need to compare every point to every cluster centroid since to bisect a cluster we just consider the points in the cluster and their distances to two centroids.)

6 Data Sets

In all of the data sets, we have removed stop words, i.e., common words such as "a", "are", "do", and "for". We have also performed stemming using Porter's suffix-stripping algorithm. Thus, all the words sharing the same stem are considered to be the same word. For example, words "compute", "computing", and "computed" are stemmed to "comput".

The summary of documents used in this paper is shown in Table 1. The details of each data set are described here. Data sets tr31 and tr45 are from TREC-5 [trec], TREC-6 [trec], and TREC-7 [trec]. Data set fbis is from the Foreign Broadcast Information Service data of TREC-5 [trec]. Data sets la1 and la2 are from the Los Angeles Times data of TREC-5 [trec].

The class labels of tr31and tr45 came from the relevance judgments provided by "qrels.1-243.part1", "qrels.1-243.part2", "qrels.251-300.part1", "qrels.251-300.part3", "qrels.trec6.adhoc.part1", "qrels.trec7.adhoc.part1", and "qrels.trec7.adhoc.part5" [trecq]. The class labels of fbis were generated from the relevance judgments provided by TREC-5 routing query relevance "qrels.1-243" [trecq].

We collected documents that have relevance judgments and then selected documents that have just a single relevance judgment. The class labels of la1, and la2 were generated according to the section names of articles, such as "Entertainment", "Financial", "Foreign", "Metro", "National", and "Sports." Data sets re0 and re1 are from Reuters-21578 text categorization test collection Distribution 1.0 [reut]. Data set wap is from the WebACE project (WAP) [han98]. Each document corresponds to a web page listed in the subject hierarchy of Yahoo!.

Data Set	Source	Number of Documents	Number of Classes	Minimum Class Size	Maximum Class Size	Average Class Size	Number of Words
re0	Reuters-21578	1504	13	11	608	115.7	11465
re1	Reuters-21578	1657	25	10	371	663	3758
wap	WebAce	1560	20	5	341	78.0	8460
tr31	TREC	927	7	2	352	132.4	10128
tr45	TREC	690	10	14	160	69.0	8261
fbis	TREC	2463	17	38	506	144.9	2000
la1	TREC	3204	6	273	943	534.0	31472
la2	TREC	3075	6	248	905	512.5	31472

Table 1: Summary description of document sets.

7 Comparison of Agglomerative Hierarchical Techniques

In this section we compare three different agglomerative hierarchical schemes against one another. We will then compare the "best" of these algorithms against our K-means and bisecting K-means algorithms.

Before describing the results, we briefly describe the different hierarchical clustering algorithms that we used. As mentioned before, the only real difference between the different hierarchical schemes is how they choose which clusters to merge, i.e., how they choose to define cluster similarity.

7.1 Techniques

Intra-Cluster Similarity Technique (IST): This hierarchical technique looks at the similarity of all the documents in a cluster to their cluster centroid and is defined by Sim(X) $= \sum_{d \in X} cosine(\boldsymbol{d}, \boldsymbol{c})$, where d is a document in cluster, X, and $\boldsymbol{c}$ is the centroid of cluster X. The choice of which pair of clusters to merge is made by determining which pair of clusters will lead to smallest decrease in similarity. Thus, if cluster Z is formed by merging clusters X and Y, then we select X and Y so as to maximize Sim(Z) – (Sim(X) + Sim(Y)). Note that Sim(Z) – (Sim(X) + Sim(Y)) is non-positive.

Centroid Similarity Technique (CST): This hierarchical technique defines the similarity of two clusters to be the cosine similarity between the centroids of the two clusters.

UPGMA: This is the UPGMA scheme as described in [DJ88, KR90]. It defines the cluster similarity as follows,

$$similarity(cluster1,\ cluster2) = \frac{\sum\limits_{\substack{d_1 \in cluster1 \\ d_2 \in cluster2}} cosine(\boldsymbol{d}_1, \boldsymbol{d}_2)}{size(cluster1) * size(cluster2)}$$

where d_1 and d_2 are, documents, respectively, in cluster1 and cluster2.

7.2 Results

The F-measure results are shown in table 8 – larger is better. UPGMA is the best, although the two other techniques are often not much worse.

Data Set	UPGMA	Centroid Similarity	Cluster Similarity
re0	**0.5859**	0.5028	0.5392
re1	**0.6855**	0.5963	0.5509
wap	**0.6434**	0.4977	0.5633
tr31	**0.8693**	0.7431	0.7989
tr45	**0.8528**	0.7105	0.8054
Fbis	**0.6717**	0.6470	0.6233
la1	**0.6963**	0.4557	0.5977
la2	**0.7168**	0.4531	0.5817

Table 2: Comparison of the F-measure for Different Clustering Algorithms

The entropy results are shown in figures 1 - 8. Notice that UPGMA and IST are the best, with similar behavior with respect to entropy. CST does poorly. Figures 1, 5, and 6 indicate that CST does about as well with respect to entropy as do IST and UPGMA in the initial phases of agglomeration, but, at some point, starts "making mistakes" as to which clusters to merge, and its performance diverges from the other schemes from then on. In the cases represented by the other figures, this divergence happens earlier. UPGMA shows similar behavior, but only when the number of clusters is very small.

Overall, UPGMA is the best performing hierarchical technique that we investigated and we compare it to K-means and bisecting K-means in the next section.

Figure1: re0 entropy

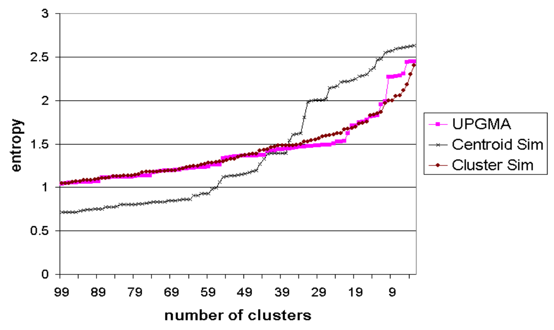


Figure 2: re1 entropy

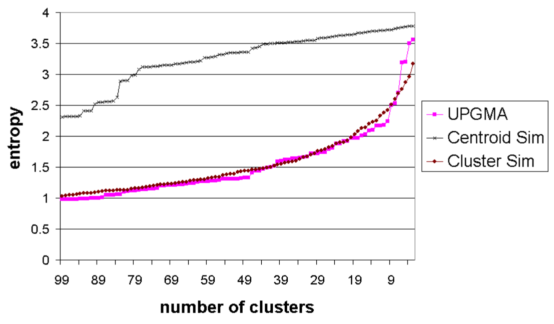


Figure 3: wap entropy

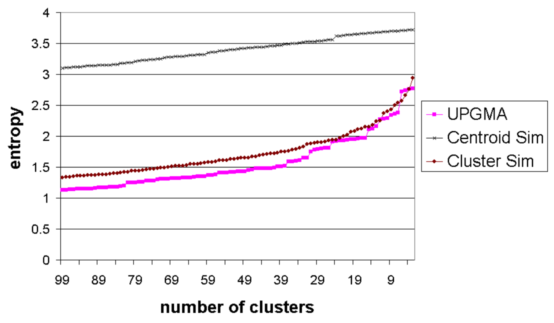


Figure 4: tr31 entropy

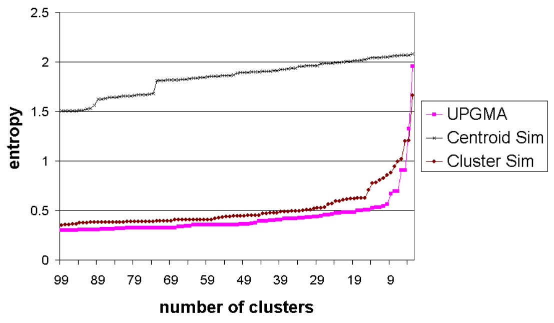


Figure 5: tr45

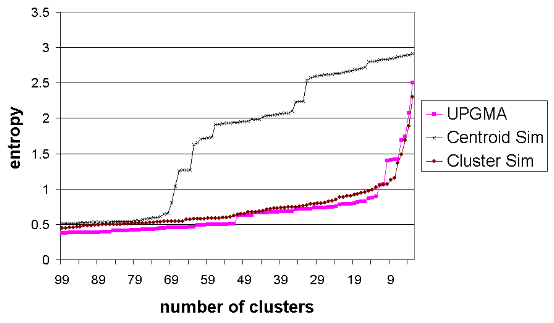


Figure 6: fbis entropy

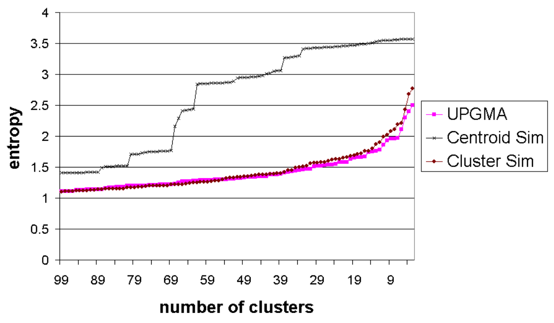


Figure 7: la1 entropy

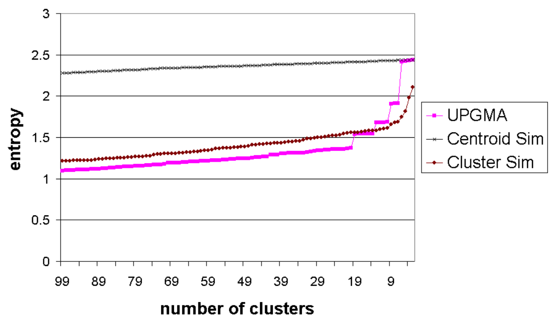
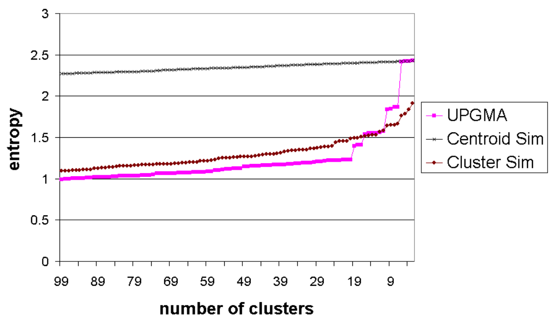


Figure 8: la2 entropy



8 Comparison of K-means, Bisecting K-means and UPGMA

8.1 Details

Here we provide some brief details of how we performed the runs that produced the results we are about to discuss.

For these experiments we equalized the number of runs for bisecting K-means versus regular K-means. If ITER is the number of trial bisections for each phase of bisecting K-means and K is the number of clusters sought, then a bisecting K-means run is equivalent to $\log_2$ (K) * ITER regular K-means runs. (There are the equivalent of ITER K-means runs for the whole set of documents for each of the $\log_2$ (K) levels in the bisection process.) For the bisecting K-means, we set ITER = 5. We did 10 runs of the bisecting K-means and for each run of the bisecting K-means we performed $\log_2$ (K) * ITER runs of a regular K-means. Since hierarchical clustering produces the same result every time, there was no need to conduct multiple runs for UPGMA.

There is also another issue that must be mentioned. Both bisecting K-means and hierarchical clustering produce clustering results that can be further "refined" by using the K-means algorithm. That is, if the centroids of the clusters produced by these two techniques are used as the initial centroids for a K-means clustering algorithm, then the K-means algorithm will change these initial centroids and readjust the clusters. The key question here is whether such refinement will improve the quality of the clusterings produced.

We also mention that hierarchical clustering with a K-means refinement is essentially a hierarchical-K-mean hybrid that is similar to techniques that other people have tried. In particular, the Scatter/Gather system [CKTP92] uses hierarchical clustering to produce "seeds" for a final K-means phase.

8.2 Results

Tables 3 - 5 show the entropy results of these runs, while tables 6 - 9 show the overall similarity results. Figure 10 also shows entropy results and is just Figure 1 with the entropy results for bisecting K-means added. Table 8 shows the comparison between F values for bisecting K-means and UPGMA. We state the three main results succinctly.

- Bisecting K-means, with or without refinement is better than regular K-means and UPGMA, with or without refinement, in most cases. Even in cases where other schemes are better, bisecting K-means is only slightly worse.
- Refinement significantly improves the performance of UPGMA for both the overall similarity and the entropy measures.
- Regular K-means, although worse than bisecting K-means, is generally better than UPGMA, even after refinement.

We make a few brief comments on the fact that we did multiple runs of K-means and bisecting K-means. For bisecting K-means, this did not improve the results much as this algorithm tends to produce relatively consistent results. For regular K-means, the results do vary quite a bit from one run to another. Thus, one run of regular K-means might produce results that are not as good as those produced by UPGMA, even without refinement.

However, even many runs of K-means or bisecting K-means are significantly quicker than a single run of a hierarchal clustering algorithm, particularly if the data sets are large. For example, for the data set, la1, with 3204 documents and 31,472 terms, a single hierarchical clustering run takes well in excess of an hour on a modern Pentium system. By comparison, a single bisecting K-means run to find 32 clusters takes less than a minute on the same machine.

Data Set	K	Bisecting K-means	Bisecting K-means with refinement	Regular K-means	Hierarchical (UPGMA)	Hierarchical (UPGMA) with refinement
re0	16	1.3305	**1.1811**	1.3839	1.9838	1.4811
re1	16	**1.6315**	1.7111	1.6896	2.0058	1.7361
wap	16	**1.5494**	1.5601	1.8557	2.0584	1.8028
tr31	16	**0.4713**	**0.4722**	0.5228	0.8107	0.5711
tr45	16	**0.6909**	**0.6927**	0.7426	1.1955	0.8665
fbis	16	1.3708	1.4053	**1.3198**	1.8594	1.3832
la1	16	0.9570	**0.9511**	1.0710	2.4046	1.2390
la2	16	0.9799	**0.9445**	0.9673	1.5955	1.1392

Table 3: Comparison of the Entropy for Different Clustering Algorithms for K = 16

Data Set	K	Bisecting K-means	Bisecting K-means with refinement	Regular K-means	Hierarchical (UPGMA)	Hierarchical (UPGMA) with refinement
re0	32	**1.0884**	1.1085	1.2064	1.5850	1.3969
re1	32	1.4229	1.3148	1.4290	1.5360	**1.2138**
wap	32	1.3314	**1.2482**	1.4422	1.7201	1.5252
tr31	32	**0.2940**	0.3327	0.4281	0.5123	0.4641
tr45	32	0.5676	0.4991	0.5293	0.7312	**0.4730**
fbis	32	**1.1872**	1.2060	1.2618	1.4538	1.2841
la1	32	**0.8659**	0.9149	1.0626	1.5375	1.0111
la2	32	0.8969	**0.8463**	0.9659	1.3568	0.9623

Table 4: Comparison of the Entropy for Different Clustering Algorithms for K = 32

Data Set	K	Bisecting K-means	Bisecting K-means with refinement	Regular K-means	Hierarchical (UPGMA)	Hierarchical (UPGMA) with refinement
re0	64	1.0662	**0.9428**	0.9664	1.3215	1.1764
re1	64	1.0249	**0.9869**	1.1177	1.1655	**0.9826**
wap	64	1.1066	**1.0783**	1.2807	1.3742	1.2825
tr31	64	0.3182	**0.2743**	0.3520	0.3985	0.3855
tr45	64	0.4613	0.4199	0.4308	0.4668	**0.3913**
fbis	64	**1.0456**	1.0876	1.0504	1.2346	1.1430
la1	64	**0.8698**	0.8748	1.0084	1.3082	1.0066
la2	64	0.7514	**0.7291**	0.9204	1.1082	0.9138

Table 5: Comparison of the Entropy for Different Clustering Algorithms for K = 64

Data Set	K	Bisecting K-means	Bisecting K-means with refinement	Regular K-means	Hierarchical (UPGMA)	Hierarchical (UPGMA) with refinement
re0	16	0.4125	0.4137	**0.4158**	0.3157	0.3784
re1	16	0.3325	**0.3341**	0.3317	0.2703	0.3084
wap	16	0.2763	**0.2771**	0.2703	0.2440	0.2533
tr31	16	0.4212	**0.4231**	0.4204	0.3560	0.3619
tr45	16	0.4182	**0.4204**	0.4190	0.3561	0.3739
fbis	16	0.4464	0.4496	**0.4514**	0.3657	0.4189
la1	16	0.2228	**0.2244**	0.2198	0.1420	0.1995
la2	16	0.2282	**0.2299**	0.2276	0.1694	0.2019

Table 6: Comparison of the Overall Similarity for K = 16

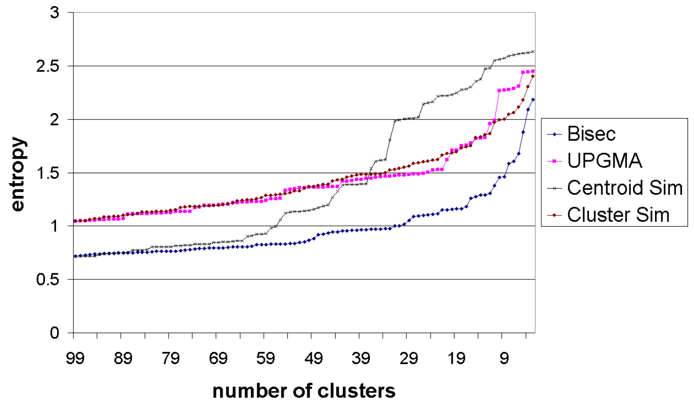
Data Set	K	Bisecting K-means	Bisecting K-means with refinement	Regular K-means	Hierarchical (UPGMA)	Hierarchical (UPGMA) with refinement
re0	32	0.4677	**0.4788**	0.4778	0.4136	0.4421
re1	32	0.3957	**0.4009**	**0.4009**	0.3369	0.3690
wap	32	0.3226	**0.3258**	0.3235	0.2786	0.2876
tr31	32	0.4802	**0.4866**	0.4795	0.4373	0.4441
tr45	32	0.4786	**0.4827**	0.4763	0.4299	0.4382
fbis	32	0.4989	0.5071	**0.5110**	0.4435	0.4827
la1	32	0.2606	**0.2640**	0.2596	0.1922	0.2247
la2	32	0.2675	**0.2738**	0.2655	0.2018	0.2437

Table 7: Comparison of Overall Similarity for K = 32

Data Set	K	Bisecting K-means	Bisecting K-means with refinement	Regular K-means	Hierarchical (UPGMA)	Hierarchical (UPGMA) with refinement
re0	64	0.5327	**0.5541**	0.5521	0.4983	0.5258
re1	64	0.4671	**0.4758**	0.4742	0.4270	0.4422
wap	64	0.3842	**0.3914**	0.3850	0.3478	0.3567
tr31	64	0.5483	**0.5536**	0.5501	0.5214	0.5232
tr45	64	0.5502	**0.5563**	0.5481	0.5168	0.5204
fbis	64	0.5495	**0.5648**	0.5627	0.5032	0.5387
la1	64	0.3047	**0.3129**	0.3080	0.2446	0.2648
la2	64	0.3177	**0.3267**	0.3212	0.2575	0.2842

Table 8: Comparison of Overall Similarity for K = 64

Figure 9: re0 entropy



Data Set	Bisecting K-means	UPGMA
re0	**0.5863**	**0.5859**
re1	**0.7067**	0.6855
wap	**0.6750**	0.6434
tr31	**0.8869**	0.8693
tr45	0.8080	**0.8528**
Fbis	**0.6814**	0.6717
la1	**0.7856**	0.6963
la2	**0.7882**	0.7168

Table 9: Comparison of the F-measure for Bisecting K-means and UPGMA

9 Explanations

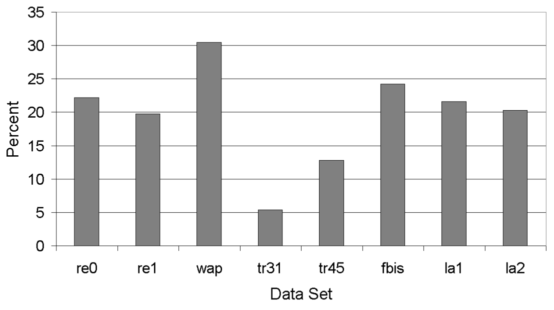
In this section we propose an explanation for why agglomerative hierarchical clustering performs poorly and why bisecting K-means does better than K-means. The arguments that we present here, while plausible, are preliminary. In future work, we hope to verify our logic by using document models to generate artificial document data sets with varying properties.

9.1 Why agglomerative hierarchical clustering performs poorly.

What distinguishes documents of different classes is the frequency with which words are used. In particular, each class typically has a "core" vocabulary of words that are used more frequently. For example, documents about finance will often talk about money, mortgages, trade, etc. while documents about sports talk about players, coaches, games, etc. These core vocabularies may overlap, documents may use more than one "core" vocabulary, and any particular document may contain words from these different "core" vocabularies, even if it does not belong to the class of documents that typically uses such words.

Each document has only a subset of all words from the complete vocabulary. Thus, because of the probabilistic nature of how words are distributed, any two documents may share many of the same words. Thus, we would expect that two documents can often be nearest neighbors without belonging to the same class. Figure 10 shows the percent of documents whose nearest neighbor is not of the same class. While this

Figure 10: Percent of Documents Whose Nearest Neighbor is of a Different Class



percentage varies widely from one data set to another, the chart does confirm what we have stated about nearest neighbor behavior in documents. As you consider the second, third, …,100th nearest neighbors, the percentage of points with a majority of non-nearest neighbors increases. (We mention that for small classes, this point is reached quite soon, just from the logic of the situation.)

Since, in many cases, the nearest neighbors of a document are of different classes, agglomerative hierarchal clustering will often put documents of the same class in the same cluster, even at the earliest stages of the clustering process. Because of the way that hierarchical clustering works, these "mistakes" cannot be fixed once they happen. A K-means refinement helps hierarchical clustering because, as we will see shortly, K-means can "overcome the problem of mixed nearest neighbors." The last two columns of Tables 3 – 9 show the benefit of a K-means refinement for hierarchical clustering very clearly.

In cases where nearest neighbors are unreliable, a different approach is needed that relies on more global properties. (This issue was discussed in a non-document context in [GRS99].) Since computing the cosine similarity of a document to a cluster centroid is the same as computing the average similarity of the document to all the cluster's documents, K-means is implicitly making use of such a "global property" approach. This explains why K-means does better vis-à-vis agglomerative hierarchical clustering, than is the case in other domains.

We stress that we are not saying that K-means or its variants are the "perfect" clustering algorithm for documents. There are a variety of issues with respect to K-means, e.g., initialization, and in practice, K-means sometimes fails to find clusters that correspond the desired document classes. However, the general approach of K-means seems better suited to documents than the approach of agglomerative hierarchical clustering.

9.2 Why bisecting K-means works better than regular K-means.

We believe that the main reason for this result is that bisecting K-means tends to produce clusters of relatively uniform size, while regular K-means is known to produce clusters of widely different sizes. Smaller clusters are often of higher quality, but this doesn't contribute much to the overall quality measure since quality measures weight each cluster's quality contribution by the cluster's size. Larger clusters, on the other hand, tend to be of lower quality and make a large negative contribution to cluster quality. We provide a simple numerical example to illustrate this for the case of entropy.

Consider a set of documents with two equal sized classes that are split into two clusters. If there is one cluster that contains few points and one cluster that contains almost all the points, then the entropy of the resulting solution is very nearly the entropy the larger cluster, which is just $-\frac{1}{2}\log_2 \frac{1}{2} - \frac{1}{2}\log_2 \frac{1}{2} = 1$. If the two clusters are equal size and the clustering algorithm manages to produce somewhat pure clusters, i.e., both clusters are more than 50% pure, then the entropy of each cluster is less than 1. The accompanying table shows the entropy for some values.

Percent Cluster Purity	Entropy
90	.47
80	.72
70	.88
60	.97
50	1.00

10 Conclusions

This paper presented the results of an experimental study of some common document clustering techniques. In particular, we compared the two main approaches to document clustering, agglomerative hierarchical clustering and K-means. For K-means we used a standard K-means and a variant of K-means, bisecting K-means. Our results indicate that the bisecting K-means technique is better than the standard K-means approach and as good or better than the hierarchical approaches that we tested. More specifically, the bisecting K-means approach produces significantly better clustering solutions quite consistently according to the entropy and overall similarity measures of cluster quality. Furthermore, bisecting K-means seems consistently to do slightly better at producing document hierarchies (as measured by the F measure) than the best of the hierarchical techniques, UPGMA. In addition, the run time of bisecting K-means is very attractive when compared to that of agglomerative hierarchical clustering techniques - $O(n)$ versus $O(n^2)$.

The reason that our relative ranking of K-means and hierarchical algorithms differs from those of other researchers could be due to many factors. First we used many runs of the regular K-means algorithm. If agglomerative hierarchical clustering techniques such as UPGMA are compared to a single run of K-means, then the comparison would be much more favorable for the hierarchical techniques. Secondly, we used incremental updating of centroids, which also improves K-means. Of course, we also used the bisecting K-means algorithm, which, to our knowledge, has not been previously used for document clustering. While there are many agglomerative hierarchical techniques that we did not try, we did try several other techniques which we did not report here. The results were similar– bisecting K-means performed as well or better then the hierarchical techniques that we tested. Finally, note that hierarchical clustering with a K-means refinement is essentially a hybrid hierarchical-K-means scheme similar to other such schemes that have been used before [CKPT92]. In addition, this scheme was better than any of the hierarchical techniques that we tried, which gives us additional confidence in the relatively good performance of bisecting K-means vis-à-vis hierarchical approaches.

We caution that the main point our paper is not a statement that bisecting K-means is "superior" to any possible variations of agglomerative hierarchical clustering or possible hybrid combinations with K-means. However, given the linear run-time performance of bisecting K-means and the consistently good quality of the clusterings that it produces, bisecting K-means is an excellent algorithm for clustering a large number of documents.

We argued that agglomerative hierarchical clustering does not do well because of the nature of documents, i.e., nearest neighbors of documents often belong to different classes. This causes agglomerative hierarchical clustering techniques to make mistakes that cannot be fixed by the hierarchical scheme. Both the K-means and the bisecting K-means algorithms rely on a more global approach, which effectively amounts to looking at the similarity of points in a cluster with respect to all other points in the cluster. This view also explains why a K-means refinement improves the entropy of a hierarchical clustering solution.

Finally, we put forward the idea that the better performance of bisecting K-means vis-à-vis regular K-means is due to fact that it produces relatively uniformly sized clusters instead of clusters of widely varying sizes.

References

[AGY99] Charu C. Aggarwal, Stephen C. Gates and Philip S. Yu, *On the merits of building categorization systems by supervised clustering,* Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Pages 352 – 356, 1999.

[APR97] Javed Aslam, Katya Pelekhov, and Daniela Rus, *A Practical Clustering Algorithm for Static and Dynamic Information Organization*, Proceedings of the 1998 ACM CIKM International Conference on Information and Knowledge Management, Bethesda, Maryland, USA, Pages 208-217, November 3-7, 1998.

[BF98] Paul Bradley and Usama Fayyad, *Refining Initial Points for K-Means Clustering*, Proceedings of the Fifteenth International Conference on Machine Learning ICML98, Pages 91-99. Morgan Kaufmann, San Francisco, 1998.

[BL85] Chris Buckley and Alan F. Lewit, *Optimizations of inverted vector searches*, SIGIR '85, Pages 97-110, 1985.

[CKPT92] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey, *Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections*, SIGIR '92, Pages 318 – 329, 1992.

[CCFW98] Moses Charikar, Chandra Chekuri, Tomas Feder, and Rajeev Motwani, *Incremental Clustering and Dynamic Information Retrieval*, STOC 1997, Pages 626-635, 1997.

[DJ88] Richard C. Dubes and Anil K. Jain, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[EW89] A. El-Hamdouchi and P. Willet, *Comparison of Hierarchic Agglomerative Clustering Methods for Document Retrieval*, The Computer Journal, Vol. 32, No. 3, 1989.

[GRS99] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim, (1998*), ROCK: A Robust Clustering Algorithm for Categorical Attributes*, In Proceedings of the 15th International Conference on Data Engineering, 1999.

[Han98] Eui-Hong (Sam) Han, Daniel Boley, Maria Gini, Robert Gross, Kyle Hastings, George Karypis, Vipin Kumar, B. Mobasher, and Jerry Moore, *WebAce: A Web Agent for Document Categorization and Exploration*. Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98).

[KS97] Daphe Koller and Mehran Sahami, *Hierarchically classifying documents using very few words*, Proceedings of the 14th International Conference on Machine Learning (ML), Nashville, Tennessee, July 1997, Pages 170-178.

[Kow97] Gerald Kowalski, *Information Retrieval Systems – Theory and Implementation*, Kluwer Academic Publishers, 1997.

[KR90] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: an Introduction to Cluster Analysis*, John Wiley and Sons, 1990.

[LA99] Bjorner Larsen and Chinatsu Aone, *Fast and Effective Text Mining Using Linear-time Document Clustering*, KDD-99, San Diego, California, 1999.

[reut] D. Lewis, Reuters-21578 text categorization text collection 1.0. http://www.research.att.com/~lewis

[Rij79] C. J. van Rijsbergen, (1989), *Information Retrieval*, Buttersworth, London, second edition.

[SS97] Hinrich Schutze and Craig Silverstein, *Projections for Efficient Document Clustering*, SIGIR '97, Philadelphia, PA, 1997.

[Sha48] Claude. E. Shannon, *A mathematical theory of communication*, Bell System Technical Journal*,* vol. 27, pp. 379-423 and 623-656, July and October, 1948.

[trec] TREC: Text REtrieval Conference. http://trec.nist.gov

[trecq] TREC: Text REtrieval Conference relevance judgments. http://trec.nist.gov/data/qrels_eng/index.html

[ZEMK97] Oren Zamir, Oren Etzioni, Omid Madani, Richard M. Karp, *Fast and Intuitive Clustering of Web Documents*, KDD '97, Pages 287-290, 1997.