HICAP: Hierarchical Clustering With Pattern Preservation

Hui Xiong [†] Michael Steinbach [†] Pang-Ning Tan [‡] Vipin Kumar [†]

Abstract

This paper describes a new approach for clustering—pattern preserving clustering—which produces more easily interpretable and usable clusters. This approach is motivated by the following observation: while there are usually strong patterns in the data—patterns that may be key for the analysis and description of the data—these patterns are often split among different clusters by current clustering approaches. This is, perhaps, not surprising, since clustering algorithms have no built in knowledge of these patterns and may often have goals that are in conflict with preserving patterns, e.g., minimize the distance of points to their nearest cluster centroids. Also, patterns are typically overlapping, i.e., may involve some of the same objects, and if the clustering algorithm produces disjoint clusters, then some patterns must be split when the objects are clustered. In this paper we describe a technique for pattern preserving clustering that first finds patterns composed of tightly connected groups of objects or attributes and then, starting from these patterns, performs agglomerative clustering using the Group Average (UPGMA) technique. We present the results of some experiments on document data that compare our approach, **HI**erarchical **C**lustering with P**A**ttern **P**reservation (HICAP), to two other clustering techniques: bisecting K-means and traditional UPGMA. These results show that, despite the extra constraint of pattern preservation, HICAP has performance very much like traditional UPGMA with respect to the cluster evaluation criteria of entropy and F-measure. More importantly, we also illustrate how patterns, if preserved, can aid cluster interpretation.

Keywords

Cluster Analysis, Hyperclique Pattern, Pattern Preserving Clustering

1 Introduction

Clustering and association analysis are important techniques for analyzing data. Cluster analysis [10] provides insight into the data by dividing the objects into groups (clusters) of objects, such that objects in a cluster are more similar to each other than to objects in other clusters. Association analysis [1], on the other hand, provides insight into the data by finding a large number of strong patterns—frequent itemsets and other patterns derived from them—in the data set. Frequent itemsets identify strongly connected sets of items (attributes) by finding attributes that occur together within a sufficiently large set of transactions. Thus, noting that clustering and association analysis can be performed either on objects or attributes, and restricting our discussion to binary transaction data, clustering and association analysis are both concerned with finding groups of *strongly related* objects or attributes, although at different levels. Association analysis finds strongly related objects or attributes on a local level, i.e., with respect to a subset of attributes or objects, while cluster analysis finds strongly related objects or attributes on a global level, i.e., by using all of the attributes or objects to compute similarity values.

More recently, we have defined a new pattern for association analysis—the *hyperclique pattern* [25]—that demonstrates a particularly strong connection between the overall similarity of a set of attributes (or objects) and the itemset (local pattern) in which they are involved. The hyperclique pattern is described in more detail later, but possesses the *high affinity property*: the attributes (objects) in a hyperclique pattern have a guaranteed level of global pairwise similarity to one another as measured by the cosine or Jaccard similarity measures [21]. Since clustering depends on similarity, it seems reasonable that the hyperclique pattern should have some connection to clustering, and thus, we have focused on answering the following two questions:

1. Can we use hyperclique patterns for clustering?
2. What happens to hyperclique patterns when data is clusterd by standard clustering techniques, e.g., how are they distributed among clusters?

The first question was investigated in the the original hyperclique paper [25], where an approach that used hyperclique patterns for clustering was presented. This paper will also pursue hyperclique based clustering, but using an approach that is motivated by the answer to the second question, which was investigated more re-

[†]Department of Computer Science and Engineering, University of Minnesota {huix, steinbac, kumar}@cs.umn.edu
[‡]Department of Computer Science and Engineering, Michigan State University ptan@cse.msu.edu

cently. We found that hypercliques are mostly destroyed by standard clustering techniques, i.e., standard clustering schemes do not preserve the hyperclique patterns, but rather, the objects or attributes comprising them are typically split among different clusters.

To understand why this is not desirable, consider a set of hyperclique patterns for documents. The high affinity property of hyperclique patterns requires that these documents must be similar to one another; the stronger the hyperclique, the more similar the documents. Thus, for strong hyperclique patterns, we would hope that any documents in the same pattern would end up in the same cluster in many or most cases. As mentioned, however, this is not what happens for traditional clustering algorithms.

There are two main reasons that hyperclique patterns are not preserved by clustering. First, clustering algorithms have no built in knowledge of these patterns and may often have goals that are in conflict with preserving patterns, e.g., minimize the distance of points from their closest cluster centroid. Second, many clustering techniques are non-overlapping, i.e., clusters cannot contain the same objects. (For hierarchical clustering, clusters on the same level cannot contain the same objects.) Since patterns are typically overlapping, clustering algorithms that produce disjoint clusters must split some patterns when the objects are clustered.

More generally, in many application domains, there are fundamental patterns that dominate the description and analysis of data within that area, e.g., in text mining, collections of words that form a topic, and in genomics, sequences of nucleotides that form a functional unit. If these patterns are not respected, then the value of a data analysis is greatly diminished for end users. In particular, if our interest is in patterns, such as hyperclique patterns, then we need a clustering approach that preserves these patterns, i.e., puts the objects or attributes of these patterns in the same cluster. Otherwise, the resulting clusters will be harder to understand since they must be interpreted solely in terms of objects instead of well-understood patterns.

There are two important considerations for developing a pattern persevering clustering approach. First, in any clustering scheme, we must take as our starting point the sets of objects or attributes that comprise the patterns of interest. If the objects or attributes of a pattern of interest are not together when we start the clustering process, they will often not be put together during clustering, since this is not the goal of the clustering algorithm. Second, if we start with the sets of objects or attributes that comprise the patterns of interest, we must not do anything in the clustering process to breakup these sets.

Hence, for pattern preserving clustering, the pattern must become the basic *object* of the clustering process. In theory, we could then use any clustering technique, although modifications would be needed to use sets of objects instead of objects as the basic starting point. However, agglomerative hierarchical clustering—described more fully later— has this property automatically, and thus, initially, we pursue an approach based on the Group Average (UPGMA) hierarchical clustering technique: **HI**erarchical **C**lustering with P**A**ttern **P**reservation (HICAP). Furthermore, we use hyperclique patterns as our patterns of interest, since they have some advantages with respect to frequent itemsets for use in pattern preserving clustering: hypercliques have the high affinity property, which guarantees that keeping objects together makes sense, and finding hypercliques is computationally much easier than finding frequent itemsets.

It is difficult to compare our approach to other clustering techniques, since a) HICAP preserves patterns and other clustering techniques do not, and b) HICAP produces overlapping clusters and most other clustering approaches do not. Also, as we briefly explain, both entropy and F-measure, which are common cluster evaluation methods for documents, have certain limitations that complicate comparisons among clustering techniques. Nonetheless, we present the results of some experiments on document data that compare our approach to two other clustering techniques, bisecting K-means and traditional UPGMA. These results show that, despite the extra constraint of pattern preservation, HICAP has performance very much like traditional UPGMA with respect to the cluster evaluation criteria of entropy and F-measure. Finally, we demonstrate how hyperclique patterns, if preserved, can help cluster interpretation.

Outline: Section 2 provides some background in clustering and also describes related work. Section 3 discusses the hyperclique pattern, while Section 4 presents the details of the HICAP algorithm. Results based on applying HICAP to three document data sets are given in Section 5, and are accompanied by an illustration of how patterns, if preserved, can aid cluster interpretation. Section 6 provides a brief conclusion and an indication of our plans for future work.

2 Clustering Background and Related Work

Cluster analysis has been the focus of considerable work, both within data mining and in other fields such as statistics, psychology, and pattern recognition. Several recent surveys may be found in [6, 9, 11], while more extensive discussions of clustering are provided by the following books [3, 10, 13]. The discussion in this section is, of necessity, quite limited.

While there are innumerable clustering algorithms, almost all of them can be classified as being either *partitional*, i.e., producing an un-nested set of clusters that partitions the objects in a data set into disjoint groups, or *hierarchical*, i.e., producing a nested sequence of partitions, with a single, all-inclusive cluster at the top and singleton clusters of individual points at the bottom. While this standard description of hierarchical versus partitional clustering assumes that each object belongs to a single cluster (a single cluster within one level, for hierarchical clustering), this requirement can be relaxed to allow clusters to overlap. Thus, in this paper we will describe algorithms as hierarchical or partitional and as overlapping or non-overlapping.*

Perhaps the best known and widely used partitional clustering technique is K-means [16], which aims to cluster a dataset into K clusters—K specified by the user—so as to minimize the sum of the squared distances of points from their closest cluster centroid. (A cluster centroid is the mean of the points in the cluster.) K-means is simple and computationally efficient, and a modification of it, bisecting K-means [20], can also be used for hierarchical clustering. Indeed, K-means is one of the best ways for generating a partitional or hierarchical clustering of documents [20, 26]. We use K-means, as implemented by CLUTO [12], as one of the methods to compare with our approach.

Traditional hierarchical clustering approaches [10] build a hierarchical clustering in an *agglomerative* manner by starting with individual points or objects as clusters, and then successively combining the two most similar clusters, where the similarity of two clusters can be defined in different ways and is what distinguishes one agglomerative hierarchical technique from another. These techniques have been used with good success for clustering documents and other types of data. In particular, the agglomerative clustering technique known as Group Average or UPGMA, which defines cluster similarity in terms of the average pairwise similarity between the objects in the two clusters, is widely used because it is more robust than many other agglomerative clustering approaches. Furthermore, a recent study [26] found UPGMA to be the best of the traditional agglomerative clustering techniques for clustering text. UPGMA is the basis of our HICAP clustering algorithm, and we use traditional UPGMA as another method to compare with our approach.

As far as we know, there are no other clustering methods based on the idea of preserving patterns. However, we mention two other types of clustering approaches that share some similarity with what we are doing here: constrained clustering and frequent item set based clustering. Constrained clustering [23] is based on the idea of using standard clustering approaches, but restricting the clustering process. Our approach can be viewed as constraining certain objects to stay together during the clustering process. However, our constraints are automatically enforced by starting with hyperclique patterns as our original clusters and then applying agglomerative hierarchical clustering, and thus, the general framework for constrained clustering is not necessary for our approach.

Our clustering technique is based on an association pattern, the hyperclique pattern, but there have been other clustering approaches that have used frequent itemsets or other patterns derived from them. One of these approaches is Frequent Itemset-based Hierarchical Clustering (FIHC) [5], which starts with clusters built around frequent itemsets. However, while our approach to clustering objects finds hypercliques of objects, and then uses them as initial clusters, FIHC uses selected itemsets—sets of binary attributes—to group objects (transactions), i.e., any object that supports an itemset goes into the same cluster. Thus, the approach of FIHC is quite different from that of HICAP, as is the approach of other clustering algorithms that are also based on frequent itemsets, e.g., [4, 17, 24]. More importantly, all the above pattern-based clustering approaches are not pattern preserving.

Finally, the hypergraph clustering approach [8] creates a hypergraph based on frequent itemsets and association rules, and then uses a hypergraph partitioning technique for finding clusters. Although hypergraph clustering inspired the clustering approach in the original hyperclique paper, this approach is not pattern preserving.

3 Basic Concepts of Association Patterns

The hyperclique pattern was the inspiration for pattern preserving clustering, and thus, the pattern that we use to explore this idea. In this section, we describe the concept of hyperclique patterns [25] after first introducing the concepts on which it is based: the frequent itemset and the association rule [1].

3.1 Frequent Itemsets and Association Rules

We quickly review some standard definitions related to association rule mining, which is an important technique for mining market basket data [1, 2].

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items. Let T be a set of transactions, where each transaction t is a set

*We admit that it is a bit strange to use the phrase, *overlapping partitional*, but in this case, we take *partitional* to mean *un-nested*.

of items $t \subseteq I$. An itemset is a set of items $X \subseteq I$, and the **support** of X, $supp(X)$, is the fraction of transactions containing X. If the support of X is above a user-specified minimum, i.e., $supp(X) \geq minsup$, then we say that X is a **frequent itemset**.

An association rule captures the fact that the presence of one set of items may imply the presence of another set of items, and is of the form $X \rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \phi$. The **confidence** of the rule $X \rightarrow Y$ is written as $conf(X \rightarrow Y)$ and is defined as $conf(X \rightarrow Y) = supp(X \cup Y)/supp(X)$, where $supp(X \cup Y)$ is the **support** of the rule. For example, suppose 70% of all transactions contain bread and milk, while 50% of the transactions contain bread, milk, and cookies. Then, the support of the rule {bread, milk} $\rightarrow$ {cookies} is 50% and its confidence is 50%/70% = 71%.

3.2 Hyperclique Patterns

A hyperclique pattern [25] is a new type of association pattern that contains items that are *highly affiliated* with each other. By high affiliation, we mean that the presence of an item in a transaction strongly implies the presence of every other item that belongs to the same hyperclique pattern. The h-confidence measure [25] is specifically designed to capture the strength of this association.

Definition 3.1. *The* **h-confidence** *of an itemset* $P = \{i_1, i_2, \cdots, i_m\}$, *denoted as* $hconf(P)$, *is a measure that reflects the overall affinity among items within the itemset. This measure is defined as* $min\{conf\{i_1 \rightarrow i_2, \ldots, i_m\}, conf\{i_2 \rightarrow i_1, i_3, \ldots, i_m\}, \ldots, conf\{i_m \rightarrow i_1, \ldots, i_{m-1}\}\}$, *where conf is the conventional definition of association rule confidence as given above.*

For instance, consider an itemset $P = \{A, B, C\}$. Assume that $supp(\{A\}) = 0.1$, $supp(\{B\}) = 0.1$, $supp(\{C\}) = 0.06$, and $supp(\{A, B, C\}) = 0.06$, where $supp$ is the support of an itemset. Then

$$\begin{aligned} conf\{A \rightarrow B, C\} &= supp(\{A, B, C\})/supp(\{A\}) = 0.6 \\ conf\{B \rightarrow A, C\} &= supp(\{A, B, C\})/supp(\{B\}) = 0.6 \\ conf\{C \rightarrow A, B\} &= supp(\{A, B, C\})/supp(\{C\}) = 1 \end{aligned}$$

Hence, $hconf(P) = min\{conf\{B \rightarrow A, C\}, conf\{A \rightarrow B, C\}, conf\ \{C \rightarrow A, B\}\} = 0.6$.

Definition 3.2. *Given a transaction database and the set of all items* $I = \{I_1, I_2, \ldots, I_n\}$, *an itemset* P *is a* **hyperclique pattern** *if and only if*

1. $P \subseteq I$ *and* $|P| > 0$.
2. $hconf(P) \geq h_c$, *where* h_c *is the minimum h-confidence threshold.*

Table 1: Examples of Hyperclique Patterns from words of the LA1 Data set

LA1 Dataset		
Hyperclique patterns	Support	H-confidence
{gorbachev, mikhail}	1.4%	93.6%
{photo, graphic, writer}	14.5%	42.1%
{sentence, convict, prison}	1.4%	32.4%
{rebound, score, basketball}	3.8%	40.2%
{season, team, game, play}	7.1%	31.4%

Table 1 shows some hyperclique patterns identified from words of the LA1 dataset, which is part of the TREC-5 collection [22] and includes articles from various news categories such as 'financial,' 'foreign,' 'metro,' 'sports,' and 'entertainment.' One hyperclique pattern in that table is {*mikhail*, *gorbachev*}, who is the ex-president of the former Soviet Union. Certainly, the presence of *mikhail* in one document strongly implies the presence of *gorbachev* in the same document and vice-versa.

Definition 3.3. *A hyperclique pattern is a* **maximal hyperclique pattern** *if no superset of this pattern is a hyperclique pattern.*

In this paper, we use maximal hyperclique patterns as the patterns that we wish to preserve.

3.3 Properties of the H-confidence measure

The h-confidence measure has three important properties, namely the anti-monotone property, the cross-support property, and the strong affinity property. Detailed descriptions of these three properties were provided in our earlier paper [25]. Here, we provide only the following brief summaries.

The anti-monotone property guarantees that if an itemset $\{i_1, \ldots, i_m\}$ has an h-confidence value of h_c, then every subset of size $m-1$ also has an h-confidence value of h_c. This property is analogous to the anti-monotone property of the support measure used in association-rule mining [1] and allows us to use h-confidence-based pruning to speed the search for hyperclique patterns in the same way that support-based pruning is used to speed the search for frequent itemsets.

The cross-support property provides an upper bound for the h-confidence of itemsets that contain items from different levels of support. The computation of this upper bound is much cheaper than the computation of the exact h-confidence value, since the it only relies on the support values of individual items in the itemset. Using this property, we can design a partitionbased approach that allows us to efficiently eliminate patterns involving items with different support levels.

The strong affinity property guarantees that if a hyperclique pattern has an h-confidence value above the minimum h-confidence threshold, h_c, then every pair of items within the hyperclique pattern must have a cosine similarity [19] greater than or equal to h_c. As a result, the overall affinity of hyperclique patterns can be controlled by setting an h-confidence threshold.

As demonstrated in our previous paper [25], the anti-monotone and cross-support properties form the basis of an efficient hyperclique mining algorithm that has much better performance than frequent itemset mining algorithms, particularly at low levels of support. Also, the number of hyperclique patterns is significantly less than the number of frequent itemsets.

4 HICAP: Hierarchical Clustering With Pattern Preservation

In this section, we present the details of the HICAP algorithm. We also discuss why hyperclique patterns are better than frequent itemsets for pattern preserving clustering.

4.1 Overview of HICAP

HICAP is based on the Group Average agglomerative hierarchical clustering technique, which is also known as UPGMA [10]. However, unlike the traditional version of UPGMA, which starts from clusters consisting of individual objects or attributes, HICAP uses hyperclique patterns to define the initial clusters, i.e., the objects or attributes of each hyperclique pattern become an initial cluster.

4.2 Algorithm Description

Figure 1 shows the pseudocode of the HICAP algorithm. This algorithm consists of two phases. In phase I, HICAP finds maximal hyperclique patterns, which are the patterns we want to preserve in the HICAP algorithm. We use only maximal hyperclique patterns since any non-maximal hyperclique will, during the clustering process, tend to be absorbed by its corresponding maximal hyperclique pattern and will, therefore, not affect the clustering process significantly. Thus, the use of non-maximal hyperclique would add complexity without providing any compensating benefits.

In phase II, HICAP conducts hierarchical clustering and outputs the clustering results. We highlight several

```
HICAP Algorithm
Input:      $D$: a document data set.
            $\theta$: a minimum h-confidence threshold.
            $\alpha$: a minimum support threshold.
Output:     CR: the hierarchical clustering result.
Variables:  S: the hyperclique pattern set.
            MS: the maximal hyperclique pattern set.
            PD: The output set of preprocessing
            LS: a set of objects which are not covered by
                identified maximal hyperclique patterns
            CS: a set containing target clustering objects
Method
Phase I: Maximum Hyperclique Dattern Discovery
 1.         S = hyperclique_miner($\theta$, $\alpha$, D)
 2.         MS = maximal_hyperclique_pattern(S)

Phase II: Hierarchical Clustering
 3.         PD = preprocessing(D)
 4.         LS = uncovered_objects(MS, D)
 5.         CS = LS $\cup$ MS
 6.         for i=1 to $|CS|$-1
 7.             find the pair of elements with max group
                average cosine value from the set CS,
 8.             merge the identified pair, and update
                CS and CR accordingly
 9.         endfor
10.         OUTPUT CR
11.         End
```

Figure 1: The HICAP Algorithm

important points. First, since hyperclique patterns can be overlapping, some of the resulting clusters will be overlapping. Second, identified maximal hyperclique patterns typically cover only 10 to 20% of all objects, and thus, HICAP also includes each uncovered object as a separate initial cluster, i.e., the hierarchial clustering starts with maximal hyperclique patterns and uncovered objects. Finally, the similarity between clusters is calculated using the average of the pairwise similarities between objects, where the similarity between objects is computed using the cosine measure.

Hyperclique Patterns Vs. Frequent Itemsets As mentioned, there has been some prior work that uses frequent itemsets as the basis for clustering algorithms [4, 5, 24]. While the goals and methods of that work are different from our own, we could have used frequent itemsets instead of hypercliques in HICAP. We chose hypercliques because we feel that they have several advantages for pattern preserving clustering. First, since hyperclique patterns are strong affinity patterns, they include objects which are strongly similar to each other with respect to the cosine measure and which should, therefore, naturally be within the same cluster. In contrast, many pairs of objects from a frequent itemset may have very poor similarity with respect to the cosine measure. Second, the hyperclique pattern mining algorithm has much better performance at low levels of support than frequent itemset mining algorithms. Thus, capturing patterns occurring at low levels of support is much easier for hyperclique patterns than frequent itemsets. Finally, the size of maximal hyperclique patterns is significantly smaller than the size of maximal frequent itemsets. In summary, a version of HICAP that uses hypercliques is far more computationally efficient that a version of HICAP that uses frequent patterns, and is more likely to produce meaningful clusters.

5 Experimental Evaluation

In this section, we present an experimental evaluation of HICAP. After a brief description of our document data sets and cluster evaluation measures, we first elaborate on why the hyperclique pattern is a good candidate for pattern preserving clustering. We then illustrate the poor behavior of traditional clustering approaches in terms of pattern preservation, and show how hyperclique patterns can be used to interpret the clustering results produced by HICAP. Finally, we evaluate the clustering performance of HICAP, UPGMA, and K-means with respect to the F-measure and entropy, including in our evaluation, a brief discussion on the limitations of each of these evaluation measures.

Experimental Data Sets. For our experiments, we used three real data sets—from several different application domains—that are widely used in document clustering research. Some characteristics of these data sets are shown in Table 2.

Table 2: Characteristics of document data sets.

Data Set	LA1	RE0	WAP
Number of Documents	3204	1504	1560
Number of Words	31472	11465	8460
Number of Classes	6	13	20
Min Class Size	273	11	5
Max Class Size	943	608	341
Min Class Size/Max Class Size	0.29	0.018	0.015
Source	TREC-5	Reuters	WebAce

The `LA1` data set is part of the TREC-5 collection [22] and contains news articles from the Los Angeles Times. The `RE0` data set is from the Reuters-21578 text categorization test collection Distribution 1.0 [15]. The data set `WAP` is from the WebACE project (WAP) [7]; each document corresponds to a web page listed in the subject hierarchy of Yahoo!. For all data sets, we used a stop-list to remove common words, and the words were stemmed using Porter's suffix-stripping algorithm [18].

Evaluation Methods. To evaluate the quality of the clusters produced by the different clustering techniques, we employed two commonly used measures of clustering quality: entropy and the F-measure [14]. Both entropy and the F-measure are 'external' criteria, i.e., they use external information—class labels in this case. Entropy measures the purity of the clusters with respect to the given class labels. Thus, if all clusters consist of objects with only a single class label, the entropy is 0. However, as the class labels of objects in a cluster become more varied, the entropy increases. The F-measure also measures cluster quality, but attains its maximum value when each class is contained in a single cluster, i.e., clusters are pure and contain all the objects of a given class. The F-measure declines as we depart from this 'ideal' situation. Formal definitions of entropy and the F-measure are given below.

Entropy. To compute the entropy of a set of clusters, we first calculate the class distribution of the objects in each cluster, i.e., for each cluster j we compute p_{ij}, the probability that a member of cluster j belongs to class i. Given this class distribution, the entropy, E_j, of cluster j is calculated using the standard entropy formula

$$E_j = -\sum_i p_{ij} log(p_{ij}) \tag{5.1}$$

where the sum is taken over all classes and the log is log base 2. The total entropy for a set of clusters is computed as the weighted sum of the entropies of each cluster, as shown in the equation

$$E = \sum_{j=1}^{m} \frac{n_j}{n} * E_j \tag{5.2}$$

where n_j is the size of cluster j, m is the number of clusters, and n is the total number of data points.

F-measure. The F-measure combines the precision and recall concepts from information retrieval [19]. 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 as follows:

$$Recall(i,j) = n_{ij}/n_i \tag{5.3}$$

$$Precision(i,j) = n_{ij}/n_j \tag{5.4}$$

where n_{ij} is the number of objects of class i that are in cluster j, n_j is the number of objects in cluster j, and n_i is the number of objects in class i. The F-measure of cluster j and class i is then give by the equation

$$F(i,j) = \frac{2 * Recall(i,j) * Precision(i,j)}{Precision(i,j) + Recall(i,j)} \tag{5.5}$$

For an entire hierarchical clustering, the F-measure of any class is the maximum value it attains at any node (cluster) in the tree, and an overall value for the F-measure [14] is computed by taking the weighted average the F-measures for each class, as given by the equation

$$F = \sum_i \frac{n_i}{n} max\{F(i,j)\} \tag{5.6}$$

where the max is taken over all clusters at all levels.

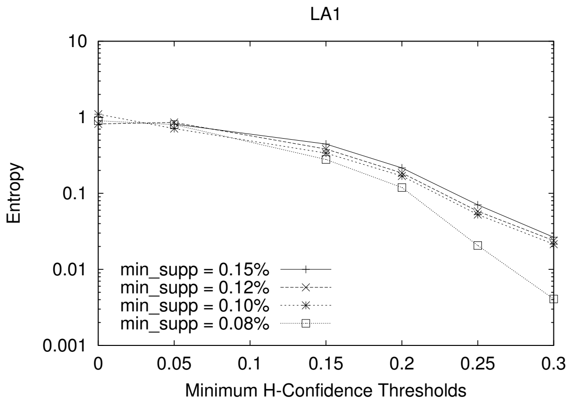

Figure 2: Illustration of the high-affinity property of hyperclique patterns on the LA1 data set.

5.1 High-Affinity Hyperclique Patterns

In this section, we present the results of an experiment that illustrates why the hyperclique pattern is a good pattern to use for pattern preserving clustering. Figure 2 shows, for the LA1 data set, the entropy of the discovered hyperclique patterns for different minimum h-confidence and support thresholds. Note that when the minimum h-confidence threshold is zero, we actually have frequent itemset patterns instead of hyperclique patterns. As Figure 2 shows, as the minimum h-confidence threshold increases, the entropy of hyperclique patterns decreases dramatically. For instance, when the h-confidence threshold is higher than 0.25, the entropy of hyperclique patterns is less than 0.1 for all the given minimum support thresholds. This indicates that hyperclique patterns are very 'pure' patterns for certain h-confidence thresholds. In other words, a hyperclique pattern includes objects which are naturally from the same class category. In contrast, the entropy of frequent patterns is high—close to 1—for all the given minimum support thresholds. This means that frequent patterns include objects from different class categories. Thus, with respect to purity, the hyperclique pattern is a better candidate than frequent patterns for pattern preserving clustering.

Another trend that we can observe in Figure 2 is that, as the minimum support threshold decreases, the entropy of hyperclique patterns from the LA1 data set trends downward. This indicates that high affinity patterns can appear at very low levels of support. As mentioned, frequent itemset mining algorithms have difficulty identifying frequent itemsets at low levels of support. In contrast, the hyperclique pattern mining algorithm has much better performance at low levels of support [25]. Hence, if we want to discover the high-affinity patterns occurring at low levels of support, the hyperclique pattern is a better choice than frequent itemset patterns.

5.2 Preserving Patterns

By design, HICAP preserves all hyperclique patterns throughout the clustering process. However, as we show in this experiment, traditional clustering algorithms—UPGMA and bisecting K-means—tend to break hyperclique patterns. Figure 3 shows, for different number of clusters, the ratio of hyperclique patterns being split by the UPGMA and bisecting K-means algorithms. For every data set, the minimum number of clusters is specified as the original number of classes in that data set. In the figure, we observe that the ratio of patterns being split for both algorithms increases as the number of clusters increases. Furthermore, even when the number of clusters equals the number of classes, UPGMA and bisecting K-means still break patterns. Finally, bisecting K-means breaks more patterns than UPGMA, because its preference for relatively uniform cluster sizes tends to break long hyperclique patterns.

5.3 Interpretation of Clusters Using Hyperclique Patterns

In this experiment, we provide two types of evidence to illustrate the usefulness of patterns for interpreting clustering results: specific examples and an analysis of the clusters on one level of the cluster hierarchy. For the first approach, we picked two clusters at random from the hierarchical clustering generated

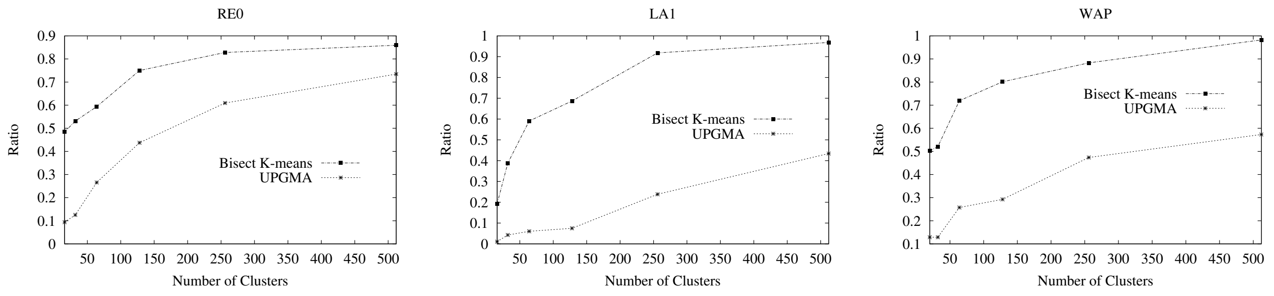

Figure 3: The ratio of pattern being split by UPGMA and bisecting k-means.

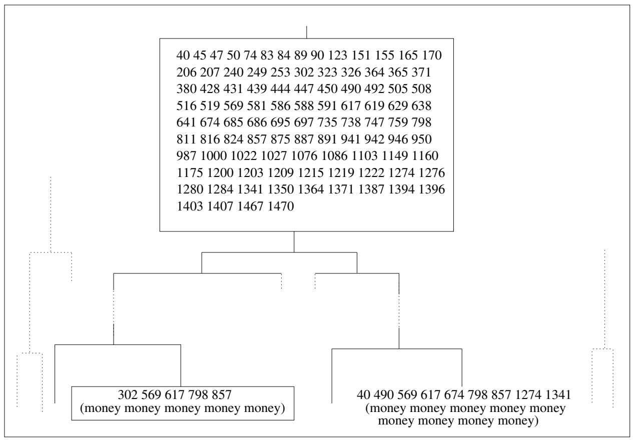

Figure 4: Cluster Interpretation I

Table 3: Classes of Documents in an Example Cluster from the RE0 data set.

money	money	money	money	money	money	money	money
money	money	money	money	money	money	money	money
money	money	money	money	money	money	money	money
money	money	money	money	money	money	money	money
money	money	money	money	money	money	money	money
money	money	money	money	interest	money	money	interest
money	money	money	money	money	money	money	money
money	money	money	money	money	money	money	money
money	money	money	money	money	money	money	money
money	money	money	money	money	money	money	money
money	money	money	money	money	money	money	money
money	money	money	money	money	money	money	money
money	money	money	money	money			

by HICAP, and then looked at the hyperclique patterns that they contained to see if the nature of these hypercliques, which include only a fraction of the documents or words in the cluster, are useful for understanding the nature of the cluster. As we show below, this was the indeed the case.

Cluster Contains Hypercliques of the Same Class Figure 4 shows a cluster randomly selected from the HICAP clustering results on the RE0 data set. One cluster with document IDs is presented. On further analysis, we found that two hyperclique patterns are in this cluster, and, as shown in the figure, both hyperclique patterns belong to the 'money' category. Since HICAP is based on the Group Average agglomerative clustering approach, it is natural to expect that other documents in the given cluster should have a significant level of similarity to the documents in two hyperclique patterns. In other words, if these two hyperclique patterns act as a 'kernel,' the documents merged into this cluster are likely to have the same class label as the documents in these two hyperclique patterns. As a result, we might expect that a large population of documents in this cluster would have the class, 'money.'

To verify this, we show the class labels of the cluster objects in Table 3. As suggested by the two hyperclique patterns, all of the documents, except two, belong to the class, 'money.'

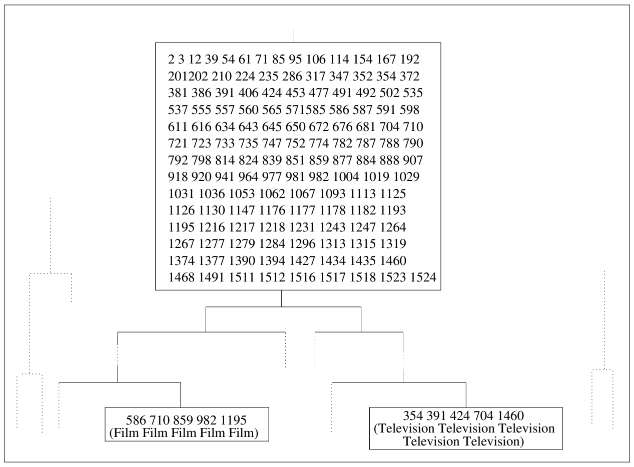

Figure 5: Cluster Interpretation II

Cluster Contains Hypercliques of Different Classes Figure 5 shows another cluster randomly picked from the HICAP clustering of the WAP data set. This cluster contains two hyperclique patterns with documents from two different categories: 'Film' and 'Television.' As a result, we would expect that this cluster should be a hybrid cluster with documents mainly from two categories: 'Film' and 'Television.' Table 4

Table 4: Classes of Documents in an Example Cluster from the WAP data set.

Television Television Television Film Film Television Stage Television Television Television Film Cable Television Television Television Variety Film Television Film Television Stage Television Film Television Film Television Film Television Film Television Television Film Cable Television Stage Film Television People Television Film People Television Cable Film Television Television Media Stage Television Film Television Film Television Television Stage Film Television Film Television Television Stage Film Television Film Television Television Film Film Television Television Cable Television Television People Television Film Television Film Television Television Film Television Variety Variety Television Film Film Cable Film Television Television Film Television Television Film Television Television Television Film Film Television Film Film Television Television Television Film Television Television Television Film Television Film Television Film Television Film Television Film Variety Film Television Film Industry Television Film Television Art Television Television Film Media Industry Stage Television Television Television Television Television

shows the class labels of the cluster objects in this cluster. Once again, the interpretation based on hyperclique patterns matches the classes found in the cluster.

Table 5: Statistics of interpretable clusters.

CNo	size	#unmatch	#hyperclique	Classes of hypercliques
1	49	5	16	People/Online
2	66	0	9	Sports
3	169	59	10	Business/Tech/Politics
4	313	2	49	Health
5	33	3	4	Film
6	61	9	9	Politics
7	18	0	7	Culture
8	44	2	1	Television
9	25	0	7	Sports
10	22	4	1	People
11	8	0	2	Television/Stage
Total	808	84	115	

Analyzing Clusters on one Level of the Cluster Hierarchy To further validate the hypothesis that the nature of the hyperclique patterns contained in a cluster tells us something about the nature of the cluster, we decided to look at the clusters on one level of the cluster hierarchy. We first identified the class of each of the hyperclique patterns—there were 115 of these patterns in the WAP data set, which together covered 265 out of 1560 documents. Finding the class of each hyperclique was an easy task since the hypercliques almost always consisted of objects of a single class, and if not, were predominantly of one class. Then, we found which of the 128 clusters contained hypercliques—there were 11 such clusters, which covered 808 of the 1560 documents (The skewed distribution of cluster sizes is a result of the skewed distribution of class sizes.). We further analyzed each cluster with respect to the classes of documents that it contained and whether the classes of the documents in the cluster matched the classes of the hypercliques in the cluster. The results of this analysis are contained in Table 5. ('CNo' is cluster number, 'size' is the number of objects in the cluster, '#unmatch' is the number of objects in the cluster that do not match a class of the hypercliques in the cluster, '# hyperclique' is the number of hypercliques in the cluster, and 'Classes of hypercliques' is the classes of the hypercliques.)

The results confirm the observations suggested by the previous two examples. If the hypercliques in a cluster are of one class, then the objects in that cluster are predominantly of the same class. On the other hand, if the hypercliques in a cluster are of mixed classes, then the objects in the cluster are also of mixed class, although they tend to be very heavily composed of the classes of the hypercliques. The worst case in the table is cluster 3, which has hyperclique patterns from three classes, and has 59 out of 169 documents that are not of these three classes. The documents in the other clusters almost always match the labels of the corresponding hyperclique patterns.

5.4 Clustering Evaluation using F-measure

In a previous study, [20] we compared bisecting K-means and UPGMA with respect to the (hierarchical) F-measure. These results are shown in Table 6. The F-score for bisecting K-means is slightly higher than that for UPGMA, a result that was also confirmed in a later study [26].

Table 6: F-Score for LA1, RE0, and WAP

Data Set	Bisecting K-means	UPGMA
re0	0.5863	0.5859
wap	0.6750	0.6434
la1	0.7856	0.6963

To see if we could find more of a distinction between techniques, we applied the F-measure only to the resulting clusters, not the whole hierarchy. (We later discuss the merits or demerits of doing this, and the limitations of the F-measure.) Figure 6 shows the F-measure values of the clustering results from HICAP, UPGMA, and bisecting K-means. In terms of the F-measure, HICAP significantly outperforms bisecting K-means in most cases for the three given data sets. Also, as the number of clusters increases, bisecting K-means tends to have even worse F-measure values since the given data sets have classes with widely different class sizes as is shown in Table 2. For instance, the minimum class size for the WAP data set is 5 and the maximum class size is 341. Because bisecting K-means splits the clusters with low average similarity, it tends to produce clusters of relatively uniform size, and hence, it is difficult for bisecting K-means to produce a cluster that

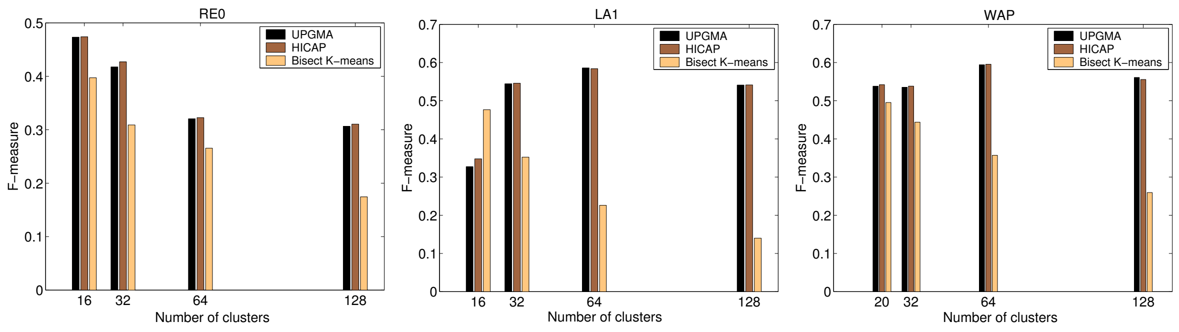

Figure 6: F-measure Comparison on the LA1, RE0, and WAP data sets.

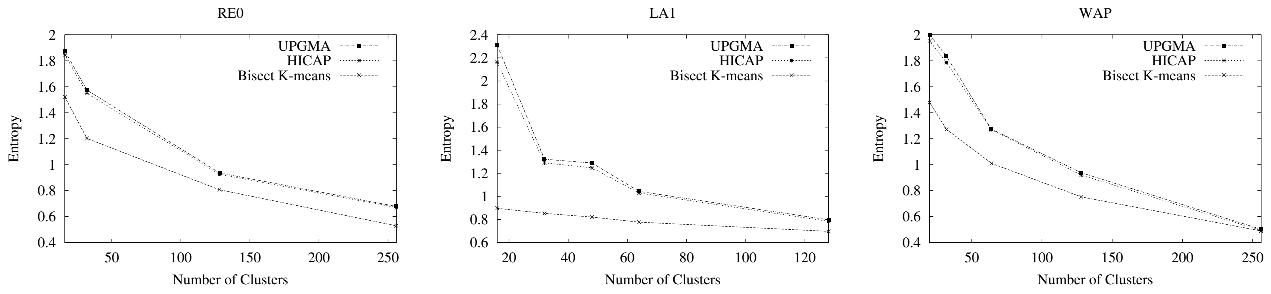

Figure 7: Entropy Comparison on the LA1, RE0, and WAP data sets.

accurately represents one of the larger classes when the data set has classes of widely varying sizes. However, we also observe that bisecting K-means has a better F-measure value on the LA1 data set when the number of clusters is 16. A possible reason for this is that the class distribution of the LA1 data set is not very skewed. As can be see in Table 2, the ratio of minimum class size and maximum class size is 0.29, which is much higher than the ratios for the RE0 and WAP data sets.

Another observation involving Figure 6 is that HICAP performs slightly better than UPGMA in most cases for the given three data sets. It is not surprising to observe this, since the difference between these two approaches is that HICAP starts from patterns, while UPGMA starts from individual objects. Thus, as compared to UPGMA, HICAP may reduce early stage merging errors, since it considers the affinity among all items within a pattern instead of only considering pair-wise similarity. However, the performance difference between HICAP and UPGMA is small, probably because the hyperclique patterns cover only 10-20% of the documents in a data set.

The Limitations of the F-measure The F-measure emphasizes both precision, i.e., does the cluster contain only objects of the same class, and recall, i.e., does the cluster contain all the objects of a class. Thus, the F-measure would seem to penalize methods, such as bisecting K-means that tend to split the objects in a class into many (almost equal size) subclusters, which would each have poor recall (class coverage) and thus, a poor F-measure. Bisecting K-means does so well on the F-measure defined over the entire hierarchy because it produces relatively large and pure clusters at the early levels of the splitting. However, as the above results also show, the F-measure evaluated on a single level gives better values for UPGMA versus bisecting K-means on a wide range of levels. This is because UPGMA has clusters that reflect the true classes on a much wider range of levels of the cluster hierarchy. Thus, a straight comparison of values of the F-measure does not capture the full story.

5.5 Clustering Evaluation using Entropy

Figure 7 shows the entropy values of the clustering results from HICAP, UPGMA, and bisecting K-means at different user-specified numbers of clusters. Bisecting K-means yields significantly better entropy values than HICAP and UPGMA for all three data sets since the entropy measure favors clustering algorithms, such as bisecting K-means, that produce clusters which have relatively uniform cluster size. (Further discussion of the limitations of entropy is provided later.) Also, for all three clustering algorithms, entropy values tend to decrease as the number of cluster increases since the resulting clusters tend to be more pure. Thus, the difference in entropy among the three algorithms decreases as the number of clusters increases.

Another observation from Figure 7 is that HICAP performs slightly better than UPGMA in most cases for the given data sets, although this performance difference is small. It is not surprising to observe this since UPGMA starts from individual objects, while HICAP starts from hyperclique patterns (and the uncovered objects).

Table 7: An Example Document Data Set

Target document data set
Sports, Sports, Sports, Sports, Sports, Sports, Sports, Sports, Sports, Sports, Sports, Sports, Sports, Entertainment, Entertainment, Entertainment, Entertainment, Foreign, Foreign, Foreign, Foreign, Metro, Metro, Metro, Metro, Metro, Metro, Metro, Metro, Metro, Metro, Metro, Politics, Politics, Politics

Table 8: Two Clustering Results.

Document Clustering		Entropy	F-measure
Clustering I	1: Sports Sports Sports Sports	0.153	0.663
	2: Sports Sports Sports Sports		
	3: Sports Sports Sports		
	4: Sports Sports Entertainment		
	5: Entertainment Entertainment Entertainment Entertainment		
	6: Foreign Foreign Foreign		
	7: Metro Metro Metro Metro		
	8: Metro Metro Metro Metro Metro		
	9: Metro Metro Foreign		
	10: Politics Politics Politics		
Clustering II	1: Sports	0.272	0.816
	2: Sports		
	3: Sports Sports Sports Sports Sports Sports Sports Sports Sports Sports Sports Entertainment		
	4: Entertainment Entertainment		
	5: Entertainment Entertainment		
	6: Foreign Foreign		
	7: Foreign Foreign		
	8: Metro		
	9: Metro Metro Metro Metro Metro Metro Metro Metro Metro Metro Politics		
	10: Politics Politics		

The Limitations of Entropy We observed that the entropy measure tends to favor clustering algorithms, such as bisecting K-means, which produce clusters with relatively uniform size. To illustrate this, we created the example document data set shown in Table 7. This data set consists of 18 documents with class labels. Two clustering results are shown in Table 8. The entropy of clustering result II is almost twice as large as the entropy of clustering result I. (The F-values give an opposite evaluation of these clusters.) Certainly, according to entropy, clustering result I is much better than clustering result II. However, this result is due to the fact that the entropy measure more heavily penalizes a large impure cluster, such as cluster 9. Consequently, for clustering algorithms that tend to produce clusters of widely different sizes, e.g., UPGMA and HICAP, entropy may not be a good measure.

6 Conclusion

In this paper, we have introduced a new goal for clustering algorithms, namely, the preservation of patterns, such as the hyperclique pattern, that capture strong connections between groups of objects. Without such an explicit goal, clustering algorithms tend to find clusters that split the objects or attributes in these patterns between different clusters. However, keeping these patterns together aids cluster interpretation.

Agglomerative hierarchical clustering is naturally pattern preserving if it takes the objects or attributes of the patterns of interest as the starting clusters. Thus, we introduced a new clustering approach, **HI**erarchical **C**lustering with P**A**ttern **P**reservation (HICAP), which is based on the Group Average (UPGMA) agglomerative clustering technique, and which uses maximal hyperclique patterns to define the initial clusters. We used hyperclique patterns for pattern preserving clustering because they have the high-affinity property, which guarantees that the objects or attributes in a hyperclique pattern are highly similar. Thus, it makes sense to put these objects or attributes in the same cluster. In contrast, frequent itemsets do not have the high-affinity property and are more expensive to find.

While HICAP, by construction, preserves patterns, we showed experimentally that K-means and the standard UPGMA clustering approaches tend to break patterns in many or most cases. We also provided some experimental results that showed that the entropy and F-measure values of clusters produced by HICAP are similar to those of clusters produced by the standard UPGMA approach. Thus, apparently it did not cost us anything to use maximal hyperclique patterns to define our starting clusters, thereby ensuring that these patterns are preserved. Finally, we demonstrated how hyperclique patterns, if preserved, can be used for interpreting clustering results.

While this paper showed, for document data sets, the potential usefulness of pattern preserving clustering based on a hierarchical clustering approach, more work is necessary to further expand the experimental results that we provided and to show the broad applicability and usefulness of a pattern preserving approach to clustering. We plan to apply HICAP to additional document data sets and to other types of data, e.g., genomics and spatial data. We also hope to investigate how pattern preserving clustering might be incorporated into non-hierarchical clustering schemes.

Finally, while HICAP can be made more general by using patterns other than hyperclique patterns, we need to investigate what conditions these patterns must meet for this to be meaningful. In particular, do patterns need to possess the high affinity property? More generally, we hope to understand what must be done to modify clustering techniques to be compatible with the goal of preserving patterns.

7 Acknowledgements

This work was partially supported by NASA grant # NCC 2 1231, NSF grant # IIS-0308264 and by Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAD19-01-2-0014. The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. Access to computing facilities was provided by the AHPCRC and the Minnesota Supercomputing Institute.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD*, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB*, 1994.

[3] M. R. Anderberg. *Cluster Analysis for Applications.* Academic Press, New York, December 1973.

[4] F. Beil, M. Ester, and X. Xu. Frequent term-based text clustering. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 436–442. ACM Press, 2002.

[5] M. E. Benjamin C.M. Fung, Ke Wang. Large hierarchical document clustering using frequent itemsets. In *Proc. of SDM.* SIAM, 2003.

[6] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, CA, 2002.

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

[8] E.-H. S. 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.

[9] J. Han, M. Kamber, and A. K. H. Tung. Spatial clustering methods in data mining: A review. In H. J. Miller and J. Han, editors, *Geographic Data Mining and Knowledge Discovery*, pages 188–217. Taylor and Francis, London, December 2001.

[10] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall Advanced Reference Series. Prentice Hall, Englewood Cliffs, New Jersey, March 1988.

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

[12] G. Karypis. Cluto: Software for clustering high-dimensional datasets. /www.cs.umn.edu/∼karypis.

[13] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley Series in Probability and Statistics. John Wiley and Sons, New York, Novemeber 1990.

[14] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Prof. of ACM SIGKDD*, 1999.

[15] D. Lewis. Reuters-21578 text categorization text collection 1.0. In *http://www.research.att.com/ lewis.*

[16] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. L. Cam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume I, Statistics.* University of California Press, September 1999.

[17] J. Pei, X. Zhang, M. Cho, H. Wang, and P. Yu. Maple: A fast algorithm for maximal pattern-based clustering. In *Proc. of ICDM*, 2003.

[18] M. F. Porter. An algorithm for suffix stripping. In *Program, 14(3)*, 1980.

[19] C. J. V. Rijsbergen. *Information Retrieval (2nd Edition).* Butterworths, London, 1979.

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

[21] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proc. of AAAI: Workshop of Artificial Intelligence for Web Search*, pages 58–64. AAAI, July 2000.

[22] TREC. In *http://trec.nist.gov.*

[23] A. K. H. Tung, R. T. Ng, L. V. S. Lakshmanan, and J. Han. Constraint-based clustering in large databases. In J. V. den Bussche and V. Vianu, editors, *Database Theory - ICDT 2001*, 2001.

[24] K. Wang, C. Xu, and B. Liu. Clustering transactions using large items. In *Proc. of ACM CIKM*, 1999.

[25] H. Xiong, P. Tan, and V. Kumar. Mining strong affinity association patterns in data sets with skewed support distribution. In *Proc. of the 3rd IEEE International Conf. on Data Mining*, pages 387–394, 2003.

[26] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proc.of ACM CIKM*, pages 515–524. ACM Press, 2002.