Decision Support Systems, 1999, to appear

Partitioning-Based Clustering for Web Document Categorization [*]

Daniel Boley, Maria Gini, Robert Gross, Eui-Hong (Sam) Han, Kyle Hastings,
George Karypis, Vipin Kumar, Bamshad Mobasher, and Jerome Moore
Department of Computer Science and Engineering
University of Minnesota, Minneapolis, MN 55455

Abstract

Clustering techniques have been used by many intelligent software agents in order to retrieve, filter, and categorize documents available on the World Wide Web. Clustering is also useful in extracting salient features of related web documents to automatically formulate queries and search for other similar documents on the Web. Traditional clustering algorithms either use *a priori* knowledge of document structures to define a distance or similarity among these documents, or use probabilistic techniques such as Bayesian classification. Many of these traditional algorithms, however, falter when the dimensionality of the feature space becomes high relative to the size of the document space. In this paper, we introduce two new clustering algorithms that can effectively cluster documents, even in the presence of a very high dimensional feature space. These clustering techniques, which are based on generalizations of graph partitioning, do not require pre-specified ad hoc distance functions, and are capable of automatically discovering document similarities or associations. We conduct several experiments on real Web data using various feature selection heuristics, and compare our clustering schemes to standard distance-based techniques, such as *hierarchical agglomeration clustering*, and Bayesian classification methods, such as *AutoClass*.

Keywords: clustering, categorization, World Wide Web documents, graph partitioning, association rules, principal component analysis.

[*]This work was supported in part by Army Research Office contract DA/DAAG55-98-1-0441, by Army High Performance Computing Research Center cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Additional support was provided by the IBM Partnership Award, and by the IBM SUR equipment grant. Access to computing facilities was provided by Minnesota Supercomputer Institute.

1 Introduction

The World Wide Web is a vast resource of information and services that continues to grow rapidly. Powerful search engines have been developed to aid in locating unfamiliar documents by category, contents, or subject. Unfortunately, queries often return inconsistent results, with document referrals that meet the search criteria but are of no interest to the user.

While it may not be currently feasible to extract in full the meaning of an HTML document, intelligent software agents have been developed which extract features from the words or structure of an HTML document and employ them to classify and categorize the documents. Clustering offers the advantage that *a priori* knowledge of categories is not needed, so the categorization process is unsupervised. The results of clustering could then be used to automatically formulate queries and search for other similar documents on the Web, or to organize bookmark files, or to construct a user profile.

In this paper, we present two new clustering algorithms based on graph partitioning and compare their performance against more traditional clustering algorithms used in information retrieval.

Traditional clustering algorithms either define a distance or similarity among documents, or use probabilistic techniques such as Bayesian classification. Many of these algorithms, however, break down as the size of the document space, and hence, the dimensionality of the corresponding feature space increases. High dimensionality is characteristic of the information retrieval applications which are used to filter and categorize documents on the World Wide Web. In contrast, our partitioning-based algorithms perform well in the presence of a high dimensional space.

In section 2 we describe the clustering algorithms, in Section 3 we present results of a number of experiments using different methods to select features from the documents, and we compare the results of the different clustering algorithms. We show that partitioning clustering methods perform better than traditional distance based clustering. Finally in Section 3 we compare our work with other similar systems and present ideas for future research.

2 Clustering Methods

Most of the existing methods for document clustering are based on either probabilistic methods, or distance and similarity measures (see [15]). Distance-based methods such as k-means analysis, hierarchical clustering [20] and nearest-neighbor clustering [23] use a selected set of words appearing in different documents as features. Each document is represented by a feature vector, and can be viewed as a point in a multi-dimensional space.

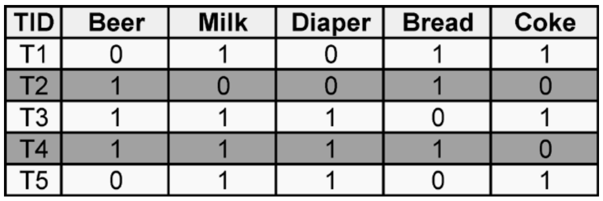
There are a number of problems with clustering in a multi-dimensional space using traditional distance- or probability-based methods. First, it is not trivial to define a distance measure in this space. Feature vectors must be scaled to avoid skewing the result by different document lengths or possibly by how common a word is across many documents. Techniques such as TFIDF [28] have been proposed precisely to deal with some of these problems, but we have found out in our experiments that using TFIDF scaling does not always help.

Second, the number of different words in the documents can be very large. Distance-based schemes generally require the calculation of the mean of document clusters, which are often chosen initially at random. In a high dimensional space, the cluster means of randomly chosen clusters will do a poor job at separating documents. Similarly, probabilistic methods such as Bayesian classification used in *AutoClass* [11, 29] do not perform well when the size of the feature space is much larger than the size of the sample set or may depend on the independence of the underlying features. Web documents suffer from both high dimensionality and high correlation among the feature values. We have found that *hierarchical agglomeration clustering* (HAC) [13] is computationally very expensive, and AutoClass has performed poorly on our examples.

Our proposed clustering algorithms, described below, are designed to efficiently handle very high dimensional spaces and large data sets, as shown in the experimental results we describe later.

2.1 Association Rule Hypergraph Partitioning (ARHP)

The Association Rule Hypergraph Partitioning (ARHP) [17, 18] is a clustering method based on the *association rule* discovery technique used in data mining. This technique is often used to discover affinities among items in a transactional database (for example, to find sales relationships among items sold in supermarket customer transactions. From a database perspective, these transactions can be viewed as a relational table in which each item represents an attribute and the domain of each attribute is either the binary domain (indicating whether the item was bought in a particular transaction) or a non-negative integer indicating the frequency of purchase within a given transaction. Figure 1 depicts a portion of a typical supermarket transaction database.



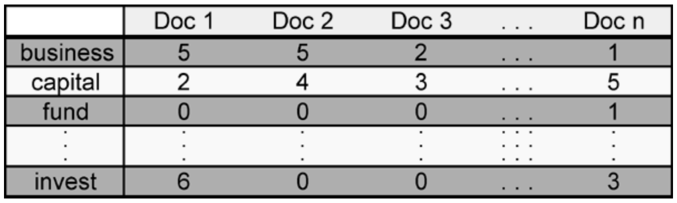
TID	Beer	Milk	Diaper	Bread	Coke
T1	0	1	0	1	1
T2	1	0	0	1	0
T3	1	1	1	0	1
T4	1	1	1	1	0
T5	0	1	1	0	1

Figure 1: Portion of a Typical Supermarket Transaction Database

The association rule discovery methods [2] first find groups of items occurring frequently together in many transactions. Such groups of items are referred to as *frequent item sets*. In the ARHP method, we use the discovered frequent item sets to form a hypergraph, where vertices are items and each hyperedge represents a frequent item set. Then a hypergraph partitioning algorithm [21] is used to find the item clusters. The similarity among items is captured implicitly by the frequent item sets.

In the document retrieval domain, it is also possible to view a set of documents in a transactional form.

In this case, each document corresponds to an item and each possible feature corresponds to a transaction. The entries in the table (domain of document attribute) represents the frequency of occurrence of a specified feature (word) in that document. A frequent item sets found using the association rule discovery algorithm corresponds to a set of documents that have a sufficiently large number of features (words) in common. These frequent item sets are mapped into hyperedges in a hypergraph. A typical document-feature dataset, represented as a transactional database, is depicted in Figure 2.



	Doc 1	Doc 2	Doc 3	...	Doc n
business	5	5	2	...	1
capital	2	4	3	...	5
fund	0	0	0	...	1
⋮	⋮	⋮	⋮	⋮	⋮
invest	6	0	0	...	3

Figure 2: A Transactional View of a Typical Document-Feature Set

A hypergraph [5] $H = (V, E)$ consists of a set of vertices (V) and a set of hyperedges (E). A hypergraph is an extension of a graph in the sense that each hyperedge can connect more than two vertices. In our model, the set of vertices V corresponds to the set of documents being clustered, and each hyperedge $e \in E$ corresponds to a set of related documents. A key problem in modeling data items as a hypergraph is determining what related items can be grouped as hyperedges and determining the weights of the hyperedge. In this case, hyperedges represent the frequent item sets found by the association rule discovery algorithm.

Association rules capture the relationships among items that are present in a transaction [3]. Let T be the set of transactions where each transaction is a subset of the item-set I, and C be a subset of I. We define the *support count* of C with respect to T to be:

$$\sigma(C) = |\{t | t \in T, C \subseteq t\}|.$$

Thus $\sigma(C)$ is the number of transactions that contain C. An *association rule* is an expression of the form $X \overset{s,\alpha}{\Longrightarrow} Y$, where $X \subseteq I$ and $Y \subseteq I$. The *support* s of the rule $X \overset{s,\alpha}{\Longrightarrow} Y$ is defined as $\sigma(X \cup Y)/|T|$, and the confidence α is defined as $\sigma(X \cup Y)/\sigma(X)$. The task of discovering an association rule is to find all rules $X \overset{s,\alpha}{\Longrightarrow} Y$, such that s is greater than a given minimum support threshold and α is greater than a given minimum confidence threshold. The association rule discovery is composed of two steps. The first step is to discover all the frequent item-sets (candidate sets that have support greater than the minimum support threshold specified). The second step is to generate association rules from these frequent item-sets.

The frequent item sets computed by an association rule algorithm such as Apriori are excellent candidates to find such related items. Note that these algorithms only find frequent item sets that have support greater than a specified threshold. The value of this threshold may have to be determined in a domain specific manner. The frequent item sets capture the relationships among items of size greater than or equal to 2. Note that distance based relationships can only capture relationships among pairs of data points whereas the frequent items sets can capture relationship among larger sets of data points. This added modeling power is nicely captured in our hypergraph model.

The hypergraph representation can then be used to cluster relatively large groups of related items by partitioning them into highly connected partitions. One way of achieving this is to use a hypergraph partitioning algorithm that partitions the hypergraph into two parts such that the weight of the hyperedges that are cut by the partitioning is minimized. Note that by minimizing the hyperedge-cut we essentially minimize the relations that are violated by splitting the items into two groups. Now each of these two parts can be further bisected recursively, until each partition is highly connected. For this task we use HMETIS [21], a multi-level hypergraph partitioning algorithm which can partition very large hypergraphs (of size $>$ 100K nodes) in minutes on personal computers.

Once, the overall hypergraph has been partitioned into k parts, we eliminate bad clusters using the following cluster fitness criterion. Let e be a set of vertices representing a hyperedge and C be a set of vertices representing a partition. The fitness function that measures the goodness of partition C is defined as follow:

$$fitness(C) = \frac{\sum_{e \subseteq C} Weight(e)}{\sum_{|e \cap C| > 0} Weight(e)}$$

The fitness function measures the ratio of weights of edges that are within the partition and weights of edges involving any vertex of this partition.

Each good partition is examined to filter out vertices that are not highly connected to the rest of the vertices of the partition. The connectivity function of vertex v in C is defined as follow:

$$connectivity(v, C) = \frac{|\{e | e \subseteq C, v \in e\}|}{|\{e | e \subseteq C\}|}$$

The connectivity measures the percentage of edges that each vertex is associated with. High connectivity value suggests that the vertex has many edges connecting good proportion of the vertices in the partition. The vertices with connectivity measure greater than a give threshold value are considered to belong to the partition, and the remaining vertices are dropped from the partition.

In ARHP, filtering out of non-relevant documents can also be achieved using the support criteria in the association rule discovery components of the algorithm. Depending on the support threshold. documents that do not meet support (i.e., documents that do not share large enough subsets of words with other documents) will be pruned. This feature is particularly useful for clustering large document sets which are returned by standard search engines using keyword queries.

2.2 Principal Direction Divisive Partitioning (PDDP) Algorithm

In the principal direction algorithm, each document is represented by a feature vector of word frequencies, scaled to unit length. The algorithm is a divisive method in the sense that it begins with all the documents in a single large cluster, and proceeds by splitting it into subclusters in recursive fashion. At each stage in the process, the method (a) selects an unsplit cluster to split, and (b) splits that cluster into two subclusters. For part (a) we use a *scatter value*, measuring the average distance from the documents in a cluster to the mean [13], though we could also use just the cluster size if it were desired to keep the resulting clusters all approximately the same size. For part (b) we construct a linear discriminant function based on the *principal direction* (the direction of maximal variance). Specifically, we compute the mean of the documents within the cluster, and then the principal direction with respect to that mean. This defines a hyperplane normal to the principal direction and passing through the mean. This hyperplane is then used to split the cluster into two parts which become the two *children* clusters to the given cluster. This entire cycle is repeated as many times as desired resulting in a binary tree hierarchy of clusters in which the root is the entire document set, and each interior node has been split into two children. The leaf nodes then constitute a partitioning of the entire document set.

The definition of the hyperplane is based on principal component analysis, similar to the Hotelling or Karhunen-Loeve Transformation [13]. We compute the principal direction as the leading eigenvector of the sample covariance matrix. This is the most expensive part, for which we use a fast Lanczos-based singular value solver [16]. By taking advantage of the high degree of sparsity in the term frequency matrix, the Lanczos-based solver is very efficient, with cost proportional to the number of nonzeroes in the term frequency matrix. This has been discussed in more detail in [7].

This method differs from that of Latent Semantic Indexing (LSI) [6] in many ways. First of all, LSI was originally formulated for a different purpose, namely as a method to reduce the dimensionality of the search space for the purpose of handling queries: retrieving some documents given a set of search terms. Secondly, it operates on the unscaled vectors, whereas we scale the document vectors to have unit length. Thirdly, in LSI, the singular value decomposition of the matrix of document vectors itself are computed, whereas we shift the documents so that their mean is at the origin in order to compute the covariance matrix. Fourthly, the LSI method must compute many singular vectors of the entire matrix of document vectors, perhaps on the order of 100 such singular vectors, but it must do so only once at the beginning of the processing. In our method, we must compute only the single leading singular vector (the vector $\mathbf{u}$), which is considerably easier to obtain. Of course we must repeat this computation on each cluster found during the course of the algorithm, but all the later clusters are much smaller than the initial "root" cluster, and hence the later computations are much faster.

In most of our experiments, we have used the norm scaling, in which each document is represented by a feature vector of word counts, scaled to unit length in the usual Euclidean norm. This leaves the sparsity pattern untouched. An alternate scaling is the TFIDF scaling [28], but this scaling fills in all zero entries with nonzeroes, drastically increasing the cost of the overall algorithm by as much as a factor of 20. In spite of the increased costs, the TFIDF scaling did not lead to any noticeable improvement in the PDDP results in our experiments [8].

2.3 Hierarchical Agglomeration Clustering

A classical algorithm we have implemented is a bottom up *hierarchical agglomeration clustering (HAC)* method based on the use of a distance function [13]. We start with trivial clusters, each containing one document. We cycle through a loop in which the two "closest clusters" are merged into one cluster. Each loop cycle reduces the number of clusters by 1, and this is repeated until the desired number of clusters is reached. For these experiments, we chose a distance function based on the "cosine" measure (essentially the cosine of the angle between the two documents in N-space), where each cluster was represented by its mean. The cluster means were scaled by the corresponding cluster sizes to discourage large clusters.

2.4 AutoClass

The other algorithm we use is *AutoClass*. AutoClass [11] is based on the probabilistic mixture modeling [29]. Given a set of data X, AutoClass finds maximum parameter values $\hat{\vec{V}}$ for a specific probability distribution functions T of the clusters.

Given $\hat{\vec{V}} = \{\hat{\vec{V_C}}, \hat{\vec{V_1}}, \hat{\vec{V_2}}, \ldots, \hat{\vec{V_k}}\}$, where $\hat{\vec{V_C}} = \{\hat{\pi_1}, \hat{\pi_2}, \ldots, \hat{\pi_k}\}$ and $\hat{\pi_j}$ is a class mixture probability, AutoClass calculates the probability that data point X_i belongs to class C_j by Bayes' rule:

$$P(X_i \in C_j) \quad = \quad \frac{\hat{\pi_j} P(X_i | \hat{\vec{V_j}})}{\sum_{l=1}^{k} \hat{\pi_l} P(X_i | \hat{\vec{V_l}})}$$

One of the advantages of AutoClass is that it has a theoretical foundation using Bayesian statistics. The clustering results provide the full description of each cluster in terms of probability distribution of each attributes. It also works well for both discrete and continuous attributes. The results of the clustering is fuzzy, i.e., it gives probabilities of one data point belonging to different clusters. Analysts can determine the cluster membership of a data point based on these probabilities.

One of the weaknesses of AutoClass is that the underlying probability model assumes independence of attributes. In many domains, this assumption is too restrictive. Another problem with the basic model is that it does not provide a satisfactory distribution function for ordered discrete attributes [11]. Furthermore unrelevant attributes (with respect to clustering) or hidden biases may dominate the clustering process.

3 Experimental Results

3.1 Experimental Setup

For the experiments we present here we selected 185 Web pages in ten broad categories: affirmative action (AA), business capital (BC), electronic commerce (EC), employee rights (ER), intellectual property (IP), industrial partnership (IPT), information systems (IS), materials processing (MP), manufacturing systems integration (MSI), and personnel management (PM).

These pages were obtained by doing a keyword search using a standard search engine. The pages were downloaded, labeled, and archived. The labeling facilitates an entropy calculation and subsequent references to any page were directed to the archive. This ensures a stable data sample since some pages are fairly dynamic in content.

Results we obtained in similar experiments with a smaller set of documents have been previously reported in [25]. Those documents were obtained in part from the Network for Excellence in Manufacturing website, on line at `http://web.miep.org:80/miep/index.html` and were used originally for the experiments described in [31]. The experiments we describe in this paper grew out of our initial set of experiments, and were used to validate on a larger dataset the results we obtained with our original experiments. We have conducted, more recently, another series of experiments with a much larger dataset (2340 documents, obtained through the Yahoo online news service) that we have used to support our scalability analysis, as described later in Section 3.2.

Word Set	Selection Criteria	Dataset Size	Comments
J_1	All words	185x10536	We select all non-stop words (stemmed).
J_2	Quantile filtering	185x946	Quantile filtering selects the most frequently occurring words until the accumulated frequencies exceed a threshold of 0.25, including all words from the partition that contributes the word that exceeds the threshold.
J_3	Top 20+ words	185x1763	We select the 20 most frequently occurring words and include all words from the partition that contributes the 20th word.
J_4	Top 5+ plus emphasized words	185x2951	We select the top 5+ words augmented by any word that was emphasized in the html document, i.e., words appearing in <TITLE>, <H1>, <H2>, <H3>, <I>, <BIG>, <STRONG>, or <EMPHASIZE> tags.
J_5	Frequent item sets	185x499	We select words from the document word lists that appear in a-priori word clusters. That is, we use an object measure to identify important groups of words.
J_6	All words with text frequency > 1	188x5106	We prune the words selected for J_1 to exclude those occurring only once.
J_7	Top 20+ with text frequency > 1	185x1328	We prune the words selected for J_3 to exclude those occurring only once.
J_8	Top 15+ with text frequency > 1	185x1105	
J_9	Top 10+ with text frequency > 1	185x805	
J_{10}	Top 5+ with text frequency > 1	185x474	

Table 1: Setup of experiments.

The word lists from all documents were filtered with a stop-list and "stemmed" using Porter's suffix-stripping algorithm [26] as implemented by [14]. We derived ten experiments, clustered the documents using the four algorithms described earlier, and analyzed the results. Our objective is to reduce the dimensionality of the clustering problem while retaining the important features of the documents. The ten experiments we conducted are distinguished by further selection rules, as shown in Table 1.

3.2 Evaluation of Clustering Results

Validating clustering algorithms and comparing performance of different algorithms is complex because it is difficult to find an objective measure of quality of clusters. We decided to use entropy [27] as a measure of goodness of the clusters. When a cluster contains documents from one class only, the entropy value is 0.0 for the cluster and when a cluster contains documents from many different classes, then entropy of the cluster is higher. The total entropy is calculated as the weighted sum of entropies of the clusters. We compare the results of the various experiments by comparing their entropy across algorithms and across feature selection methods (Fig. 3). Note that the hypergraph partitioning method does not cluster all the documents, so the entropy is computed only for the documents clustered.

Our experiments suggest that clustering methods based on partitioning seem to work best for this type of information retrieval applications, because:

1. they do not require calculation of the mean of randomly chosen clusters, and so the issue of having cluster means very close in space does not apply;
2. they are linearly scalable with respect to the cardinalities of the document and feature spaces (in contrast to HAC and AutoClass which are quadratic);
3. the quality of the clusters is not affected by the dimensionality of the data sets.

In general, all the methods had similar behavior across the experiments in that the filtering based on word frequencies did not have any major impact, except for the frequent item set used in experiment J_5, which is discussed later. Both the ARHP and PDDP methods performed better than the traditional methods (except for HAC with norm scaling) regardless of the feature selection criteria.

Algorithms such as AutoClass and HAC with TFIDF scaling become computationally prohibitive as the dimensionality is increased. For example, when no feature selection criteria was used (dataset size of 185 $\times$ 10538), ARHP and PDDP took less than 2 minutes, whereas HAC took 1 hour and 40 minutes and AutoClass took 38 minutes.

We have tried the PDDP algorithm on a larger dataset (2340 documents) and our experiments show that the algorithm scales up linearly with the number of non-zero entries in the term frequency matrix [8]. As each document uses only a small fraction of the entire dictionary of words, the term frequency matrix is very sparse. In this larger dataset only 0.68% of the entries were nonzero. The algorithm is able to take advantage of this sparsity, yielding scalable performance, as illustrated in 4.

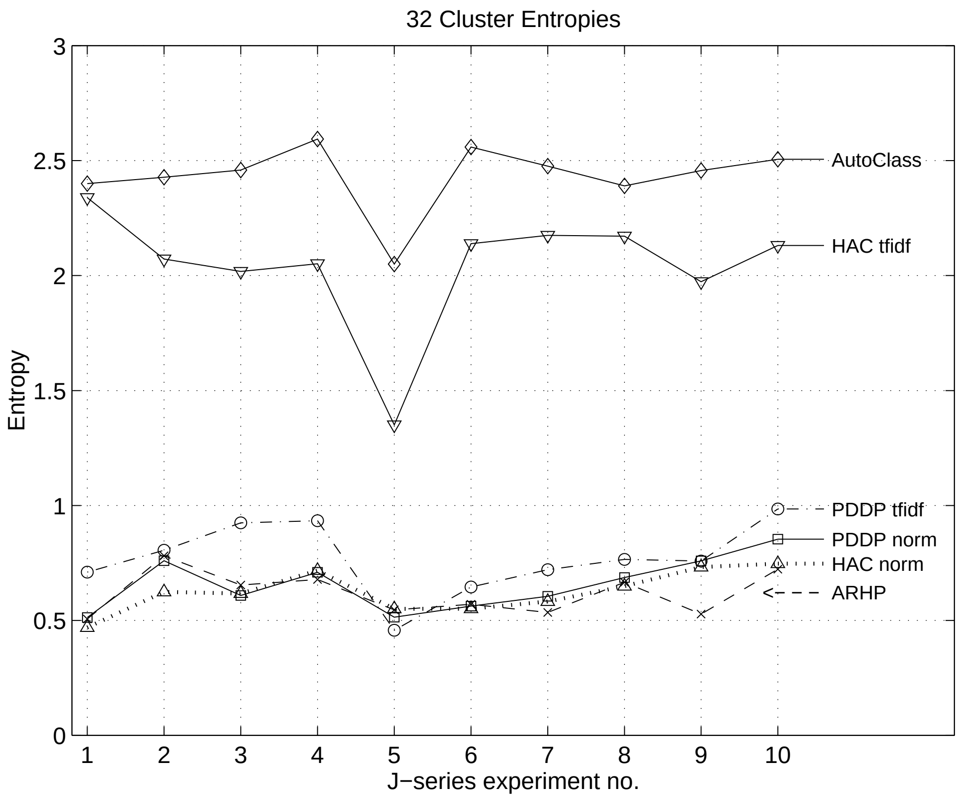


Figure 3: Entropy comparison of different algorithms with 32 clusters. The results shown for ARHP are for slightly different numbers of clusters for each experiment, precisely 30 clusters for J_1, 28 for J_2, 32 for J_3, 35 for J_4, 35 for J_5, 31 for J_6, 33 for J_7, 32 for J_8, 40 for J_9, 38 for J_10. Note that lower entropy indicates better cohesiveness of clusters.

Aside from overall performance and quality of clusters, the experiments point to a few other notable conclusions. As might be expected, in general clustering algorithms yield better quality clusters when the full set of feature is used (experiment J_1). Of course, as the above discussion shows, for large datasets the computational costs may be prohibitive. It is therefore important to select a smaller set of representative features to improve the performance of clustering algorithms without losing too much quality. Our experiments with various feature selection methods represented in J_1 through J_{10}, show that restricting the feature set to those only appearing in the frequent item sets (discovered as part of the association rule algorithm), has succeeded in identifying a small set of features that are relevant to the clustering task. In fact, for most algorithms, the experiment J_5 produced results that were better than those obtained by using the full set.

It should be noted that the conclusions drawn in the above discussion have been confirmed by an earlier experiment using a totally independent set of documents [25].

For any particular experiment, we can better judge the quality of the clustering by looking at the distribution of class labels among clusters. Figure 5 shows the class distribution for the J1 experiment

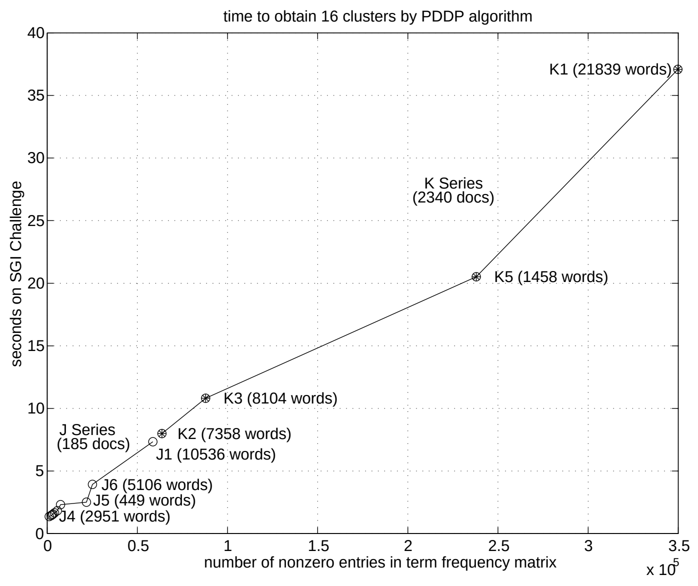


Figure 4: Time to obtain 16 clusters for various data sets using the PDDP Algorithm. The time in seconds on an SGI challenge (vertical axis) is plotted against the number of nonzeroes in the term frequency matrix (horizontal axis).

PDDP with norm scaling									
AA	BC	EC	ER	IP	IPT	IS	MP	MSI	PM
14	.	.	.	.	.	.	.	.	.
6	.	.	2	.	.	.	.	.	1
.	10	.	.	.	1	.	.	.	.
.	9	.	.	.	.	.	1	.	1
.	.	15	.	.	.	1	.	.	.
.	.	.	13	.	.	.	.	.	1
.	.	3	1	7	.	1	1	.	.
.	.	.	.	12	.	.	.	.	.
.	.	.	.	.	7	1	1	.	.
.	.	.	.	.	6	3	1	2	.
.	.	.	.	.	4	.	1	4	.
.	.	.	.	.	.	11	.	1	.
.	.	.	.	.	.	.	11	.	.
.	.	.	.	.	.	2	1	11	.
.	.	.	.	.	.	.	.	.	12
.	.	1	.	.	1	.	.	.	4
entropy = 0.690									

PDDP with TFIDF scaling									
AA	BC	EC	ER	IP	IPT	IS	MP	MSI	PM
13	.	.	.	.	.	.	.	.	1
6	.	.	.	.	.	.	.	.	.
.	13	.	.	.	.	1	.	.	.
.	.	11	.	.	.	5	.	.	.
.	.	8	.	.	.	.	.	.	.
.	.	.	11	.	.	.	.	.	1
.	4	.	.	12	.	1	.	.	.
.	2	.	.	.	9	.	.	1	.
.	.	.	.	.	8	5	2	3	.
.	.	.	.	.	.	2	9	4	.
.	.	.	.	.	.	.	3	.	.
.	.	.	.	.	.	2	1	6	.
.	.	.	.	.	.	.	1	2	.
1	.	.	5	7	.	.	.	.	11
.	.	.	.	.	2	3	1	2	5
.	.	.	.	.	.	.	.	.	1
entropy = 1.058									

Figure 5: Distribution of documents among clusters using the PDDP algorithm with or without TFIDF scaling for 16 clusters. Each column shows how many documents for each label appear in each of the clusters.

using the PDDP algorithm with and without TFIDF scaling. Full results (including the data sets) are available on the Web at `http://www.cs.umn.edu/~jmoore/wap2.html`. Similarly, Figure 6 shows the class distribution for the J1 experiment using the HAC algorithm with and without TFIDF scaling, Figure 7 shows the results of ARHP, and Figure 8 shows the results of AutoClass.

HAC with norm scaling									
AA	BC	EC	ER	IP	IPT	IS	MP	MSI	PM
19	.	.	.	.	.	.	.	.	1
.	16	.	.	1	.	.	.	.	.
.	.	9	.	.	.	2	1	.	.
.	.	8	.	.	.	.	.	.	.
1	.	.	8	.	.	.	.	.	.
.	.	.	7	1	.	.	.	.	.
.	.	.	.	17	.	.	.	.	.
.	2	2	.	.	5	.	.	1	.
.	.	.	.	.	8	.	.	.	.
.	.	.	.	.	3	1	3	3	.
.	.	.	.	.	.	8	.	.	.
.	.	.	.	.	.	.	11	2	.
.	.	.	.	.	3	4	.	7	.
.	.	.	.	.	.	2	2	5	.
.	1	.	.	.	.	2	.	.	5
.	.	.	1	.	.	.	.	.	13
entropy = 0.684									

HAC with TFIDF scaling									
AA	BC	EC	ER	IP	IPT	IS	MP	MSI	PM
8	.	.	.	.	.	.	.	.	.
2	.	1	2	.	2	.	.	2	.
2	5	3	1	.	1	2	.	1	1
.	4	.	.	3	3	1	3	2	.
1	.	2	.	2	.	.	1	1	1
.	2	4	.	.	1	.	1	.	.
.	2	1	3	1	1	3	1	2	2
.	.	.	3	.	2	1	1	.	1
1	.	1	4	1	2	1	3	3	.
4	4	2	.	5	.	.	.	.	1
1	.	.	.	3	.	2	1	.	1
.	.	.	2	3	.	3	.	.	.
.	.	.	.	.	3	.	2	2	1
1	1	.	.	.	1	1	2	1	1
.	1	2	1	.	2	3	.	2	5
.	.	3	.	1	1	2	2	2	5
entropy = 2.406									

Figure 6: Distribution of documents among clusters using the Hierarchical Agglomeration Clustering algorithm with or without TFIDF scaling for 16 clusters.

ARHP									
AA	BC	EC	ER	IP	IPT	IS	MP	MSI	PM
9	.	.	.	.	.	.	.	.	.
9	.	.	1	.	.	.	.	.	.
.	8	.	.	.	.	.	.	.	.
.	.	10	.	.	.	.	.	.	.
.	.	2	2	2	.	.	1	.	.
.	.	.	4	.	1	.	.	.	3
.	.	.	9	.	.	.	.	.	.
.	.	.	.	11	.	.	.	.	.
1	3	1	.	6	.	.	.	.	.
.	.	.	.	.	9	.	.	.	.
.	6	1	.	.	7	.	.	1	.
.	.	2	.	.	.	9	.	1	.
.	1	3	.	.	2	7	.	.	.
.	.	.	.	.	.	1	6	1	.
.	.	.	.	.	.	.	3	8	.
.	.	.	.	.	.	1	6	7	.
.	.	.	.	.	.	.	.	.	14
1	1	.	.	.	.	1	1	.	2
entropy = 0.835									

Figure 7: Distribution of documents among clusters using the ARHP algorithm for 18 clusters. The last cluster contains documents that were not clustered by ARHP, which yielded 17 clusters.

AutoClass									
AA	BC	EC	ER	IP	IPT	IS	MP	MSI	PM
2	.	.	2	2	.	.	.	.	.
5	8	3	2	4	6	6	.	2	6
.	4	.	.	1	1	2	.	2	.
.	2	.	.	1	.	2	.	1	.
3	.	6	1	4	1	2	1	.	.
2	.	3	7	3	.	1	.	.	2
.	.	1	.	.	6	.	.	.	.
2	2	2	.	.	3	6	5	1	5
4	.	.	1	.	2	.	5	1	1
.	.	1	.	.	.	.	2	.	.
.	1	.	1	1	.	.	3	5	.
.	.	.	1	2	.	.	1	4	.
2	2	3	1	1	.	.	.	2	5
entropy = 2.050									

Figure 8: Distribution of documents among clusters using the AutoClass algorithm for 16 clusters. Not all the documents are clustered.

4 Related Work

A number of Web agents use various information retrieval techniques [15] and characteristics of open hypertext Web documents to automatically retrieve, filter, and categorize these documents [10, 9, 12].

For example, HyPursuit [30] uses semantic information embedded in link structures as well as document content to classify and group documents by the terms they contain and their hyperlink structures. The system requires that information be maintained in the routers.

BO (Bookmark Organizer) [24] combines hierarchical agglomerative clustering techniques and user interaction to organize collection of Web documents listed in a personal bookmark file.

Pattern recognition methods and word clustering using the Hartigan's K-means partitional clustering algorithm are used in [31] to discover salient HTML document features (words) that can be used in finding similar HTML documents on the Web. The clustering algorithm does not scale well to large numbers of documents. Broder [9] calculates a sketch for every document on the web and then clusters together similar documents whose sketches exceed a threshold of resemblance. Given a document's URL, similar documents can be easily identified, but an index for the whole WWW needs to be maintained.

Maarek [24] uses the Hierarchical Agglomerative Clustering method to form clusters of the documents listed in a personal bookmark file. Individual documents are characterized by profile vectors consisting of pairs of lexically affine words, with document similarity a function of how many indices they share. This method may not scale well to large document searches.

The Syskill & Webert system [1] represents an HTML page with a Boolean feature vector, and then uses naive Bayesian classification to find web pages that are similar, but for only a given single user profile. Balabanovic [4] presents a system that uses a single well-defined profile to find similar web documents for a user. Candidate web pages are located using best-first search, comparing their word vectors against a user profile vector, and returning the highest -scoring pages. A TFIDF scheme is used to calculate the word weights, normalized for document length. The system needs to keep a large dictionary and is limited to one user.

A well-known and widely used technique for dimensionality reduction is Principal Component Analysis (PCA) [19]. Consider a data set with n data items and m variables. PCA computes a covariance matrix of size $m \times m$, and then calculate the k leading eigenvectors of this covariance matrix. These k leading eigenvectors of this matrix are principal features of the data. The original data is mapped along these new principal directions. This projected data has lower dimensions and can now be clustered using traditional clustering algorithms such as K-means [20], Hierarchical clustering [20], or *AutoClass*.

PCA provides several guidelines on how to determine the right number of dimension k for given data based on the proportion of variance explained or the characteristic roots of the covariance matrix. However, as noted in [19], different methods provide widely different guidelines for k on the same data, and thus it can be difficult to find the right number of dimension. The choice of a small k can lose important features of the data. On the other hand, the choice of a large k can capture most of the important features, but the dimensionality might be too large for the traditional clustering algorithms to work effectively.

Latent Semantic Indexing (LSI) [6] is a dimensionality reduction technique extensively used in information retrieval domain and is similar in nature to PCA. Instead of finding the singular value decomposition of the covariance matrix, it finds the singular value decomposition of the original $n \times m$ data.

Both PCA and LSI are preprocessing methods which produce a much lower dimensional representation of the dataset for subsequent processing by another algorithm. In the context of query systems, LSI has been singularly successful in reducing the noise in the data, leading to much higher precision in results from user queries [6]. They may also be considered as possible preprocessing modules in the context of unsupervised clustering, and some preliminary experiments in this direction have been carried out using LSI followed by K-means and PDDP, yielding respective entropies of .834 and .859. To obtain these results, we used LSI to extract a dimension 10 approximation to the term frequency matrix, which was then used as the basis for the subsequent K-means or PDDP method.

The main difficulty with the LSI or PCA methods is the necessity to compute the k leading singular values and vectors of the term frequency matrix, where k is the desired dimension. A naive dense matrix solver takes $O(n^3)$ operations to compute it and hence is prohibitively expensive. A method which takes advantage of sparsity could be used to speed this up substantially. An example is the Lanczos method [16] which has been used with great success in the PDDP algorithm. However, it is considerably more difficult to compute the leading k singular values and vectors in LSI than just the one leading singular vector as in PDDP. But even if the time is available, the resulting low dimension approximation will typically be dense. This substantially increases the processing cost for the subsequent clustering method as well as potentially occupying as much space as the original data, depending on the choice of k.

The Kohonen Self-Organizing Feature Map [22] is a neural network based scheme that projects high dimensional input data into a feature map of a smaller dimension such that the proximity relationships among input data are preserved. On data sets of very large dimensionality such as those discussed here, convergence could be slow, depending upon the initialization.

5 Conclusion

In this paper we have presented two new methods for clustering, namely, Association Rule Hypergraph Partitioning and Principal Direction Divisive Partitioning, that are particularly suitable for the type of information retrieval applications discussed above. These methods do not depend on distance measures, and perform well in high dimensional spaces.

Our experiments suggest that both of these methods perform better than other traditional clustering algorithms regardless of the techniques used for feature selection. In particular, they both perform well, even when all of the features from each document are used in clustering. In addition, the experiments suggest that if the features selected are restricted to those present in frequent item sets, such as those derived from the Apriori Algorithm, then the traditional methods tend to perform better. It is also evident that, the hypergraph partitioning method may perform better, if the features selected include those words emphasized by document authors through the use of HTML tags.

Our future research plans include developing methods for incremental clustering or classification of documents after discovering an initial set of clusters. Furthermore, we plan to investigate the use of clustering techniques proposed here for word clustering. These word clusters can then be used to classify new documents or to search for related documents on the Web.

References

[1] Mark Ackerman and et al. Learning probabilistic user profiles. *AI Magazine*, 18(2):47–56, 1997.

[2] A. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast discovery of association rules. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996.

[3] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast discovery of association rules. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996.

[4] Marko Balabanovic, Yoav Shoham, and Yeogirl Yun. An adaptive agent for automated Web browsing. *Journal of Visual Communication and Image Representation*, 6(4), 1995. http://www-diglib.stanford.edu/cgi-bin/WP/get/SIDL-WP-1995-0023.

[5] C. Berge. *Graphs and Hypergraphs.* American Elsevier, 1976.

[6] M. W. Berry, S. T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595, 1995.

[7] D. L. Boley. Principal Direction Divisive Partitioning. Technical Report TR-97-056, Department of Computer Science, University of Minnesota, Minneapolis, 1997.

[8] D. L. Boley. Hierarchical taxonomies using divisive partitioning. Technical Report TR-98-012, Department of Computer Science, University of Minnesota, Minneapolis, 1998.

[9] Andrei Z. Broder, Steven C. Glassman, and Mark S. Manasse. Syntactic clustering of the Web. In *Proc. of 6th International World Wide Web Conference*, April 1997. http://proceedings.www6conf.org/HyperNews/get/PAPER205.html.

[10] C. Chang and C. Hsu. Customizable multi-engine search tool with clustering. In *Proc. of 6th International World Wide Web Conference*, 1997.

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

[12] Liren Chen and Katya Sycara. Webmate : A personal agent for browsing and searching. In *Proc. of 2nd International Conference on Autonomous Agents*, 1998.

[13] Richard O. Duda and Peter E. Hart. *Pattern Classification and scene analysis.* John Wiley & Sons, 1973.

[14] W. B. Frakes. Stemming algorithms. In W. B. Frakes and R. Baeza-Yates, editors, *Information Retrieval Data Structures and Algorithms*, pages 131–160. Prentice Hall, 1992.

[15] W. B. Frakes and R. Baeza-Yates. *Information Retrieval Data Structures and Algorithms.* Prentice Hall, Englewood Cliffs, NJ, 1992.

[16] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins Univ. Press, 3rd edition, 1996.

[17] E.H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering based on association rule hypergraphs. In *Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 9–13, Tucson, Arizona, 1997.

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

[19] J. E. Jackson. *A User's Guide To Principal Components.* John Wiley & Sons, 1991.

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

[21] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in VLSI domain. In *Proceedings ACM/IEEE Design Automation Conference*, 1997.

[22] T. Kohonen. *Self-Organization and Associated Memory.* Springer-Verlag, 1988.

[23] S.Y. Lu and K.S. Fu. A sentence-to-sentence clustering procedure for pattern analysis. *IEEE Transactions on Systems, Man and Cybernetics*, 8:381–389, 1978.

[24] Yoelle S. Maarek and Israel Z. Ben Shaul. Automatically organizing bookmarks per contents. In *Proc. of 5th International World Wide Web Conference*, May 1996. http://www5conf.inria.fr/fich_html/papers/P37/Overview.html.

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

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

[27] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[28] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[29] D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions.* John Wiley & Sons, 1985.

[30] Ron Weiss, Bienvenido Velez, Mark A. Sheldon, Chanathip Nemprempre, Peter Szilagyi, Andrzej Duda, and David K. Gifford. Hypursuit: A hierarchical network search engine that exploits content-link hypertext clustering. In *Seventh ACM Conference on Hypertext*, March 1996. http://paris.lcs.mit.edu/~rweiss/.

[31] Marilyn R. Wulfekuhler and William F. Punch. Finding salient features for personal Web page categories. In *Proc. of 6th International World Wide Web Conference*, April 1997. http://proceedings.www6conf.org/HyperNews/get/PAPER118.html.