Artificial Intelligence Review **13:** 365–391, 1999.
© 2000 *Kluwer Academic Publishers. Printed in the Netherlands.*

Document Categorization and Query Generation on the World Wide Web Using WebACE

DANIEL BOLEY, MARIA GINI, ROBERT GROSS, EUI-HONG (SAM) HAN, KYLE HASTINGS, GEORGE KARYPIS, VIPIN KUMAR, BAMSHAD MOBASHER and JEROME MOORE

Abstract. We present WebACE, an agent for exploring and categorizing documents on the World Wide Web based on a user profile. The heart of the agent is an unsupervised categorization of a set of documents, combined with a process for generating new queries that is used to search for new related documents and for filtering the resulting documents to extract the ones most closely related to the starting set. The document categories are not given *a priori*. We present the overall architecture and describe two novel algorithms which provide significant improvement over Hierarchical Agglomeration Clustering and AutoClass algorithms and form the basis for the query generation and search component of the agent. We report on the results of our experiments comparing these new algorithms with more traditional clustering algorithms and we show that our algorithms are fast and sacalable.

Keywords: clustering, divisive partitioning, graph partitioning, principal component analysis, web documents

1. Introduction

After a short description of the architecture of WebACE in Section 3, we describe the clustering algorithms in Section 4. In Section 5, we report on the results obtained on a number of experiments using different methods to select sets of features from the documents, and show that our partitioning-based clustering methods perform better than traditional distance based clustering. We also analyze the complexity of the two clustering algorithms and show they are scalable. In Section 6, we show how to use words obtained from clusters of documents to generate queries for related documents on the Web.

2. Related Work

The heterogeneity and the lack of structure that permeates much of the information sources on the World Wide Web makes automated discovery, organization, and management of Web-based information difficult. Traditional search and indexing tools of the Internet and the World Wide Web such as Lycos, Alta Vista, WebCrawler, MetaCrawler, and others provide some comfort to users, but they do not generally provide structural information nor categorize, filter, or interpret documents. A recent study provides a comprehensive and statistically thorough comparative evaluation of the most popular search tools (Leighton and Srivastava 1997).

In recent years these factors have prompted researchers to develop more intelligent tools for information retrieval, such as intelligent Web agents. The agent-based approach to Web mining involves the development of sophisticated AI systems that can act autonomously or semi-autonomously on behalf of a particular user, to discover and organize Web-based information. Generally, the agent-based Web mining systems can be placed into the following categories:

Intelligent Search Agents. Several intelligent Web agents have been developed that search for relevant information using characteristics of a particular domain (and possibly a user profile) to organize and interpret the discovered information. For example, agents such as FAQ-Finder (Hammond et al. 1995), Information Manifold (Kirk et al. 1995), and OCCAM (Kwok and Weld 1996) rely either on pre-specified and domain specific information about particular types of documents, or on hard coded models of the information sources to retrieve and interpret documents. Other agents, such as ShopBot (Doorenbos et al. 1996) and ILA (Perkowitz and Etzioni 1995), attempt to interact with and learn the structure of unfamiliar information sources. ShopBot retrieves product information from a variety of vendor sites using only general information about the product domain. ILA, on the other hand, learns models of various information sources and translates these into its own internal concept hierarchy.

Information Filtering/Categorization. A number of Web agents use various information retrieval techniques (Frakes and Baeza-Yates 1992) and characteristics of open hypertext Web documents to automatically retrieve, filter, and categorize. For example, HyPursuit (Weiss et al. 1996) uses semantic information embedded in link structures as well as document content to create cluster hierarchies of hypertext documents, and structure an information space. BO (Bookmark Organizer) (Maarek and Shaul 1996) combines hierarchical clustering techniques and user interaction to organize a collection of Web documents based on conceptual information. Pattern recognition methods and word clustering using the Hartigan's K-means partitional clustering algorithm are used in Wulfekuhler and Punch (1997) to discover salient HTML document features (words) that can be used in finding similar HTML documents on the Web.

Personalized Web Agents. Another category of Web agents includes those that obtain or learn user preferences and discover Web information sources that correspond to these preferences, and possibly those of other individuals with similar interests (using collaborative filtering). A few recent examples of such agents include WebWatcher (Armstrong et al. 1995), Syskill and Webert, and others. For example, Syskill and Webert (Ackerman et al. 1997) utilizes a user profile and learns to rate Web pages of interest using a Bayesian classifier. Balabanovic (Balabanovic et al. 1995) uses a single well-defined profile to find similar Web documents. Candidate Web pages are located using best-first search. The system needs to keep a large dictionary and is limited to a single user.

WebACE incorporates aspects from all three categories. It is an intelligent search agent which automatically generates a personalized user profile as well as an automatic categorization of search results.

3. WebACE Architecture

WebACE's architecture is shown in Figure 1. As the user browses the Web, the profile creation module builds a custom profile by recording documents of interest to the user. The number of times a user visits a document and the total amount of time a user spends viewing a document are just a few methods for determining user interest (Ackerman et al. 1997, 1995; Balabanovic et al. 1995). Once WebACE has recorded a sufficient number of interesting documents, each document is reduced to a document vector and the document vectors are passed to the clustering modules. WebACE uses two novel algorithms for clustering which can provide significant improvement in both run-time performance and cluster quality over the HAC and AutoClass algorithms. These are described in Section 4.

After WebACE has found document clusters, it can use the clusters to generate queries and search for similar documents. WebACE submits the queries to the search mechanism and gathers the documents returned by the searches, which are in turn reduced to document vectors. These new documents can be used in a variety of ways. One option is for WebACE to cluster the new documents, filtering out the less relevant ones. Another is to update the existing clusters by having WebACE insert the new documents into the clusters. Yet another is to completely re-cluster both the new and old documents. Finally, the user can decide to add any or all of the new

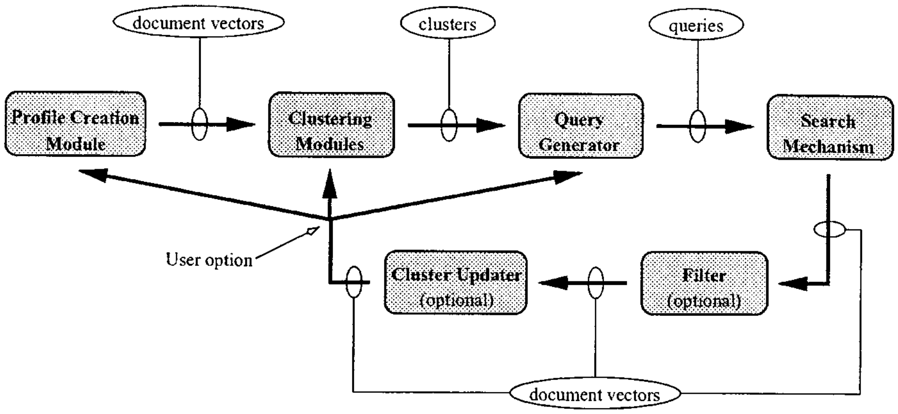


Figure 1. WebACE architecture.

documents to his profile. The query generation methods and the algorithms for incrementally updating existing clusters are discussed in Section 6. WebACE is modular so that either clustering method of Section 4 can be plugged in.

WebACE is implemented as a browser independent Java application. Monitoring the user's browsing behavior is accomplished via a proxy server. The proxy server allows WebACE to inspect the browser's HTTP requests and the resulting responses. Upon execution, WebACE spawns a browser and starts a thread to listen for HTTP requests from the browser. As the browser makes requests, WebACE creates request threads to handle them. This allows multi-threaded browsers the capability of having multiple requests pending at one time. The lifespan of these request threads is short, i.e. the duration of one HTTP request. Conversely, the browser listener thread persists for the duration of the application.

4. Clustering Methods

Existing approaches to document clustering are generally based on either probabilistic methods, or distance and similarity measures (see Frakes and Baeza-Yates 1992). Distance-based methods such as *k*-means analysis, hierarchical clustering (Jain and Dubes 1988) and nearest-neighbor clustering (Lu and Fu 1978) use a selected set of words (features) appearing in different documents as the dimensions. Each such feature vector, representing a document, can be viewed as a point in this 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. Some words are more frequent in a document than other words. Simple frequency of the occurrence of words is not adequate, as some documents are larger than others. Furthermore, some words may occur frequently across documents. Techniques such as TFIDF (Salton and McGill 1983) have been proposed precisely to deal with some of these problems.

Secondly, the number of all the words in all the documents can be very large. Distance-based schemes generally require the calculation of the mean of document clusters. In a *k*-means algorithm, randomly generated initial clusters of a very high dimensional dataset will have to calculate mean values which do not differ significantly from one cluster to the next. Hence the clustering based on these mean values does not always produce very good clusters. Similarly, probabilistic methods such as Bayesian classification used in AutoClass (Cheeseman and Stutz 1996), do not perform well when the size of the feature space is much larger than the size of the sample set. This type of data distribution seems to be characteristic of document categorization applications on the Web, such as categorizing a bookmark file. Furthermore, the underlying probability models usually assume independence of attributes (features). In many domains, this assumption may be too restrictive.

It is possible to reduce the dimensionality by selecting only frequent words from each document, or to use some other method to extract the salient features of each document. However, the number of features collected using these methods still tends to be very large, and due to the loss of some of the relevant features, the quality of clusters tends not to be as good. Other, more general methods, have also been proposed for dimensionality reduction which attempt to transform the data space into a smaller space in which relationship among data items is preserved. Then the traditional clustering algorithms can be applied to this transformed data space. Principal Component Analysis (PCA) (Jackson 1991), Multidimensional Scaling (MDS) (Jain and Dubes 1988) and Kohonen Self-Organizing Feature Maps (SOFM) (Kohonen 1988) are some of the commonly used techniques for dimensionality reduction. In addition, Latent Semantic Indexing (LSI) (Anderson 1954; Deerwester et al. 1990; Berry 1992; Berry et al. 1995) is a method frequently used in the information retrieval domain that employs a dimensionality reduction technique similar to PCA. An inherent problem with dimensionality reduction is that in the presence of noise in the data, it may result in the degradation of the clustering results. This is partly due to the fact that by projecting onto a smaller number of dimensions, the noise data may appear closer to the clean data in the lower dimensional space. In many domains, it is not always possible or practical to remove the noise as a preprocessing step. In addition, performing dimensionality reduction prior to clustering often adds a computationally prohibitive step.

Our proposed clustering algorithms which are described in this section are designed to efficiently handle very high dimensional spaces, without the need for dimensionality reduction. In contrast to traditional clustering methods, our proposed methods are linearly scalable, an advantage which makes these methods particularly suitable for use in Web retrieval and categorization agents. For our evaluation, we used two sets of sample documents retrieved from the Web to compare these algorithms to two well-known methods: Bayesian classification as used by AutoClass (Cheeseman and Stutz 1996) and *hierarchical agglomeration clustering (HAC)* based on the use of a distance function (Duda and Hart 1973).

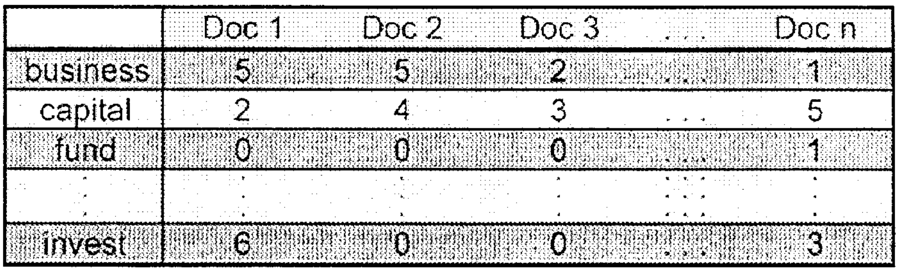
AutoClass is based on the probabilistic mixture modeling (Titterington et al. 1985), and given a data set it finds maximum parameter values for a specific probability distribution function of the clusters. The clustering results provide the full description of each cluster in terms of probability distribution of each attribute. The HAC method starts with trivial clusters, each containing one document and iteratively combines smaller clusters that are sufficiently “close” based on a distance metric. In HAC, the features in each document vector are usually weighted using the TFIDF scaling (Salton and McGill 1983), which is an increasing function of the feature’s text frequency and its inverse document frequency in the document space.

4.1 *Association rule hypergraph partitioning algorithm*

The ARHP method (Han et al. 1997a,b, 1998) is used for clustering related items in transaction-based databases, such as supermarket bar code data, using association rules and hypergraph partitioning. From a database perspective, the 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.

The ARHP method first finds set of items that occur frequently together in transactions using association rule discovery methods (Agrawal et al. 1996). These frequent item sets are then used to group items into hypergraph edges, and a hypergraph partitioning algorithm (Karypis et al. 1997) 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



	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.

table represent 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.

A hypergraph (Berge 1976) $H = (V, E)$ 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 (Agrawal et al. 1996). 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:

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

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.

Assignment of weights to the resulting hyperedges is more tricky. One obvious possibility is to use the support of each frequent item set as the weight of the corresponding hyperedge. Another possibility is to make the weight as a function of the confidence of the underlying association rules. For size two hyperedges, both support and confidence provide similar information. In fact, if two items *A* and *B* are present in equal number of transactions (i.e., if the support of item set {A} and item set {B} are the same), then there is a direct correspondence between the support and the confidence of the rules between these two items (i.e., greater the support for {A, B}, more confidence for rules "{A} ⇒ {B}" and "{A} ⇒ {B}"). However, support carries much less meaning for hperedges of size greater than two, as, in general, the support of a large hyperedge will be much smaller than the support of smaller hyperedges. Another, more natural, possibility is to define weight as a function of the support and confidence of the rules that are made of a group of items in a frequent item set. Other options include correlation, distance or similarity measure.

In our current implementation of the model, each frequent item-set is represented by a hyperedge $e \in E$ whose weight is equal to the average confidence of the association rules, called *essential* rules, that have all the items of the edge and has a singleton right hand side. We call them *essential* rules, as they capture information unique to the given frequent item set. Any rule that has only a subset of all the items in the rule is already included in the rules of subset of this frequent item set. Furthermore, all the rules that have more than 1 item on the right hand size are also covered by the subset of the frequent item set. For example, if {A, B, C} is a frequent item-set, then the hypergraph contains a hyperedge that connects A, B, and C. Consider a rule {A} ⇒ {B, C}. Interpreted as an implication rule, this information is captured by {A} ⇒ {B} and {A} ⇒ {C}. Consider the following essential rules (with confidences noted on the arrows) for the item set {A, B, C}: {A, B} $\overset{0.4}{\Rightarrow}$ {C}, {A, C} $\overset{0.6}{\Rightarrow}$ {B}, and {B, C} $\overset{0.8}{\Rightarrow}$ {A}. Then we assign weight of 0.6 ($\frac{0.4+0.6+0.8}{3}$ = 0.6) to the hyperedge connecting A, B, and C.

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 (Karypis 1997), 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:

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

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. Note that this fitness criterion can be incorporated into the partitioning algorithm as a stopping condition. With this stopping condition, only the partitions that do not meet the fitness criterion are partitioned further.

Each good partition is examined to filter our 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:

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

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 given 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.

4.2 *Principal Direction Divisive Partitioning*

The method of Principal Direction Divisive Partitioning (PDDP) (Boley 1997) is based on the computation of the leading principal direction (also known as principal component) for a collection of documents and then cutting the collection of documents along a hyperplane resulting in two separate clusters. The algorithm is then repeated on each separate cluster. The result is a binary tree of clusters defined by associated principal directions and hyperplanes. The PDDP method computes a root hyperplane, and then a child hyperplane for each cluster formed from the root hyperplane, and so on. The algorithm proceeds by splitting a leaf node into two children nodes using the leaf's associated hyperplane.

The leaf to be split next at each stage may be selected based on a variety of strategies. The simplest approach is to split all the clusters at each level of the binary tree before proceeding to any cluster at the next level. However, in our experiments, this resulted in imbalance in the sizes of the clusters, including some clusters with only 1 document. Another option is to use any appropriate measure of cohesion. For simplicity of computation, the experiments shown in this paper have been conducted using a modified *scatter* value (Duda and Hart 1973) defined below.

Each document is represented by a column of word counts and all the columns are collected into a *term frequency matrix* M, in a manner similar to Latent Semantic Indexing (LSI) (Berry et al. 1995). Specifically, the i, j-th entry, M_{ij}, is the number of occurrences of word w_i in document d_j. To make the results independent of document length, each column is scaled to have unit length in the usual Euclidean norm: $\hat{M}_{ij} = M_{ij}/\sqrt{\Sigma_i M_{ij}^2}$, so that $\Sigma_i \hat{M}_{ij}^2 = 1$. An alternative scaling is the TFIDF scaling (Salton 1983), but this scaling fills in all the zero entries in M. In our experiments, only up to 3% of the entries were nonzero, and the PDDP algorithm depends on this sparsity for its performance. Hence the TFIDF scaling substantially raises the cost of the PDDP algorithm while not yielding any improvement of the cluster quality (Boley 1997).

At each stage of the algorithm a cluster is split as follows. The centroid vector for each cluster is the vector c whose i-th component is $c_i = \Sigma_j \hat{M}_{ij}/k$, where the sum is taken over all documents in the cluster and k is the number of documents in the cluster. The principal direction for each individual cluster is the direction of maximum variance, defined to be the eigenvector corresponding to the largest eigenvalue of the unscaled sample covariance matrix $(\hat{M} - \mathbf{ce})(\hat{M} - \mathbf{ce})'$, where $\mathbf{e}$ is a row vector of all ones and $'$ denotes the matrix transpose. In our algorithm, it is obtained by computing the leading left singular vector of $(\hat{M} - \mathbf{ce})$ using a "Lanczos"-type iterative method (Boley 1997). In the "Lanczos"-type iterative method, the matrix $(\hat{M} - \mathbf{ce})$ is used only to form matrix vector products, which can be computed fast by using

$$(\hat{M} - \mathbf{ce})\mathbf{v} = \hat{M}\mathbf{v} - \mathbf{c}(\mathbf{ev}). \tag{1}$$

In this way, the matrix $(\hat{M} - \mathbf{ce})$ need not be formed explicitly, thus preserving sparsity. The resulting singular vector is the principal direction, and all the documents are projected onto this vector. Those documents with positive projections are allocated to the right child cluster, the remaining documents are allocated to the left child cluster. With this notation, we can also give the following precise definition of the *scatter* of a cluster: $\Sigma_{ij}(\hat{M}_{ij} - c_i)^2$, where the sum is taken over all documents d_j in the cluster and all the words w_i, and c_i is the i-th component of the cluster's centroid vector. In other words, the *scatter* is sum of squares of the distances from each document in the cluster to the cluster mean. The process stops when all the scatter values for the individual clusters fall below the scatter value of centroid vectors collected together.

This process differs from that in Berry et al. (1995) in that (a) we first scale the columns to have unit length to make the results independent of the document length, (b) we translate the collection of document columns so that their mean lie at the origin, (c) we compute only the single leading singular value with its associated left and right singular vectors, (d) we repeat the process on each cluster during the course of the algorithm. In LSI as described in Berry et al. (1995), the SVD is applied once to the original untranslated matrix of word counts, and the first k singular values and associated vectors are retrieved, for some choice of k. This removes much of the noise present in the data, and also yields a representation of the documents of reduced dimensionality, reducing the cost and raising the precision of user queries. The matrix produced by this LSI computation could be used as a preprocessing step to the PDDP algorithm, but generally lacks any sparsity. In addition, clusters of small variance may be lost in the noise dimensions removed by the LSI technique. In PDDP, however, property (d) ensures that such small clusters will eventually appear, though they could be divided by a hyperplane early in the splitting process leading to multiple small clusters.

5. Experimental Evaluation

5.1 *Comparative evaluation of clustering algorithms*

To compare our clustering methods with the more traditional algorithms, we selected 185 web pages in 10 broad categories: business capital (BC), intellectual property (IP), electronic commerce (EC), information systems (IS), affirmative action (AA), employee rights (ER), personnel management (PM), industrial partnership (IPT), manufacturing systems integration (MSI), and materials processing (MP).The pages in each category were obtained by doing a keyword search using a standard search engine. These pages were then 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.

The word lists from all documents were filtered with a stop-list and "stemmed" using Porter's suffix-stripping algorithm (Porter 1980) as implemented by Frakes (1992). We derived 10 experiments (according to the method used for feature selection) and clustered the documents using the four algorithms described earlier. The objective of feature selection was to reduce the dimensionality of the clustering problem while retaining the important features of the documents. Table 1 shows the feature selection methods that characterize various experiments.

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 as a measure of goodness of the clusters (with the caveat that the best entropy is obtained when each cluster contains exactly one document). For each cluster of documents, the category distribution of documents is calculated first. Then using this category distribution, the entropy of each cluster j is calculated using the formula $E_j = -\Sigma_{C_i} p_{C_i} \log P_{c_i}$ where p_{C_i} is the fraction of documents within the cluster with the category label C_i, the sum is taken over all categories, C_1, C_2, $\ldots$. In our example, C_1 = BC, C_2 = IP, etc. When a cluster contains documents from one category only, the entropy value is 0.0 for the cluster and when a cluster contains documents from many different categories, then entropy of the cluster is higher. The total entropy is calculated as the sum of entropies of the clusters weighted by the size of each cluster: $E_{\text{total}} = \Sigma_j E_j n_j / n$, where n_j is the number of documents in cluster j and n is the total number of documents. We compare the results of the various experiments by comparing their entropy across algorithms and across feature selection methods (Figure 6). Figure 3 shows the category distribution of documents in each cluster of the best AutoClass result with the entropy value 2.05. Comparing this result

Table 1. Setup of experiments. The "D" experiments were constructed in similar fashion. Their sizes can be found in Figure 9

Word set	Selection criteria	Dataset size	Comments
E1	All words	185×10536	We select all non-stop words (stemmed).
E2	All words with text frequency >1	188 × 5106	We prune the words selected for E1 to exclude those occurring only once.
E3	Top 20+ words	185×1763	We select the 20 most frequently occurring words and include all words from the partition that contributes the 20th word.
E4	Top 20+ with text frequency >1	185×1328	We prune the words selected for E3 to exclude those occurring only once.
E5	Top 15+ with text frequency >1	185×1105	
E6	Top 10+ with text frequency >1	185×805	
E7	Top 5+ with text frequency >1	185×474	
E8	Top 5+ plus emphasized words	185×2951	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⟩, ⟨B⟩, or ⟨EM⟩ tags.
E9	Quantile filtering	185×946	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.
E10	Frequent item sets	185×499	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.

to one of PDDP result with entropy value of 0.69 in Figure 4 and one of ARHP result with entropy value of 0.79 in Figure 5, we can see the big differences in the quality of the clusters obtained from these experiments. Note that in most experiments, documents with each document label have ended up in several clusters. An open question is how to re-agglomerate clusters containing similar documents that were separated by the clustering algorithm prematurely.

Our experiments suggest that clustering methods based on partitioning seem to work best for this type of information retrieval applications, because they are linearly scalable w.r.t. the cardinalities of the document and feature spaces (in contrast to HAC and AutoClass which are quadratic). In particular, both the hypergraph partitioning method and the principal component

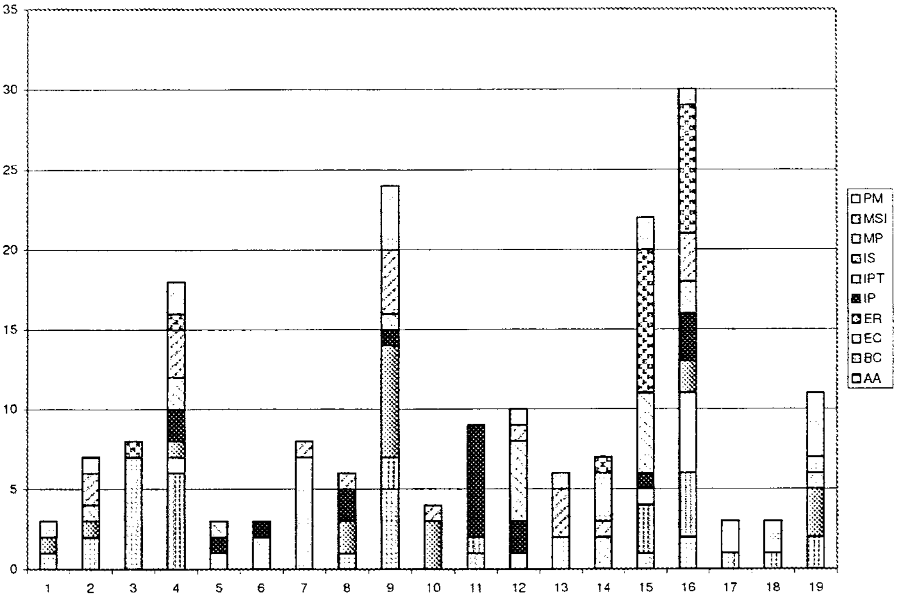


Figure 3. Class distribution of AutoClass clusters.

methods performed much better than the traditional methods regardless of the feature selection criteria used.

There were also dramatic differences in run times of the four methods. For example, when no feature selection criteria was used (dataset size of 185 × 10538), ARHP and PDDP took less than 2 minutes, whereas HAC took 1 hour and 40 minutes and AutoClass took 38 minutes.

Aside from overall performance and the 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 E_1). Of course, as the above discussion shows, for large datasets the computational costs may be prohibitive, especially in the case of HAC and AutoClass methods. It is therefore important to select a smaller set of representative features to improve the performance of clustering algorithms without loosing too much quality. Our experiments with various feature selection methods represented in E_1 through E_{10}, clearly 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, in the case of AutoClass and HAC, the experiment E_{10} produced results that were better than those obtained by using the full set.

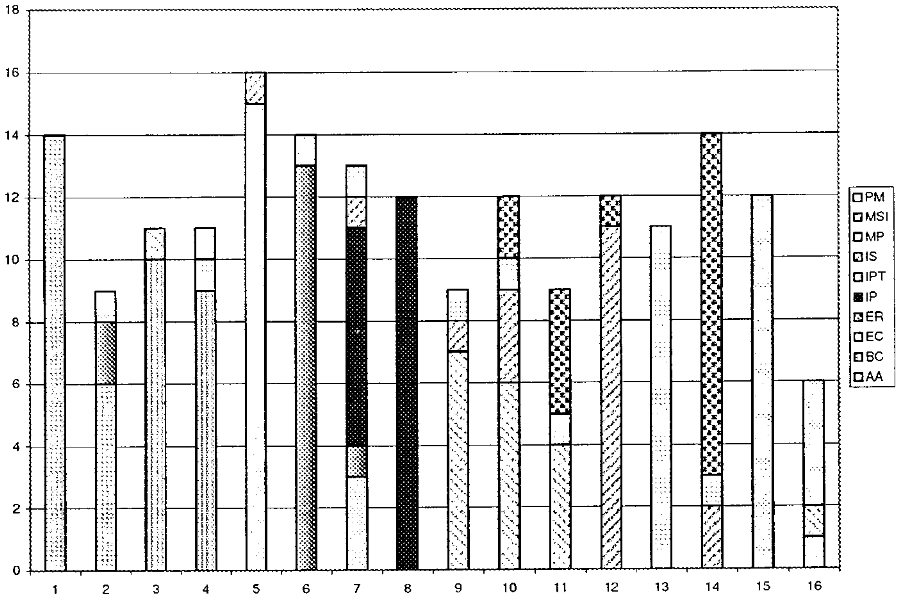


Figure 4. Class distrivution of *PDDP* clusters.

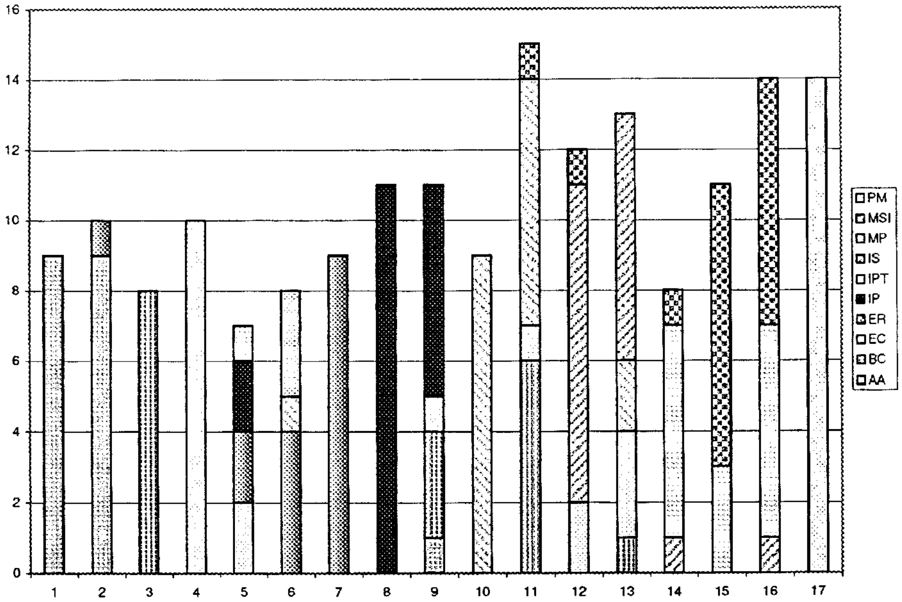


Figure 5. Class distribution of *ARHP* clusters.

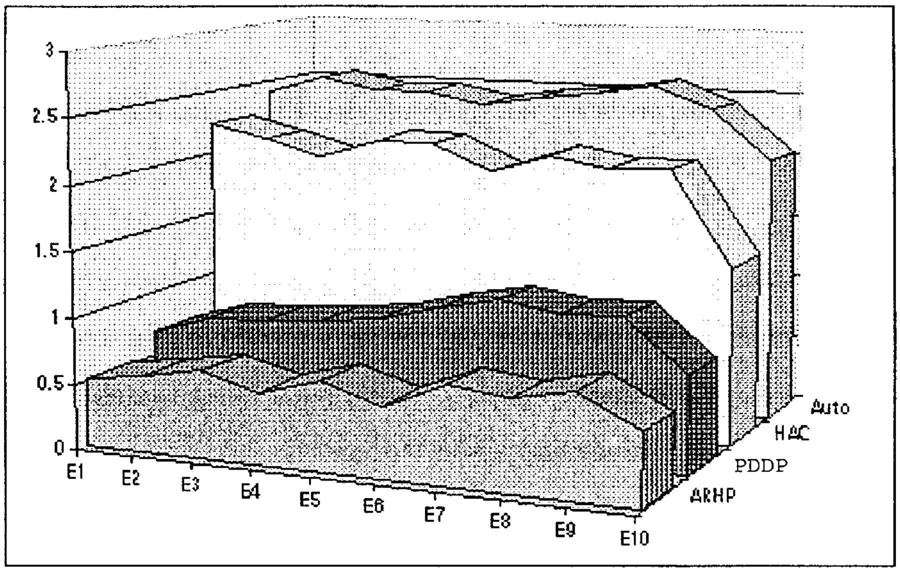


Figure 6. Entropy of different algorithms. Note that lower entropy indicates better cohesiveness of clusters.

It should be noted that the conclusions drawn in the above discussion have been confirmed by another experiment using a totally independent set of documents (Moore et al. 1997).

Dimensionality reduction using LSI/SVD

As noted above, feature selection methods, such as those used in our experiments, are often used in order to reduce the dimensionality of clustering problems in information retrieval domains. The objective of these methods is to extract the most salient features of the document data set to be used as the dimensions in the clustering algorithm. Another approach to dimentionality reduction is that of Latent Semantic Indexing (LSI) (Berry et al. 1995), where the SVD (Berry 1992) is applied once to the original untranslated matrix of word counts, and the first k singular values and associated vectors are retrieved, for some choice of k. This removes much of the noise present in the data, and also yields a representation of the documents of reduced dimensionality.

We compared AutoClass, HAC, and PDDP, after the application of LSI to the complete dataset (E_1). Specifically, we compared the entropies for each method without using LSI, to three LSI datasets with $k = 10$, $k = 50$, $k = 100$. It should be noted that, in the case of HAC, we used LSI on the dataset with the TFIDF scaling in order to be able to compare the results to the original

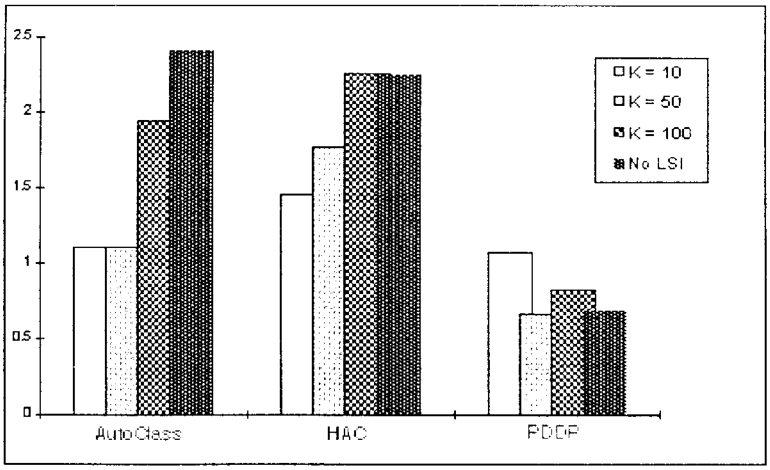


Figure 7. Comparison of Entropies for the E_1, With and Without LSI.

results (without the LSI). However, the TFIDF scaling may not necessarily be the right choice, and a more systematic study with different scaling methods is called for. A summary of our experimental results with LSI is depicted in Figure 7.

This limited set of experiments suggests that LSI can indeed improve the results in the case of AutoClass and HAC (at least for some values of *k*), though not enough to match the entropies in the cases of PDDP or ARHP without dimensionality reduction. In the case of PDDP, LSI did not seem to provide quality improvements to any substantial degree. Furthermore, in all of these cases, the results seem highly dependent on the right choice of *k*. The correct choice for *k* may, in fact, vary from data set to data set, or (as the experiments suggest), from algorithm to algorithm.

Also, both HAC and AutoClass ran dramatically faster on the data with small number of dimensions than on the original data. For the small data set (E1) used for these experiments, the time to run SVD was relatively small. Hence, the overall runtime including SVD computation and clustering improved. However, for a large matrix, the runtime for computing SVD could be quite substantial and may make it impractical to perform dimensionality reduction before clustering.

The precise cost of the SVD computation for both LSI and PDDP depends on many variables, including the number of eigenvalues sought and their distribution, and hence is much beyond the scope of this paper. The underlying Lanczos iteration needed to obtain the $k \gg 1$ eigenvalues in LSI is more or less the same as that used to obtain the single eigenvalue in PDDP, except that in the latter there is no need to deflate eigenvalues as they converge to prevent them from re-appearing. However, at least k, and often at least $2k$, iterations are needed in the LSI case (Berry 1992). On the other hand, obtaining the single leading eigenvector can be accomplished in relatively few iterations, typically around 15 to 20, and never more than 25 in our application. We also note that though the SVD computation in PDDP must be repeated, it is applied to the entire matrix only once.

Finally, we note that we have not run ARHP on the LSI datasets, as ARHP would have required the computation of frequent item sets (documents) based on a very small number of (k) transactions. Frequent item sets discovered using a small number of transactions are not likely to be useful, because they are not statistically significant. Hence, the hypergraph constructed from these frequent item sets does not capture the original relationships present among the documents.

5.2 *Scalability of clustering algorithms*

The scalability of our clustering methods is essential if they are to be practical for large numbers of documents. In this section we give some experimental evidence that our methods are indeed scalable.

We have applied our clustering methods to a large data set denoted “D1” consisting of 2,340 documents using a dictionary of 21,839 words. We then constructed three other data sets labeled D3, D9, D10 using reduced dictionaries using the same strategies as E3, E9, E10, respectively, in Table 1. The number of words in the resulting reduced dictionaries is reported in Figure 9.

Scalability of PDDP

Figure 8 and 9 illustrate the performance of the PDDP algorithm on these datasets. Figure 8 shows that the entropies for D1 and D10 are lower than those for D3 and D9, when the number of clusters agrees. This is consistent with the results shown in Figure 6.

The PDDP algorithm is based on an efficient method for computing the principal direction. The method for computing the principal direction is itself based on the Lanczos method (Golub and Van Loan 1996) in which the major cost arises from matrix-vector products involving the term frequency matrix. The total cost is the cost of each matrix-vector product times the total number of products. The number of products has never exceeded 25 in any of the experiments we have tried. The term frequency matrix is a very sparse matrix: in the D1 data set only about 0.68% of the entries are nonzero. The cost of a matrix-vector product involving a very sparse matrix depends not on the dimensions of the matrix, but rather on the number of nonzero entries. Using the notation of the previous section, we seek the leading singular vector of the dense matrix ($\hat{M} - \mathbf{ce}$), but matrix-vector products involving this matrix can

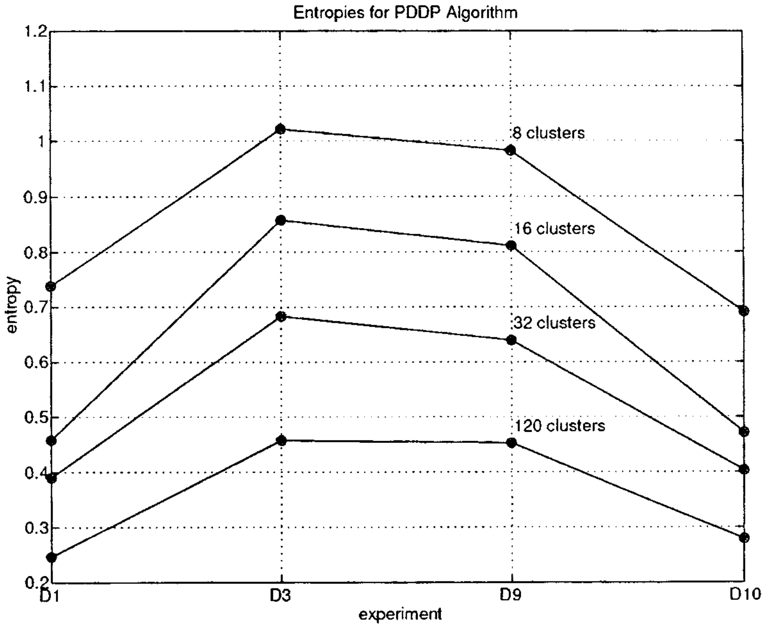


Figure 8. Entropies from the PDDP algorithm with various number of clusters.

be computed as shown in equation (1) without forming this matrix explicitly. This is all that is required for the Lanczos method. Hence the total cost of the PDDP algorithm is governed by the number of nonzero entries in the term frequency matrix. This is illustrated in Figure 9. Notice that even though the data set D10 has many fewer words than the other data sets, its cost is more than for D3 or D9 becauase D10's term frequency matrix is 10 times more dense than D1's matrix: about 6.9% of its entries are nonzero.

Scalability of ARHP

The problem of finding association rules that meet a minimum support criterion has been shown to be linearly scalable with respect to the number of transactions (Agrawal et al. 1996). It has also been shown in Agrawal et al. (1996) that association rule algorithms are scalable with respect to the number of items assuming the average size of transactions is fixed. Highly efficient algorithms such as Apriori are able to quickly find association rules in very large databases provided the support is high enough.

The complexity of HMETIS for a k-way partitioning is $O((V + E)$ where V is the number of vertices and E is the number of edges. The number of vertices in an association-rule hypergraph is the same as the number of documents to be clustered. The number of hyperedges is the same as the number of frequent

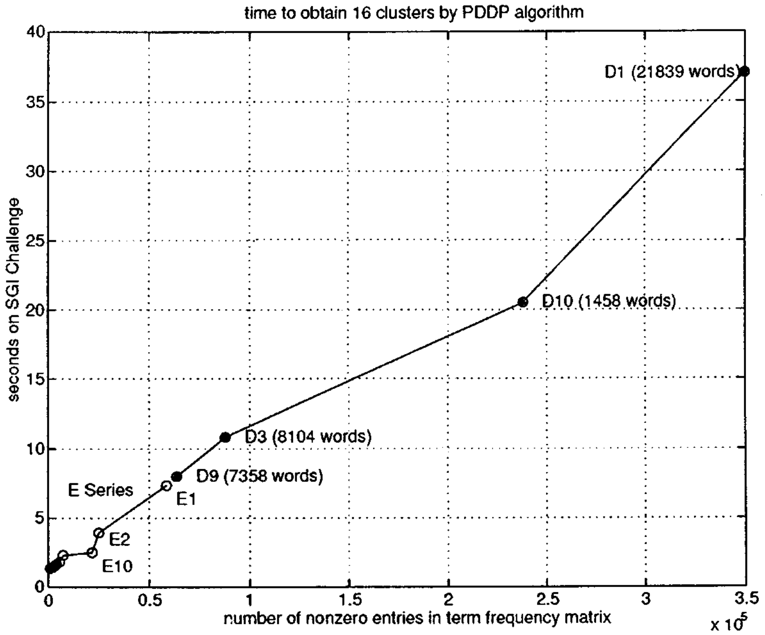


Figure 9. Times for the PDDP algorithm on an SGI versus number of nonzeros in term frequency matrix, for both E and D series with 16 clusters. The D experiments had 2,340 documents and the indicated number of words, and were constructed just like the corresponding E experiments in Table 1.

item-sets with support greater than the specified minimum support. Note that the number of frequent item sets (i.e., hyperedges) does not increase as the number of words increases. Hence, our clustering method is linearly scalable with respect to the number of words in the documents.

Figure 10 shows the entropies from the ARHP algorithm. This result is also consistent with Figure 6 where the entropies for D1 and D10 are lower than for D3 and D9. Each data set produces different size hypergraph. Table 2 shows the size of hypergraph for several experiments from E and D series. Figure 11 shows the CPU time for partitioning these hypergraphs. The run time for partitioning these hypergraphs supports the complexity analysis that says the run time is proportional to the size of the hypergraph ($V + E$).

6. Search for and Categorization of Similar Documents

One of the main tasks of the agent is to search the Web for documents that are related to the clusters of documents. The key question here is how to find a representative set of words that can be used in a Web search. With a single

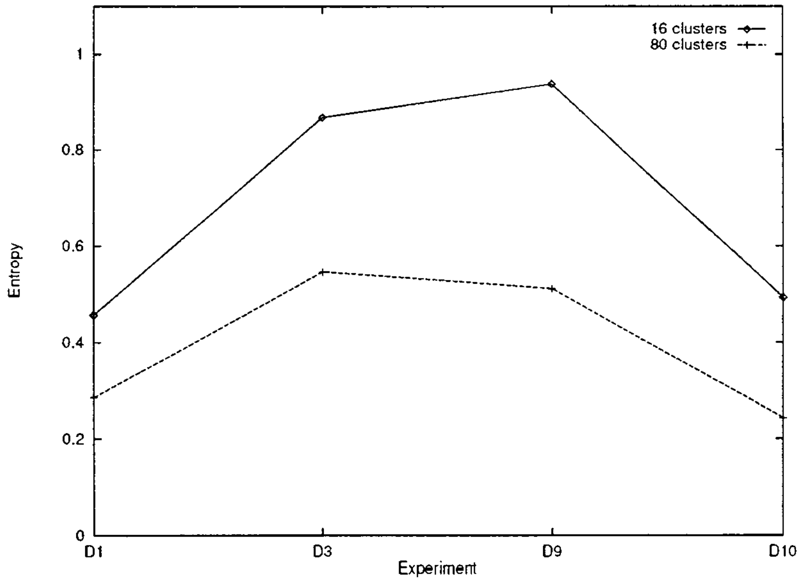


Figure 10. Entropies from the ARHP algorithm with various number of clusters.

document, the words appearing in the document become a representative set. However, this set of words cannot be used directly in a search because it excessively restricts the set of documents to be searched. The logical choice for relaxing the search criteria is to select words that are very frequent in the document.

The characteristic words of a cluster of documents are the ones that have high document frequency and high average text frequency. Document frequency of a word refers to the frequency of the word across documents. Text frequency of a word refers to word frequency within a document. We define the TF word list as the list of k words that have the highest average text frequency and the DF word list as the list of k words that have the highest document frequency.

For each cluster, the word lists TF and DF are constructed. $TF \cap DF$ represents the characteristic set of words for the cluster, as it has the words that are frequent across the document and have high average frequency. The query can be formed as

$$(c_1 \wedge c_2 \ldots \wedge c_m) \wedge (t_1 \vee t_2 \ldots \vee t_n)$$

where $c_i \in TF \cap DF$ and $t_i \in TF - DF$.

We formed queries from the business capital cluster discussed in Section 5. We found the characteristic words of the cluster ($TF \cap DF$) and issued the following query to Yahoo web search engine:

Table 2. Size of hypergraphs for several experiments from E and D series

Experiments	Number of edges	Number of vertices
E1	12091	185
E3	6572	185
E9	4875	185
E10	15203	185
D1	95882	2068
D3	81677	2065
D9	128822	2028
D10	89173	2147

> +capit$*$ +busi$*$ +financ$*$ +provid$*$ +fund$*$ +develop$*$ +compani$*$ +financi$*$ +manag$*$

The search returned 2280 business related documents. We then added the most frequent words that were not in the previous list ($TF - DF$) to form the following query:

> +capit$*$ +busi$*$ +financ$*$ +provid$*$ +fund$*$ +develop$*$ +compani$*$ +financi$*$ +manag$*$ loan$*$ invest$*$ program$*$ credit$*$ industri$*$ tax$*$ increas$*$ cost$*$ technologi$*$ sba$*$ project$*$

Alta Vista search using this query returned only 372 business related documents which seemed highly related to the existing documents in the cluster. First page returned by the query is shown in Figure 12.

The documents returned as the result of queries can be handled in several ways as shown in Figure 1. ARHP could be used to filter out non-relevant documents among the set of documents returned by the query as discussed in Section 4.1. The degree of filtering can be increased either by setting higher support criteria for association rules discovery or by having a tighter connectivity constraint in the partition.

Resulting documents can be incrementally added to the existing clusters using ARHP or PDDP depending on the method used for clustering. With ARHP, for each new document, existing hyperedges are extended to include the new document and their weights are calculated. For each cluster, the connectivity of this new document to the cluster is measured by adding the weights of all the extended hyperedges within the cluster. The new document is placed into the cluster with the highest connectivity. The connectivity ratio between the chosen cluster and the remaining clusters indicates whether the

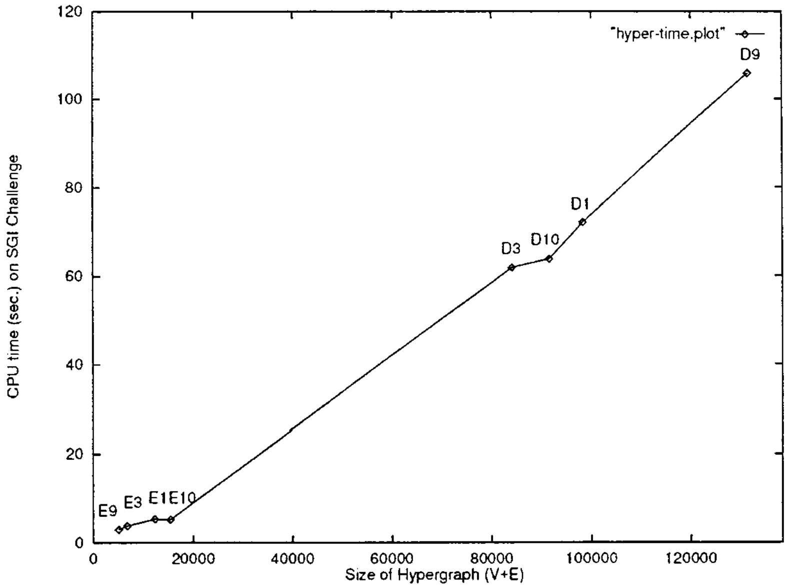


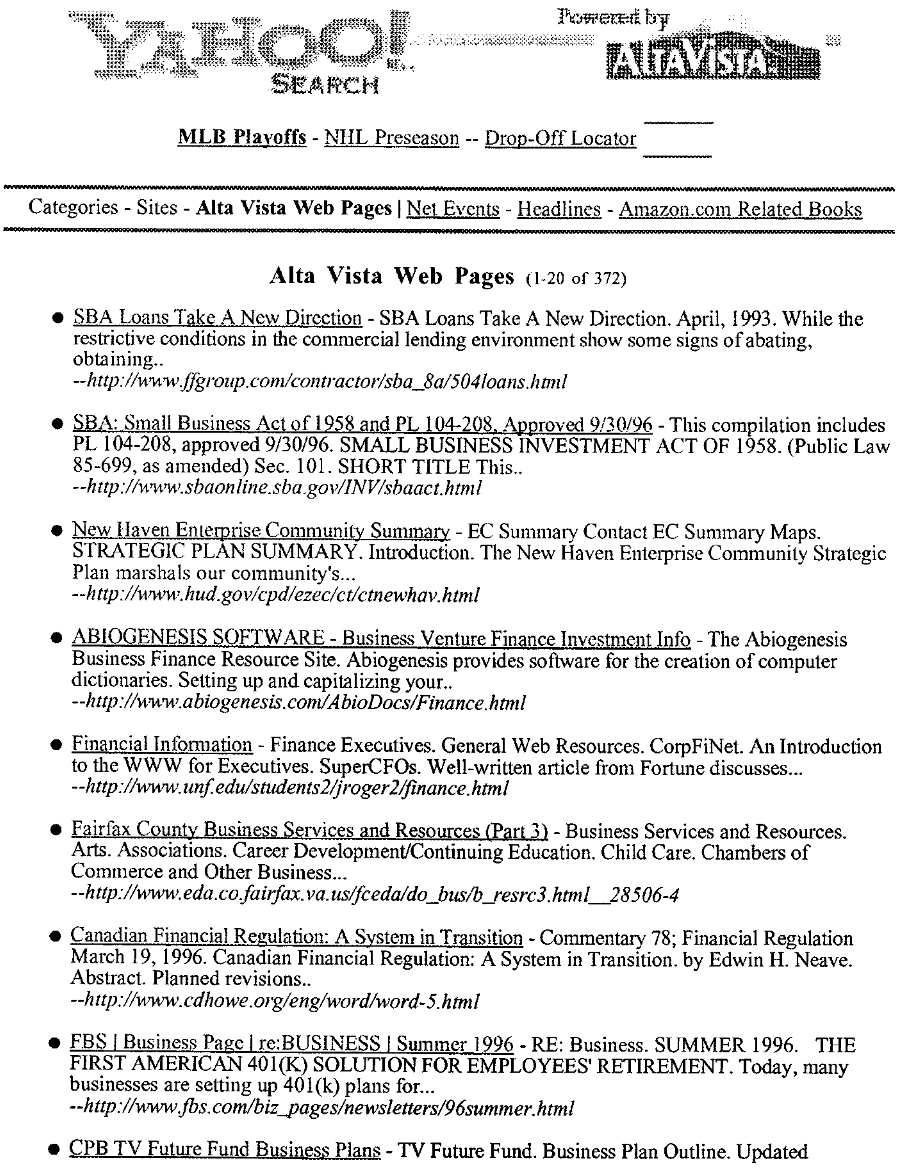
Figure 11. Hypergraph partitioning time for both E and D series. The number of partitions was about 16.

new document strongly belongs to the chosen cluster. If the connectivity of the document is below some threshold for all clusters, then the document can be considered as not belonging to any of the clusters.

With PDDP, the binary tree can also be used to filter new incoming documents by placing the document on one or the other side of the root hyperplane, then placing it on one or the other side of the next appropriate hyperplane, letting it percolate down the tree until it reaches a leaf node. This identifies the cluster in the original tree most closely related to the new incoming document. If the combined scatter value for that cluster with the new document is above a given threshold, then the new document is only loosely related to that cluster, which can then be split in two.

7. Conclusion

In this paper we have proposed an agent to explore the Web, categorizing the results and then using those automatically generated categories to further explore the Web. We have presented sample performance results for the categorization (clustering) component, and given some examples showing how those categories are used to return to the Web for further exploration.



MLB Playoffs - NHL Preseason -- Drop-Off Locator

Categories - Sites - **Alta Vista Web Pages** | Net Events - Headlines - Amazon.com Related Books

Alta Vista Web Pages (1-20 of 372)

- SBA Loans Take A New Direction - SBA Loans Take A New Direction. April, 1993. While the restrictive conditions in the commercial lending environment show some signs of abating, obtaining..
--http://www.ffgroup.com/contractor/sba_8a/504loans.html
- SBA: Small Business Act of 1958 and PL 104-208, Approved 9/30/96 - This compilation includes PL 104-208, approved 9/30/96. SMALL BUSINESS INVESTMENT ACT OF 1958. (Public Law 85-699, as amended) Sec. 101. SHORT TITLE This..
--http://www.sbaonline.sba.gov/INV/sbaact.html
- New Haven Enterprise Community Summary - EC Summary Contact EC Summary Maps. STRATEGIC PLAN SUMMARY. Introduction. The New Haven Enterprise Community Strategic Plan marshals our community's...
--http://www.hud.gov/cpd/ezec/ct/ctnewhav.html
- ABIOGENESIS SOFTWARE - Business Venture Finance Investment Info - The Abiogenesis Business Finance Resource Site. Abiogenesis provides software for the creation of computer dictionaries. Setting up and capitalizing your..
--http://www.abiogenesis.com/AbioDocs/Finance.html
- Financial Information - Finance Executives. General Web Resources. CorpFiNet. An Introduction to the WWW for Executives. SuperCFOs. Well-written article from Fortune discusses...
--http://www.unf.edu/students2/jroger2/finance.html
- Fairfax County Business Services and Resources (Part 3) - Business Services and Resources. Arts. Associations. Career Development/Continuing Education. Child Care. Chambers of Commerce and Other Business...
--http://www.eda.co.fairfax.va.us/fceda/do_bus/b_resrc3.html__28506-4
- Canadian Financial Regulation: A System in Transition - Commentary 78; Financial Regulation March 19, 1996. Canadian Financial Regulation: A System in Transition. by Edwin H. Neave. Abstract. Planned revisions..
--http://www.cdhowe.org/eng/word/word-5.html
- FBS | Business Page | re:BUSINESS | Summer 1996 - RE: Business. SUMMER 1996. THE FIRST AMERICAN 401(K) SOLUTION FOR EMPLOYEES' RETIREMENT. Today, many businesses are setting up 401(k) plans for...
--http://www.fbs.com/biz_pages/newsletters/96summer.html
- CPB TV Future Fund Business Plans - TV Future Fund. Business Plan Outline. Updated

Figure 12. Search results from Yahoo.

For the categorization component, our experiments have shown that the ARHP algorithm and the PDDP algorithm are capable of extracting higher quality clusters while operating much faster compared to more classical algorithms such as HAC or AutoClass. This is consistent with our previous results (Moore et al. 1997). The ARHP algorithm is also capable of filtering out documents by setting a support threshold. Our experiments show that the PDDP and ARHP algorithms are fast and scale with the number of words in the documents.

To search for similar documents keyword queries are formed by extending the characteristic word sets for each cluster. Our experiments show that this method is capable of producing small sets of relevant documents using standard search engines.

In the future, we will explore the performance of the entire agent as an integrated and fully automated system, comparing the relative merits of the various algorithms for clustering, query generation, and document filtering, when used as the key components for this agent. In particular, we will conduct further experimental evaluation of our query generation mechanism and classification of new documents into existing clusters. Another area for future work involves the development of a method for evaluating quality of clusters which is not based on *a priori* class labels.

References

Ackerman L. M. et al. (1997). Learning Probabilistic User Profiles. *AI Magazine* **18**(2): 47–56.

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. & Verkamo, A. I. (1996). Fast Discovery of Association Rules. In Fayyad, U.M. Piatetsky-Shapiro, G., Smyth, P. & Uthurusamy, R. (eds.) *Advances in Knowledge Discovery and Data Mining*, 307–328. AAAI/MIT Press.

Anderson, T. W. (1954). On Estimation of Parameters in Latent Structure Analysis. *Psychometrika* **19**: 1–10.

Armstrong, R. Freitag, D., Joachims, T. & Mitchell, T. (1995). WebWatcher: A Learning Apprentice for the World Wide Web. In *Proc. AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*. AAAI Press.

Balabanovic, M., Shoham, G. & Yun, Y. (1995). An Adaptive Agent for Automated Web Browsing. *Journal of Visual Communication and Image Representation* **6**(4).

Berge, L. C. (1976). *Graphs and Hypergraphs*. American Elsevier.

Berry, M. W. (1992). Large-Scale Sparse Singular Value Computations. *International Journal of Supercomputer Applications* **6**(1): 13–49.

Berry, M. W., Dumais, S. T. & O'Brien, G. W. (1995). Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review* **37**: 573–595.

Boley, D. L. (1997). *Principal Direction Divisive Partitioning*. Technical Report TR-97-056, Department of Computer Science, University of Minnesota, Minneapolis.

Cheeseman, L. & Stutz, J. (1996). Bayesian Classification (Autoclass): Theory and Results. In Fayyad, U. M., Piatesky-Shapiro, G., Smyth, P. & Uthurusamy, R. (eds.) *Advances in Knowledge Discovery and Data Mining*, 153–180. AAAI/MIT Press.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. & Harshman, R. (1990). Indexing by Latent Semantic Analysis. *J. Amer. Soc. Inform. Sci.* **41**: 41.

Doorenbos, R. B., Etzioni, O. & Weld, D. S. (1996). *A Scalable Comparison Shopping Agent for the World Wide Web*. Technical Report 96-01-03, University of Washington, Dept. of Computer Science and Engineering.

Duda, R. O. & Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. John Wiley & Sons.

Frakes, W. B. (1992). Stemming Algorithms. In Frakes, W. B. & Baeza-Yates, R. (eds.) *Information Retrieval Data Structures and Algorithms*, 131–160. Prentice Hall.

Frakes, W. B. & Baeza-Yates, R. (1992). *Information Retrieval Data Structures and Algorithms*. Prentice Hall: Englewood Cliffs, NJ.

Golub, G. H. & Van Loan, C. F. (1996). *Matrix Computations*, 3rd edn. Johns Hopkins Univ. Press.

Hammond, K., Burke, R., Martin C. & Lytinen, S. (1995). FAQ-Finder: A Case-Based Approach to Knowledge Navigation. In *Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments*. AAAI Press.

Han, E. H., Karypis, G., Kumar, V. & Mobasher, B. (1997a). Clustering Based on Association Rule Hypergraphs (Position Paper). In *Workshop on Research Issues on Data Mining and Knowledge Discovery*, 9–13. Tucson, Arizona.

Han, E. H., Karypis, G., Kumar, V. & Mobasher, B. (1997b). *Clustering in a High-Dimensional Space Using Hypergraph Models*. Technical Report TR-97-063, Department of Computer Science, University of Minnesota, Minneapolis.

Han, E. H., Karypis, G., Kumar, V. & Mobasher, B. (1998). Hypergraph Based Clustering in High-Dimensional Data Sets: A Summary of Results. *Bulletin of the Technical Committee on Data Engineering* **21**(1).

Jackson, J. E. (1991). *A User's Guide to Principal Components*. John Wiley & Sons.

Jain A. K. & Dubes, R. C. (1988). *Algorithms for Clustering Data*. Prentice Hall.

Karypis, G., Aggarwal, R., Kumar V. & Shekhar, S. (1997). Multilevel Hypergraph Partitioning: Application in VLSI Domain. In *Proceedings ACM/IEEE Design Automation Conference*.

Kirk, T., Levy, A. Y., Sagiv, Y. & Srivastava, D. (1995). The Information Manifold. In *Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments*. AAAI Press.

Kohonen, T. (1988). *Self-Organization and Association Memory*. Springer-Verlag.

Kwok, C. & Weld, D. (1996). Planning to Gather Information. In *Proc. 14th National Conference on AI*.

Leighton, V. H. & Srivastava, J. (1997). *Precision Among WWW Search Services (Search Engines): Alta Vista, Excite, Hotbot, Infoseek, Lycos*. http://www,winona,msus.edu/is-f/library-f/webind2/webind2.htm.

Lu, S. Y. & Fu, K. S. (1978). A Sentence-to-Sentence Clustering Procedure for Pattern Analysis. *IEEE Transactions on Systems, Man and Cybernetics* **8**: 381–389.

Maarek, Y. S. & Shaul, I. Z. Ben (1996). Automatically Organizing Bookmarks per Content. In *Proc. of 5th International World Wide Web Conference*.

Moore, J., Han, E., Boley, D., Gini, M., Gross, R., Hastings, K., Karypis, G., Kumar, V. & Mobasher, B. (1997). Web Page Categorization and Feature Selection Using Association Rule and Principal Component Clustering. In *7th Workshop on Information Technologies and Systems*.

Perkowitz, M. & Etzioni, O. (1995). Category Translation: Learning to Understand Information on the Internet. In *Proc. 15th International Joint Conference on AI*, pp. 930–936. Montreal, Canada.

Porter, M. F. An Algorithm for Suffix Stripping. *Program* **14**(3): 130–137.

Salton, G. & McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill.

Titterington, D. M., Smith, A. F. M. & Makov, U. E. (1985). *Statistical Analysis of Finite Mixture Distributions*. John Wiley & Sons.

Weiss, R., Velez, B., Sheldon, M. A., Nemprempre, C., Szilagyi, P., Duda, A. & Gifford, D. K. (1996). Hypursuit: A Hierarchical Network Search Engine that Exploits Content-Link Hypertext Clustering. In *Seventh ACM Conference on Hypertext*.

Wulfekuhler, M. R. & Punch, W. F. (1997). Finding Salient Features for Personal Web Page Categories. In *Proc of 6th International World Wide Web Conference*.