WebACE: A Web Agent for Document Categorization and Exploration

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

Abstract

We propose an agent for exploring and categorizing documents on the World Wide Web. The heart of the agent is an automatic categorization of a set of documents, combined with a process for generating new queries used to search for new related documents and filtering the resulting documents to extract the set of documents 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 traditional clustering algorithms and form the basis for the query generation and search component of the agent.

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. Relying on large indexes to documents located on the Web, search engines determine the URLs of those documents satisfying a user's query. Often queries return inconsistent search 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 semantic features from the words or structure of an HTML document. These extracted features are then employed 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 WebACE, an agent for document categorization and exploration that operates on Web documents. A novel part of the paper is the description of two new clustering algorithms based on graph partitioning, that provide a significant improvement in performance over traditional clustering algorithms used in information retrieval. Many traditional algorithms break down as the size of the document space, and thus the dimensionality of the corresponding feature space, increases. High dimensionality is characteristic of the type of information retrieval applications which are used to filter and categorize hypertext documents on the World Wide Web. In contrast, our partitioning-based algorithms do not rely on a specific choice of a distance function and do scale up effectively in a high dimensional space.

After a short description of the architecture of WebACE in Section 3, we describe the clustering algorithms in Section 4. The results obtained on a number of experiments using different methods to select sets of features from the documents show that partitioning clustering methods perform better than traditional distance based clustering. In Section 5 we show how to use words obtained from clusters of documents to generate queries for related documents.

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.

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 [14], Information Manifold [18], and OCCAM [19] 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,

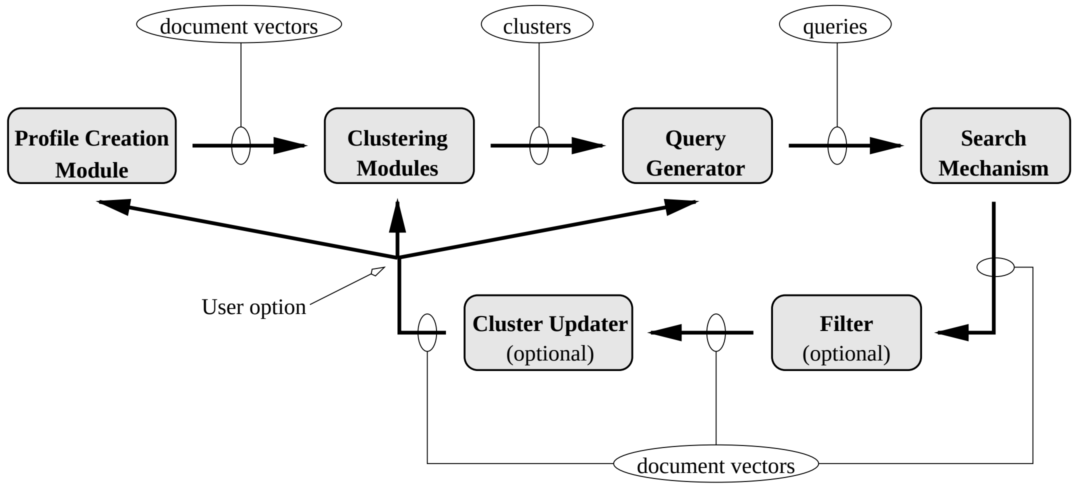

Figure 1: WebACE Architecture

such as ShopBot [9] and ILA [23], attempt to interact with and learn the structure of unfamiliar information sources.

Information Filtering/Categorization A number of Web agents use various information retrieval techniques [12] and characteristics of open hypertext Web documents to automatically retrieve, filter, and categorize. For example, HyPursuit [27] uses semantic information embedded in link structures as well as document content to create cluster hierarchies of hypertext documents. BO (Bookmark Organizer) [21] 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 [28] to discover salient HTML document features (words) that can be used in finding similar HTML documents on the Web.

Personalized Web Agents Another group 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 [3], Syskill & Webert, and others. Syskill & Webert [1] utilizes a user profile and learns to rate Web pages of interest using a Bayesian classifier. Balabanovic [4] 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.

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 [1, 3, 4]. 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 traditional algorithms. These are described in Section 4.

After WebACE has found document clusters, it uses 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 documents to his profile. The query generation methods and the algorithms for incrementally updating existing clusters are discussed in Section 5.

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 [12]). Distance-based methods such as k-means analysis, hierarchical clustering [16] and nearest-neighbor clustering [20] 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 [25] 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. If the dimensionality is high, then the calculated mean values 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 [8], 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.

Our proposed clustering algorithms which are described in this section are designed to efficiently handle very high dimensional spaces, and furthermore, they do not require the definition of ad hoc distance or similarity metrics. 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 compare these algorithms to two well-known methods: Bayesian classification as used by AutoClass [8] and *hierarchical agglomeration clustering (HAC)* based on the use of a distance function [10].

AutoClass is based on the probabilistic mixture modeling [26], and given a data set it finds maximum parameter values for a specific probability distribution functions of the clusters. The clustering results provide the full description of each cluster in terms of probability distribution of each attributes.

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 is usually weighted using the TFIDF function, 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

In [15], a new method was proposed for clustering related items in transaction-based databases, such as supermarket bar code data, using association rules and hypergraph partitioning. This method first finds set of items that occur frequently together in transactions using association rule discovery methods [2]. These frequent item sets are then used to group items into hypergraph edges, and a hypergraph partitioning algorithm [17] is used to find the item clusters. The similarity among items is captured implicitly by the frequent item sets.

In document clustering, each document corresponds to an item and each possible feature corresponds to a transaction. 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 in common. These frequent item sets are mapped into hyperedges in a hypergraph. 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 this model, the set of vertices V corresponds to the documents, and each hyperedge $e \in E$ corresponds to a set of related documents found. For example, if $\{d_1, d_2, d_3\}$ is a frequent item set, then the hypergraph contains a hyperedge that connects d_1, d_2 and d_3. The weight of a hyperedge is calculated as the average confidence [2] of all the association rules involving the related documents of the hyperedge. The confidence of an association rule involving documents like $\{d_1, d_2\} \Longrightarrow \{d_3\}$ is the conditional probability that a feature occurs in document d_3 whenever it occurs in d_1 and d_2.

Next, a hypergraph partitioning algorithm is used to partition the hypergraph 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 partitioning the documents into different clusters. Similarly, this method can be applied to word clustering. In this setting, each word corresponds to an item and each document corresponds to a transaction.

This method uses the *Apriori* algorithm [2] which has been shown to be very efficient in finding frequent item sets and HMETIS [17] which can partition very large hypergraphs (of size $>$ 100K nodes) in minutes on personal computers.

An additional advantage of ARHP is that it can be used to filter out non-relevant documents while clustering a document space, and thus improving the quality of the document clusters. This filtering capability is mainly due to 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 Component Divisive Partitioning

The method of Principal Component Divisive Partitioning [7] is a top down clustering method. Starting with a "root" cluster encompassing the entire document set, each unsplit cluster is split into two "child" clusters until a desired number of clusters is reached. The clusters formed in this way are in a hierarchy which has the form of a binary tree whose root is the initial "root" cluster. The leaf nodes of this tree are the unsplit clusters. At termination, these unsplit clusters are the final set of clusters returned by this algorithm. The algorithm proceeds by repeatedly selecting a leaf node in the binary tree and splitting that node, generating two subclusters which become the two children to the node just split.

The behavior of the algorithm is controlled by the method used to select the next node to split, as well as how that cluster is split. We discuss the latter process first, since this is the key to the computational efficiency of this method. Each document is represented by a vector $\mathbf{d}$ of word frequencies,

Word Set	Selection Criteria	Dataset Size	Comments
E1	All words	185x10536	We select all non-stop words (stemmed).
E2	All words with text frequency > 1	188x5106	We prune the words selected for E11 to exclude those occurring only once.
E3	Top 20+ words	185x1763	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	185x1328	We prune the words selected for E3 to exclude those occurring only once.
E5	Top 15+ with text frequency > 1	185x1105	
E6	Top 10+ with text frequency > 1	185x805	
E7	Top 5+ with text frequency > 1	185x474	
E8	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.
E9	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.
E10	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.

Table 1: Setup of experiments.

scaled to have unit length to make the results independent of document length. A cluster is represented by a set of documents, $D = \{\mathbf{d}_1, \ldots, \mathbf{d}_k\}$. We define a linear discriminant function $g(\mathbf{d}) = \mathbf{u}^T(\mathbf{d} - \mathbf{w})$, where $\mathbf{u}$, $\mathbf{w}$ are vectors to be determined. The linear discriminant function is used to define the splitting of the cluster: if $g(\mathbf{d}) \leq 0$, the document $\mathbf{d}$ is placed in the new left child, otherwise $\mathbf{d}$ is placed in the new right child. Thus the behavior of each node in the binary tree is determined entirely by the two vectors $\mathbf{u}$, $\mathbf{w}$.

Space does not permit a complete description of how $\mathbf{u}$, $\mathbf{w}$ are computed, but we can briefly describe what those vectors are mathematically. The vector $\mathbf{w}$ is defined to be the *mean* or *centroid* vector for the cluster. The vector $\mathbf{u}$ is the direction of maximal variance. This direction is defined to be the eigenvector corresponding to the largest eigenvalue of the covariance matrix of the documents in the cluster. The computation of $\mathbf{u}$ is the most expensive step in this whole process. We have used a fast Lanczos-based solver for the singular values of the matrix of documents in the cluster [13]. This algorithm is able to take full advantage of the fact that less than 4% of all the entries in the document vectors are nonzero. We remark that we did not use TFIDF scaling [25] because (a) the quality of the PDDP algorithm results was not noticeably different, and (b) TFIDF fills in all the zero entries with nonzero values, substantially increasing the costs.

The method as a whole allows the user to choose any convenient method for deciding which cluster to split next at each stage. In our experiments we have adopted a *scatter* value, defined to be the sum of squares of the distances from each document in the cluster to the cluster centroid. The cluster with the largest such scatter value is selected next.

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.

4.3 Experimental Evaluation of Clustering

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 [24] as implemented by [11]. We derived 10 experiments (according to the method used for feature selection)

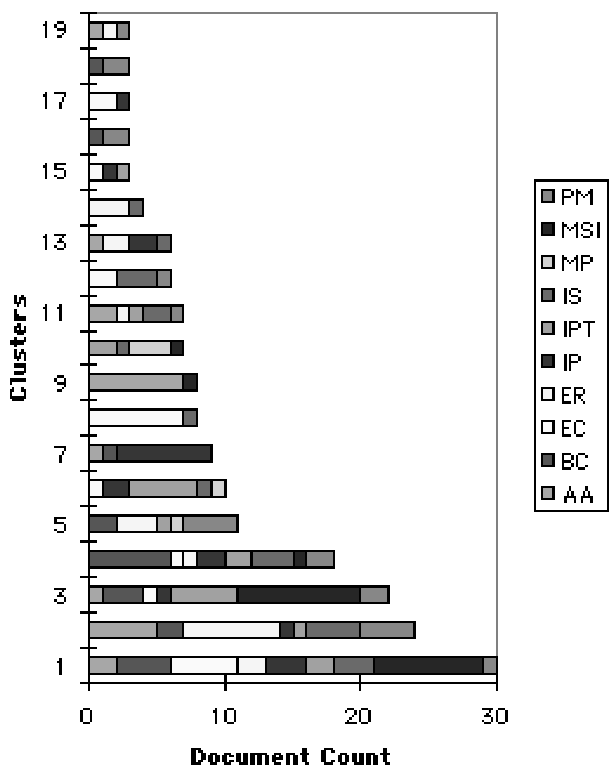

Figure 2: Class distribution of AutoClass clusters.

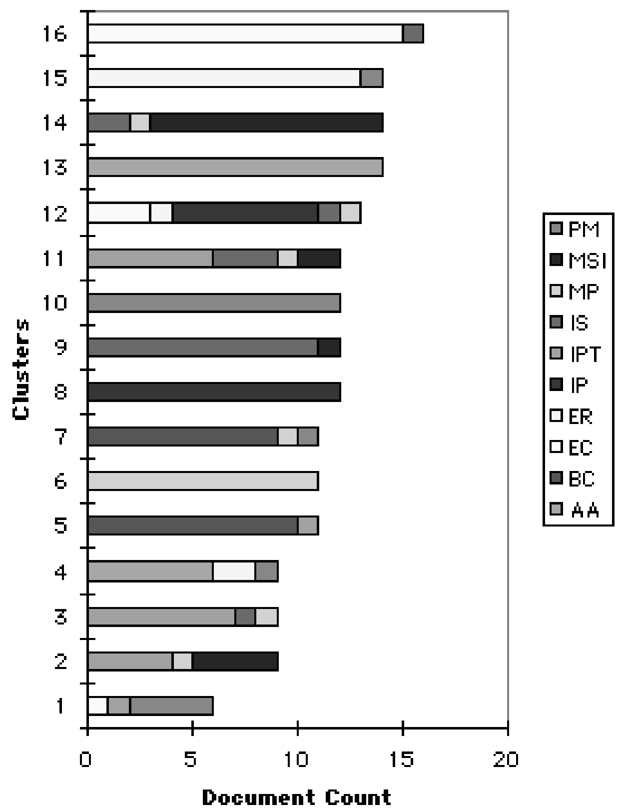

Figure 3: Class distribution of *PDDP* clusters.

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 retain 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 class distribution of documents is calculated first. Then using this class distribution, the entropy of each cluster is calculated. 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. 5). Figure 2 shows the class distribution of documents in each cluster of the best AutoClass result with the entropy value 2.05. Comparing this result to one of PDDP result with entropy value of 0.69 in Figure 3 and one of ARHP result with entropy value of 0.79 in Figure 4, we can see the big differences in the quality of the clusters obtained from these experiments.

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 depend in a choice of a distance function; (2) they do not require calculation of the mean of the clusters, and so the issue of having cluster means very close in space does not apply; (3) they are not sensitive to the dimensionality of the data sets; and (4) 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 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 $\times$ 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 losing 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.

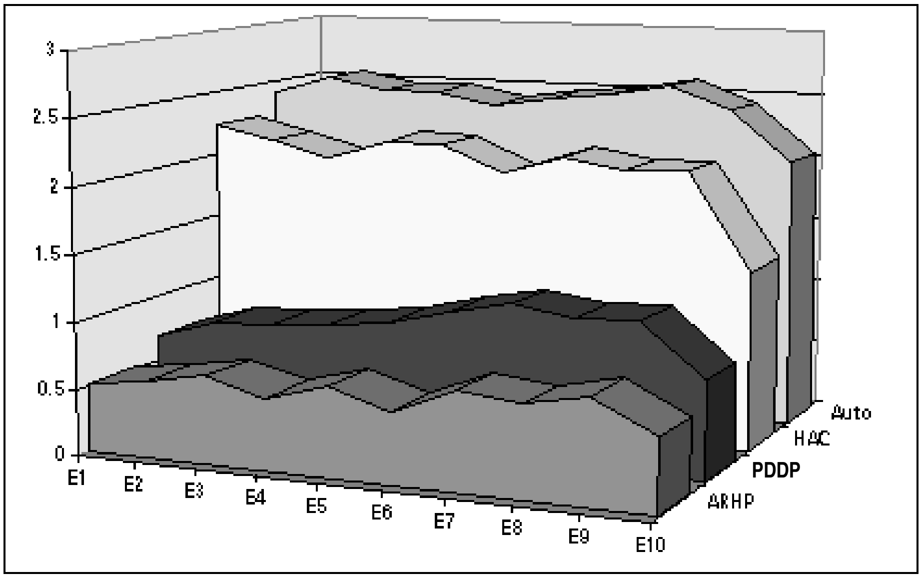

Figure 5: Entropy of different algorithms. Note that lower entropy indicates better cohesiveness of clusters.

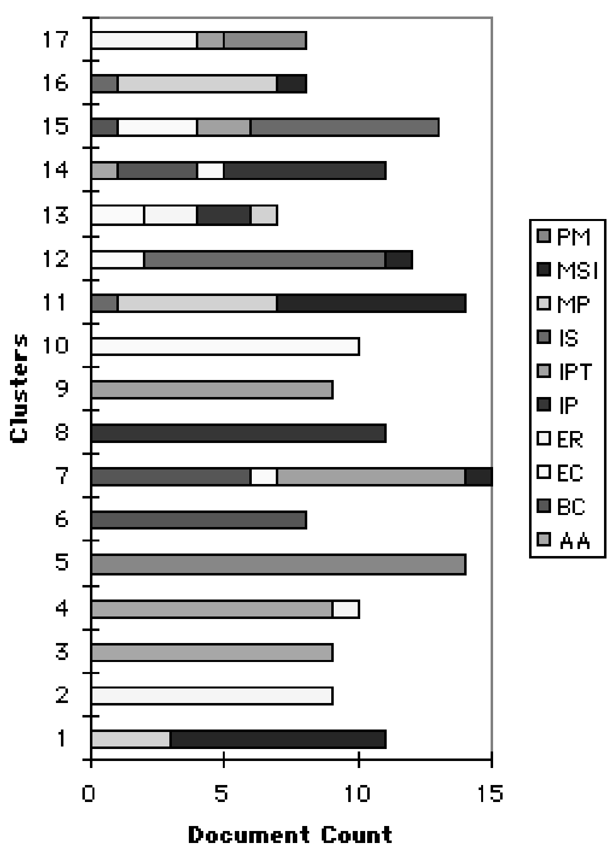

Figure 4: Class distribution of *ARHP* clusters.

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 [22].

5 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 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 4.3. We found the characteristic words of the cluster ($TF \cap DF$) and issued the following query to Yahoo web search engine:

```
+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*
```

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 6: First page of search results from Alta Vista (without the graphics).

```
+compani* +financi* +manag* loan* invest* program*
credit* industri* tax* increas* cost* technologi*
sba* project*
```

AltaVista 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 6.

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 documented 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 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 cluster.

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 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.

6 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.

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 classical algorithms such as HAC or AutoClass. This is consistent with our previous results [22]. The ARHP algorithm is also capable of filtering out documents by setting a support threshold.

To search for similar documents 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.

References

[1] M. Ackerman 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. Armstrong, D. Freitag, T. Joachims, and T. Mitchell. WebWatcher: A learning apprentice for the world wide web. In *Proc. AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*. AAAI Press, 1995.

[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.

[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] P. Cheeseman and J. Stutz. Bayesian 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.

[9] R. B. Doorenbos, O. Etzioni, and D. S. Weld. A scalable comparison shopping agent for the World Wide Web. Technical Report 96-01-03, University of Washington, Dept. of Computer Science and Engineering, 1996.

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

[11] 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.

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

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

[14] K. Hammond, R. Burke, C. Martin, and S. Lytinen. 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, 1995.

[15] 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.

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

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

[18] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The information manifold. In *Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments*. AAAI Press, 1995.

[19] C. Kwok and D. Weld. Planning to gather information. In *Proc. 14th National Conference on AI*, pages 32–39, 1996.

[20] 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.

[21] Y. S. Maarek and I.Z. Ben Shaul. Automatically organizing bookmarks per content. In *Proc. of 5th International World Wide Web Conference*, 1996.

[22] 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.

[23] M. Perkowitz and O. Etzioni. Category translation: learning to understand information on the internet. In *Proc. 15th International Joint Conference on AI*, pages 930–936, Montral, Canada, 1995.

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

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

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

[27] 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.

[28] 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.