

Fast Clustering of Web Users Based on Browsing Patterns

Yongjian Fu Kanwalpreet Sandhu Ming-Yi Shih
Computer Science Department
University of Missouri-Rolla
{yongjian,ksandhu,mingyi}@umr.edu

ABSTRACT

We propose the clustering of the Web users based on patterns of their browsing activities on the Web. The browsing pattern of a Web user consists of the pages the user visited and the times spent on them. Such patterns are extracted from a Web server's log, and then organized into sessions which represent units of interaction between Web users and the Web server. A generalization-based clustering method is employed which first generalizes the sessions and then applies a hierarchical clustering algorithm to find clusters in the generalized sessions. Our experiments on a large data set show that the approach is fast and effective for the clustering of Web users.

Keywords: Data Mining, Web Mining, Clustering, Browsing Patterns, Data Generalization, Dimension Reduction.

1 INTRODUCTION

The growing popularity of the World Wide Web, or the Web, has changed the way people work, study, communicate, and do business. Many organizations have set up their home pages to provide Web-based facilities such as on-line shopping, technical support, virtual classroom, to name a few. Web mining, the knowledge discovery from the Web, has become an important research area [4].

Research in Web mining can be broadly classified into three categories: resource discovery, knowledge discovery in the documents, and knowledge discovery in users. Resource discovery helps users to locate the documents by intelligent search. Knowledge discovery in the documents is the study to find interesting characteristics and patterns about Web documents. Knowledge discovery in users tries to find interesting characteristics and patterns of the Web users and their usage of the Web.

An interesting topic in Web mining is the analysis of the users of the Web. To make their Web site more effective, Web site administrators or webmasters needs to understand the users of the Web and

their usages. Because of the number and the diversity of the users of the Web, it is obviously unrealistic to study each individual user. Therefore, an important step in the user analysis is to cluster the users into some groups or classes, based on their common properties. The webmaster then may analyze the characteristic of the groups and provide suitable services for them.

In this paper, we will study the clustering of the Web users based on their browsing activities or patterns on the Web. Users with similar browsing patterns are clustered into classes (clusters). For example, if a number of customers spend quite a lot time on browsing pages about “baby food”, “maternity garment”, and “kids wear”, they may be clustered into a group which could later be analyzed by webmasters or domain experts as “expecting parents”. The webmaster may, for example, arrange the Web pages so that the above pages are interlinked together. Also, when a user has browsed the “baby foods” and the “maternity garments” pages, a link to “kids wear” page can be dynamically created and inserted in the current page [18].

To cluster the users, the Web server's log data is first processed to extract sessions, which are basic units of interaction between the Web users and the Web server. A generalization-based clustering method is employed to cluster the sessions, in which the sessions are generalized using the attribute-oriented induction and clustered using a hierarchical clustering algorithm. The approach is tested on a real large data set and our experiments show encouraging results.

This paper is organized as follows. In Section 2, the background and related work in clustering and Web user clustering are introduced. The extraction of sessions from Web server log is discussed in Section 3. Our approach for clustering Web users based on their browsing patterns is presented in Section 4. In Section 5, the experiments on a real large data set are reported. The study is concluded in Section 6, along with some future work.

2 BACKGROUND AND RELATED WORK

Earlier studies on Web usages, such as access statistics, lack the in-depth understanding of user browsing patterns, such as pages visited and time spent on each page. These user browsing patterns provide accurate, active, and objective information about the Web usages of the users. Moreover, most Web servers, e.g., NCSA's HTTPD [6] and Microsoft's IIS [9], contain such information in their log of page requests. It is interesting to study the clustering of the Web users based on their browsing activities extracted from the Web server's log files.

A Web server's log file will contain records of users' requests of pages. A typical record contains the client's IP address, the date and time, the URL of the page, the protocol, the return code of server, and the size of the page if the request is successful. A few samples are given below which are excerpted from the log of the University of Missouri-Rolla's (UMR) Web server, which runs HTTPD 1.0. The IP addresses are modified for privacy reasons. The URLs of the pages are relative to the UMR's home page address, `http://www.UMR.edu`.

```
smith.cs.UMR.edu [01/Apr/1997:00:03:24 -0600]
"GET /~amigos/Virtual/flame.jpg HTTP/1.0" 200 3388
ultra1.cs.UMR.edu [01/Apr/1997:00:03:24 -0600]
"GET /~cgiwrap/htdocs/notes.html HTTP/1.0" 200 586
cc1.student.UMR.edu [01/Apr/1997:00:03:24 -0600]
"GET /~hungwen/ HTTP/1.0" 200 3314
me211.student.UMR.edu [01/Apr/1997:00:03:25 -0600]
"GET /images/UMR/UMR125sm.gif HTTP/1.0" 200 4967
```

From such a Web server's log file, user browsing patterns can be extracted. A user's browsing pattern consists of the pages she visited and the time she spent on each page. Each user can then be represented by a set of (page-id, time) pairs. The details of the process and the issues involved are discussed in Section 3.1.

2.1 Clustering Algorithms

Given a set of objects, each represented by a vector, a clustering algorithm groups them into some classes or clusters. The clustering algorithm searches for an optimal set of clusters based on certain cluster quality or error measure. Clustering has been studied extensively in statistics, machine learning, pattern recognition, and database systems, with a large number of clustering algorithms developed [10, 11, 12, 1, 20].

The two common approaches in statistical clustering are partitioning clustering and hierarchical clustering [10, 11]. The former finds a single, flat partition of the objects by iteratively adjusting the clusters to minimize the square error. Starting with an initial

partition (seeds) the rest of the objects are assigned to the cluster whose centroid is closest to the object based on a predefined distance or similarity measure. After the objects have been assigned, the centroids of the clusters are recomputed and the process repeats.

Hierarchical clustering methods can be further classified into agglomerative methods and divisive methods where the former starts with single object clusters and consecutively merges the clusters to form higher level clusters and the latter starts with a single cluster consisting of all objects and iteratively splits the cluster(s) into smaller clusters.

A set of clustering algorithms, called conceptual clustering algorithms [5], have been proposed in machine learning to deal with nominal or categorical data. They extend statistical methods by developing a logical description, such as first order predicates, for clusters and objects. The optimal clusters are searched in the conceptual space defined by the descriptions.

Based on fuzzy sets, fuzzy clustering algorithms have been developed in pattern recognition [1], which can find fuzzy clusters, i.e., an object can partially belong to a cluster. In other words, each cluster is a fuzzy set and the membership of an object to a cluster is defined by a membership function between 0 and 1.

Most previous methods have a time complexity of $O(n^3)$ where n is the number of objects. Therefore, the methods do not scale up well for large data sets. Two algorithms developed in database area, CLARANS[15] and DBSCAN[3], manage to reduce the time complexity to $O(n^2 * k)$ where k is the number of clusters, using random searching and sampling. However, this is still less than desirable for very large data sets. Besides, like most other methods, they assume that the whole data set can be fitted into main memory, which is a serious limitation for very large data sets, as in our case of Web users.

Recently, an algorithm, Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH), has been proposed in database area, [20] which tries to minimize both CPU time and disk I/O. It usually creates a good clustering in just one scan of the data set and the complexity of the algorithm is $O(n \log(n))$, which make it especially attractive for very large data sets.

2.2 Clustering of Web Users

The clustering of the Web users based on their browsing patterns has been studied by Yan et al. [18] and Shahabi et al. [16]. In [18], a page space is constructed by treating each page as a dimension and a browsing session of a user is mapped into a point (vector) in

the space, represented by the time the user spent on each page. However, the clustering algorithm they used, the *leader algorithm*, is sensitive to the object input order and the quality of the final clusters is unpredictable.

An applet-based user profiler is proposed in [16] which can get more accurate times users spent on pages. Using the *k*-means method, the users are clustered based on their navigation paths, i.e., the pages as well as the orders they are requested and the links the users follow. However, the advantages of using paths are not convincing and the computation is much more complex. Besides, the *k*-means method works in a batch mode whereas the data are generated intermittently.

Furthermore, both use partitioning clustering algorithms which will generate a set of flat clusters. When the number of users are large, the algorithm will generate either too many clusters or very large clusters (with many members). A more appropriate approach is the hierarchical clustering.

In our study, the BIRCH algorithm [20] is chosen as the clustering algorithm because it is fast, incremental, and hierarchical. However, the direct application of BIRCH on the user browsing data as described above is very inefficient and may not find interesting clusters.

A Web server usually contains thousands, even millions of pages. If each user is represented by a vector of time on pages, the dimensionality of the vector will be huge. Sparse representation, such as the one mentioned before, may be used, but will impose a lot of overheads on memory and storage management because the dimensionality of centroids of clusters change dynamically. Moreover, clustering is to find groups with similar browsing patterns which does not necessarily correspond to page level. Groups that are not obvious at page-level may emerge when considered at a higher level.

Based on the above observations, an efficient clustering approach is proposed in this paper. The approach employs the generalization-based clustering method which integrates the attributed-oriented induction method [7] with BIRCH [20], to cluster Web users based on their browsing patterns. The server log data is first processed to identify sessions of Web usages. The sessions are then generalized using the attribute-oriented induction method. The generalized sessions are finally clustered using BIRCH. The approach is tested on real log data from the University of Missouri-Rolla's Web server. Our experiments show that the approach is efficient and it has found several interesting clusters within the data set.

3 SESSION EXTRACTION

To deal with the unpredictable nature of Web browsing and make the problem tractable, a concept, *session*, is introduced as the unit of interaction between a user and a Web server. A session consists of pages browsed by a user within a certain amount of time. Clusters are found in the sessions instead of the users' entire histories. Concepts similar to session have also been proposed in other studies [14, 13, 16, 2]. In this paper, we will use session and user interchangeably.

The sessions can be extracted by grouping the subsequent pages requested by the same user together. A session is represented by a session identifier and a set of page/time pairs. For example, a session $(sid_0, p_0, 10, p_1, 30, p_2, 20)$ tells a user spent 10 seconds on page p_0 , 30 seconds on page p_1 , and 20 seconds on page p_2 . The data in a Web server log is transformed into a set of sessions. Several factors and issues which affect the process are addressed below.

- Most Web servers only record the IP addresses of the client's computer, which can be shared by more than one users. This is not a concern for our purpose because each session usually reflects an interaction of a single user although different sessions may be contributed by different users.
- If the elapse of time between two consecutive requests from the same user exceeds a *max_idle_time*, *max_idle_time*, it will be a breaking point between two sessions. The *max_idle_time* is set by the user and may be updated according to some statistics, such as the ratio of time over page size, to reflect individual's differences in browsing habits. Because of the unpredictable behaviors of users, this could avoid extremely long sessions and non-stop sessions. For example, a user may leave for lunch and totally forget. When the person comes back hours or days later, a new session will be generated.
- The time spent on a page is estimated by subtracting the times of two consecutive requests from the same user. The time spent on the last page is estimated using the average time of all other pages. The actual time spent on each page is hard to calculate precisely because of network traffic, server load, user reading speed, etc. A user profiler is proposed in [16] which uses a client side JAVA applet and can capture more accurately the time spent. However, this requires the deployment of a client side program and modification of Web pages on the server. Besides, it is still impossible to measure the user's

actual viewing time because the person may be distracted after the page is loaded.

- Two thresholds, *min_time* and *min_page*, are used to exclude noises in the data. If the time spent on a page is less than *min_time*, the page is assumed to be uninteresting to the user or an index page and is discarded. If a particular session contains less pages than *min_page*, the session is assumed to be random noise and is removed.
- Some pages contain images as background which should be filtered out. This can be done, for example, by removing image pages by looking at their file name extensions. [17]
- A lot of Web browsers have caching functions so that the server log may not reflect the actual history of browsing. This is still a challenge, but the new protocols such as the HTTP/1.1 can help to solve the problem.

The Web server's log is scanned to identify sessions. A session is created when a new IP address is met in the log. Subsequent requests from the IP address is added to its session as long as the elapse of time between two consecutive requests does not exceed *max_idle_time*. Otherwise, the current session is closed and a new session is created. The steps in identifying sessions are outlined as follows.

1. Read the next record in the Web server's log file.
2. Parse the record and filter out background images.
3. Decide the session according to the IP address.
4. If the elapsed time from last request is within *max_idle_time*, the page is appended into the session. Otherwise, the session is closed and a new session is created for the IP address. The closed session is filtered using *min_time* and *min_page*, and outputed.

In addition, two tables, which map session-ids to IP addresses and page-ids to URLs respectively, are also generated. Moreover, statistics, such as the mean and variance of the time over page size ratio, can be collected as basis for estimating *max_idle_time*.

4 GENERALIZATION-BASED CLUSTERING OF SESSIONS

As mentioned earlier, most clustering methods work for structured data, that is, each data point is represented by a vector in an n -dimensional space. However, lots of data are unstructured or semi-structured,

e.g., transactional data in retails, pages a Web user browses, textual data, etc. The brute force way of dealing with such data, is to represent each object as a vector on all dimensions. For examples, representing each user as a vector of times the user spent on all pages. Such a representation not only makes the dimensionality huge, but also introduces lots of zeros in data. A sparse representation may be used to overcome the problem, but it is inconvenient to handle for many algorithms including BIRCH.

A generalization-based clustering method is introduced in this paper which combines attribute-oriented induction [7] with a clustering algorithm to deal with such kind of semi-structured data. The method first generalizes objects using attribute-oriented induction based on a concept hierarchy which is a partial order among values. The generalized objects are then clustered using the clustering algorithm.

The generalization-based clustering is illustrated here in the Web user clustering. The sessions extracted in Section 3 is generalized based on a concept hierarchy on Web pages – called page hierarchy. The generalized sessions are then passed on to BIRCH for clustering.

4.1 Generalization of Sessions

By analyzing the Web pages, we realize that pages are not randomly created, rather they are organized into a hierarchical structure, called *page hierarchy*. A page hierarchy is a partial order of Web pages. A leaf node in a page hierarchy represents a file in the server, while a non-leaf node represents a directory. A node is a child of another node if the parent's page contains the child's page. For example, a simple page hierarchy for some pages in the UMR's Web server is shown in Figure 1.

Figure 1: An example of page hierarchy.

The page hierarchy can be created automatically or semi-automatically with the help of Webmasters. For example, the hierarchy in Figure 1 can be extracted from the URLs of the pages as shown in the

example (the mnemonic names of generalized pages are optional).

The sessions found in Section 3 are generalized using attribute-oriented induction in which the pages in each session are replaced by their corresponding high level pages based on the page hierarchy. Duplicate pages are then removed with their times added together.

The generalization of the sessions involves two steps: page hierarchy construction and attribute-oriented induction of sessions, which are explained as follows.

1. Construction of page hierarchy.

A page hierarchy is initialized with only the root which represents the home page of the Web server. For each URL in the URL table generated in Section 3, if it is not in the page hierarchy, a node for the page is created. Next, the URL is parsed, and for each prefix which is a legal URL but not in the hierarchy, a node for the page is created. For every pair of nodes in which one's URL is a closest prefix of the other, a link is added. For example, for the URL `http://www.umn.edu/~regwww/ugcr97/ee.html`, nodes for itself, its first prefix `http://www.umn.edu/~regwww/ugcr97/`, and its second prefix `http://www.umn.edu/~regwww/` may be created and a link is added between itself and its first prefix, between its first prefix and its second prefix, and between its second prefix and the root.

2. Attribute-oriented induction of sessions.

For each session found Section 3, its pages are replaced by their corresponding high level pages in the page hierarchy, using the tree climbing strategy in attribute-oriented induction. The level to climb to is decided by the user or inferred from a user-given threshold which specifies the maximum number of high level pages, called *generalized pages*, in results. If two pages in a session are generalized to the same high level page, one of them is removed and its time is added to the other's. The session is said to be generalized and a session so obtained is called a generalized session. For example, according to the page hierarchy in Figure 1, a session (sid_1 , (*Undergraduate Electrical Engineering Courses*, 25), (*Undergraduate Engineering Mechanics Courses*, 48), (*Graduate Electrical Engineering Courses*, 32), (*Graduate Engineering Mechanics Courses*, 19)), will be generalized into a generalized session (sid_1 , (*Undergraduate Courses*, 73), (*Graduate Courses*, 51)).

Since the number of generalized pages is much less than that of the original pages, the generalization of sessions greatly reduce the dimensionality. As a result, a generalized session can then be represented by a regular vector, $(sid, t_1, t_2, \dots, t_n)$, where $t_i, i = 1, \dots, n$, is the total time the user spent on the i -th generalized page and its descendents. Note the page IDs of these generalized pages are not included in the vector because all sessions are on the same set of generalized pages.

4.2 Clustering of Generalized Sessions

The BIRCH algorithm proposed by Zhang et al.[20] is chosen to cluster the generalized objects in our generalization-based clustering method because of the reasons stated above. The BIRCH algorithm is summarized below, followed by its application in the clustering of generalized sessions.

The core of the BIRCH algorithm is the Clustering Feature (CF) vector. Given a set of objects, S , $CF(S) = \langle N, \bar{L}S, SS \rangle$, where N , $\bar{L}S$, and SS are the number, the linear sum, and the square sum of objects in S , respectively. CF vector can be viewed as the summary of the objects, but containing enough information for clustering. A CF tree is a tree where a non-leaf node stores entries of $(CF_i, pointer_to_child_i)$, and a leaf node stores entries of (CF_j) .

A CF tree is dynamically constructed by BIRTH which inserts objects incrementally into the CF tree during clustering. When a new object is inserted into the CF tree, it goes down from the root to a leaf node by choosing the closest child according to a distance measure. If any entry of the leaf node can incorporate the new object within a diameter threshold T , that entry's CF vector is updated. Otherwise, a new entry is created to host the new object. In the later case, if there is no space left on the leaf node, that leaf node is split by selecting two furthest entries as seeds in the two resulting nodes and the rest entries are rearranged according to their distances to the seeds. The ancestors of the leaf node are adjusted by updating the corresponding CF vectors. In case there is a split, a new entry is added to record the new leaf node in the parent node. The parent node is split in a similar way if there is no space and if this go up to the root, the tree is one level deeper. When the tree grows too large to be hold in memory, the threshold T is enlarged and the current tree is converted into a new tree by inserting all leaf nodes of the current tree into the new tree, which is guaranteed to be smaller.

The generalized sessions are passed on to the BIRCH algorithm which creates a CF tree as follows. A minor change is made in BIRCH to increase node size so that it can accommodate high dimensional

data.

1. Initialize an empty CF tree.
2. While there are more generalized sessions
3. read the next generalized session
4. insert it into the CF tree
5. Output the CF tree

The resulting tree is a hierarchical clustering of the generalized sessions. The pages in clusters can be interpreted by referring to the page hierarchy.

5 EXPERIMENTS

The generalization-based clustering of Web users based on browsing patterns has been implemented and tested on a data set collected from the UMR's Web server log. The data set contains roughly one day's access records on the UMR Web server (<http://www.umn.edu>). It contains more than 2.5 million records with a total size of 270MB. The experiments are carried out on a Sun SparcStation Ultra 1 with 64MB of memory running Solaris 2.5. We tested the algorithms on subsets of the dataset which are summarized in Table 1.

Table 1: Data Sets Used in Experiments

data set	records	size	pages	hosts
50k	50,000	5MB	5,731	3,694
100k	100,000	10MB	8,168	6,304
200k	200,000	20MB	12,191	11,473
300k	300,000	30MB	15,201	16,498
400k	400,000	41MB	17,574	21,153
500k	500,000	55MB	21,308	26,107

The records in the data sets are first processed to extract sessions. The *max_idle_time* is set to 30 minutes. The *min_time* and *min_page* are set to 5 and 2 respectively. These seem to be reasonable settings. Other settings are under testing and different settings will be compared to understand the effect of these parameters on results. The number of sessions identified in the data sets is shown in Figure 2. From the figure, it is clear that the number of sessions is linear to the number of records in the data sets. The time spent on session extraction is reported together with the time spent on session generalization.

All the sessions are then generalized to level 2 which is the level just below the root. This is chosen because we want to see the effect of attribute-oriented induction on dimensionality reduction. Experiments are

Figure 2: Number of sessions identified.

Figure 3: Time in session extraction and session generalization.

under way for other levels of generalization, which will be reported in the future. The page hierarchy is the same for all data sets. The generalized sessions have a dimensionality of 1,628 except for the 50k data set which is 1,194. Basically, all pages at level 2 or their descendents are accessed except in the case of 50k. The total time in session extraction and session generalization is almost linear to the data set size as shown in Figure 3.

The generalized sessions are clustered using the BIRCH algorithm. Figure 4 shows the execution times of the algorithm on the data sets. The time is linear to the number of sessions except for the 50k data set which is much less comparing with others. This is because the 50k data set has a smaller dimensionality which makes the distance computation in BIRCH less expensive.

It can be summarized that the approach we pro-

ACKNOWLEDGMENTS

This work is supported by University of Missouri Research Board Grant R-3-42434. We thank Dr. Tian Zhang for the source code of BIRCH and Meg Brady for the testing data set.

References

- [1] J. C. Bezdek and S. K. Pal. *Fuzzy Models for Pattern Recognition*. IEEE Press, 1992.
- [2] M. S. Chen, J. S. Park, and P.S. Yu. Efficient data mining for path traversal patterns in distributed systems. *Proc. 1996 Int'l Conf. on Distributed Computing Systems*, 385, May 1996.
- [3] M. Ester, H.-P. Kriegel, and X. Xu. Knowledge discovery in large spatial databases: Focusing techniques for efficient class identification. In *Proc. 4th Int. Symp. on Large Spatial Databases (SSD'95)*, pages 67–82, Portland, Maine, August 1995.
- [4] O. Etzioni. The world-wide web: Quangmire or gold mine? *Communications of ACM*, 39:65–68, 1996.
- [5] D. Fisher and P. Langley. Approaches to conceptual clustering. In *Proc. 9th Int. Joint Conf. AI*, pages 691–697, Los Angeles, CA, August 1985.
- [6] National Center for Supercomputing Applications. *NCSA httpd*. <http://hoohoo.ncsa.uiuc.edu/docs/Overview.html>, 1995.
- [7] J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases: An attribute-oriented approach. In *Proc. 18th Int. Conf. Very Large Data Bases*, pages 547–559, Vancouver, Canada, August 1992.
- [8] J. Han and Y. Fu. Dynamic generation and refinement of concept hierarchies for knowledge discovery in databases. In *Proc. AAAI'94 Workshop on Knowledge Discovery in Databases (KDD'94)*, pages 157–168, Seattle, WA, July 1994.
- [9] Microsoft Inc. *Internet Information Server*. <http://www.microsoft.com/ntserver/web/exec/overview/overview.asp>, 1999.
- [10] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Printice Hall, 1988.
- [11] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

Figure 4: Time in clustering.

posed in this paper scales up well for large data set. Besides, the resulting clusters are analyzed and several meaningful clusters are found. For example, there is a group of users who are interested in mechanical engineering professors' home pages and another in admission office's Web pages.

6 CONCLUSION AND FUTURE WORK

The clustering of Web users based on their browsing patterns is proposed. A generalization-based clustering method is employed to cluster Web users efficiently. The method introduces attributed-oriented induction in clustering in order to deal with the large dimensionality of data. Our experiments on a large real data set show that the approach is efficient and practical.

Currently, we are conducting more extensive tests with different parameter settings including *max_idle_time*, *min_time*, *min_page*, and generalization level, to study their effects on session extraction and session generalization. The discoveries from the experiments will be reported in the future.

It is found that even after attribute-oriented induction, the dimensionality of generalized sessions is still very large. A possible way to deal with this is first clustering the Web pages [19] and then organizing them into the page hierarchy. Another way is to apply further dimensionality reduction method such Principle Component Analysis on generalized sessions.

Lots of Web sites require their users to register. A natural extension of our method is to combine user's registration information, such as age, income level, address, etc, with their browsing patterns in clustering.

- [12] R. S. Michalski and R. Stepp. Automated construction of classifications: Conceptual clustering versus numerical taxonomy. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 5:396–410, 1983.
- [13] B. Mobasher, N. Jain, S. Han, and J. Srivastava. *Web Mining: Pattern Discovery from World Wide Web Transactions*. Technical Report, University of Minnesota, available at <ftp://ftp.cs.umn.edu/users/kumar/webmining.ps>, 1996.
- [14] J. Moore, S. 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*. Workshop on Information Technologies and Systems, available at <ftp://ftp.cs.umn.edu/users/kumar/web-wits.ps>, 1997.
- [15] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *Proc. 1994 Int. Conf. Very Large Data Bases*, pages 144–155, Santiago, Chile, September 1994.
- [16] C. Shahabi, A. Z. Zarkesh, J. Adibi, and V. Shah. Knowledge discovery from users web-page navigation. In *Proc. of 1997 Int. Workshop on Research Issues on Data Engineering (RIDE'97)*, Birmingham, England, April 1997.
- [17] A. Woodruff, P. M. Aoki, E. Brewer, P. Gauthier, and L. A. Rowe. *An Investigation of Documents from the World Wide Web*. 5th Int. World Wide Web Conference, Paris, France, May, 1996.
- [18] T. W. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal. *From User Access Patterns to Dynamic Hypertext Linking*. 5th Int. World Wide Web Conference, Paris, France, May, 1996.
- [19] O. Zamir, O. Etzioni, O. Madani, and R. Karp. Fast and intuitive clustering of web documents. In *Proc. 1997 Int'l Conf. on Data Mining and Knowledge Discovery (KDD'97)*, pages 287–290, Newport Beach, CA, August 1997.
- [20] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data*, pages 103–114, Montreal, Canada, June 1996.