A Generalization-Based Approach to Clustering of Web Usage Sessions

Yongjian Fu, Kanwalpreet Sandhu, and Ming-Yi Shih

Computer Science Department
University of Missouri-Rolla
{yongjian,ksandhu,mingyi}@umr.edu

Abstract. The clustering of Web usage sessions based on the access patterns is studied. Access patterns of Web users are extracted from Web server log files, and then organized into sessions which represent episodes of interaction between the Web users and the Web server. Using attribute-oriented induction, the sessions are then generalized according to a page hierarchy which organizes pages based on their contents. These generalized sessions are finally clustered using a hierarchical clustering method. Our experiments on a large real data set show that the approach is efficient and practical for Web mining applications.

1 Introduction

With the rapid development of the World Wide Web, or the Web, many organizations now put their information on the Web and provide Web-based services such as on-line shopping, user feedback, technical support, etc. Web mining, the discovery of knowledge on the Web, has become an interesting research area [8]. Research in Web mining has been broadly classified into Web content mining and Web usage mining [7], where the former finds patterns in Web pages and links, and the later finds patterns in the usage of the Web.

An important topic in Web usage mining is the clustering of Web usage sessions. A *Web usage session*, or simply a *session*, is an episode of interaction between a Web user and the Web server. A session consists of the pages the user visited in the episode and the time spent on each page. Because of the number and diversity of Web users, it is obviously unrealistic to study individual sessions. Therefore, an important step in Web usage mining is to cluster the sessions based on their common properties. By analyzing the characteristics of these clusters, webmasters may understand Web usage better and may provide more suitable, customized services to users. The clustering of sessions can provide a very useful tool for many applications in education and e-commerce.

In this paper, the clustering of sessions based on browsing activities or access patterns on the Web is studied. Sessions that exhibit similar browsing activities are grouped together. For example, if a number of customers spend quite a lot time on browsing pages about "baby furniture", "baby toys", and "maternity wear", their sessions may be clustered into a group which could later be analyzed

by webmasters or domain experts as "expecting parents". The webmaster then may arrange the Web pages so that the above pages are linked together and proper advertizements may be inserted. Additionally, when a user has browsed the "baby furniture" and the "baby toys" pages, a link to "maternity wear" pages can be dynamically created and inserted in the current page.

In our approach, the server log data is first processed to identify the sessions. Sessions are then generalized using attribute-oriented induction [9] which will greatly reduce the dimensionality of data. The generalized data is finally clustered using an efficient hierarchical clustering algorithm, BIRCH [24]. The approach is tested on a large, real world data set. Our experiments show that the approach is efficient and we have found several interesting clusters within the data set.

The paper is organized as follows. In Section 2, the background and related work in Web usage clustering are introduced. In Section 3, the process to identify sessions from a Web server log is discussed. We propose a generalization-based clustering method in Section 4, along with its application in the clustering of sessions. In Section 5, results from the experiments of the proposed approach on a large data set are presented. Issues related to session identification and generalization are discussed in Section 6. The study is concluded in Section 7.

2 Background

Earlier studies on Web usage, such as access statistics, lack the in-depth understanding of user browsing patterns, such as pages traversed and times spent on those pages. These user browsing patterns provide accurate, active, and objective information about Web usage. Moreover, most Web servers, e.g., NCSA's HTTPD and Microsoft's IIS, contain such information in their logs of page requests.

A lot of studies have been conducted in Web usage mining. Some focus on the mining of association rules and navigation patterns in the user access paths [3, 7, 5]. Others build data cubes from Web server logs for OLAP and data mining [22, 4]. There is also research on data preparation [6] and query language [19] for Web usage mining.

Let us first explain the Web server log since it is the source for almost all Web usage mining. A Web server log contains records of user accesses. Each record represents a page request from a Web user (called client). A typical record contains the client's IP address, the date and time the request is received, the URL of the page requested, the protocol of the request, the return code of the server indicating the status of the request handling, and the size of the page if the request is successful. Several examples 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`.

```
john.cs.umr.edu - - [01/Apr/1997:01:20:01 -0600]
```

```
  "GET /~regwww/ssfs97/fs97.html HTTP/1.0" 200 1094
john.cs.umr.edu - - [01/Apr/1997:01:20:01 -0600]
  "GET /~regwww/ssfs97/rdball.gif HTTP/1.0" 200 967
john.cs.umr.edu - - [01/Apr/1997:01:20:17 -0600]
  "GET /~regwww/ssfs97/depf97.html HTTP/1.0" 200 5072
john.cs.umr.edu - - [01/Apr/1997:01:21:08 -0600]
  "GET /~regwww/ssfs97/ecf97.html HTTP/1.0" 200 12146
```

In Web usage clustering, sessions are extracted from the Web server log and clustered by a clustering algorithm.

The data preparation process to identify the sessions in our approach is similar to that in many other studies, such as [15, 21, 18, 19]. It produces reasonable results with a simple and fast algorithm. More sophisticated approaches are possible, such as those discussed in [6].

Clustering has been studied extensively in statistics, machine learning, pattern recognition, and database systems, with a large number of algorithms developed [12–14, 2, 24]. Recently, an algorithm, Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [24], was proposed in the database area. It represents clusters in a Clustering Feature (CF) tree and uses a similar insertion algorithm to the B+ tree insertion algorithm. BIRCH has several features which are desirable for our purpose.

- It usually creates a good clustering in just one scan of the data set. The complexity of the algorithm is $O(n \log(n))$, so it scales up well for very large data sets.
- It builds clusters hierarchically such that a high-level cluster is further divided into subclusters and so on. A high-level cluster may represent the sessions which reflect general common interests and their subgroups represent more specific common interests.
- It is an incremental algorithm such that when new data arrives the clustering can be updated incrementally, not starting from scratch. This is important because new data continues to be added as long as users continue browsing the Web pages.

However, the direct application of BIRCH on the primitive user access data as described above is very inefficient and may not find many interesting clusters as explained below.

- A Web server usually contains thousands, even millions of pages. It is obviously impractical to represent each session as a high dimensional vector in which each dimension represents a page. A sparse representation will cause the dimensionality of the clusters to change dynamically, i.e, the non-sparse items in the centroid of a cluster may grow when new vectors are added. This imposes a lot of overhead on memory and storage management as well as on data structure handling.
- Our goal is to cluster the sessions with similar access patterns. However, it is not easy to find many sessions with common pages because of the diversity

of the Web users. This will lead to either small clusters or clusters with little similarity. It is more likely to find groups who share the common interests in the themes of pages. For example, we may not find a large number of users who visit pages on "Okla Homer Smith Mavado crib - white finish", "Kids II Bright Starts musical mobile", and "Dividends Maternity Pullover Tunic", but we can probably find lots of users who browse pages on "baby furniture", "baby toys", and "maternity wear".

Based on the above observations, we propose a *generalization-based clustering method* which combines attribute-oriented induction [9] and BIRCH to generate a hierarchical clustering of the sessions. In this method, the sessions are first generalized in attribute-oriented induction according to a data structure, called page hierarchy, which organizes the Web pages into a hierarchical structure based on their topics. The generalized sessions are then clustered by BIRCH.

The clustering of sessions based on access patterns has been studied in [21] and [18]. In [21], 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 used, the *leader algorithm*, is simple and the quality of the final clusters is unpredictable.

An applet-based user profiler is proposed in [18] which can more accurately record the times users spend on pages. Using the k-means method, sessions are clustered based on their navigation paths, i.e., the pages as well as the order 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.

Unlike previous approaches to Web usage clustering in [21] and [18], our approach employs an efficient, incremental, and hierarchical clustering algorithm, BIRCH. More significantly, we introduce attribute-oriented induction to generalize sessions before they are clustered. Our approach has the following advantages over the previous approaches.

- The generalization of sessions allows us to find clusters which cannot be found in the original sessions. These additional clusters represent similarities at higher levels of abstraction. It should be pointed out that the clustering of generalized sessions will still find clusters in the original sessions, since the generalization is performed according to the page hierarchy which tries to preserve the original structure of the pages.
- The dimensionality of sessions is greatly reduced because of the generalization. This not only reduces the computational complexity, but also relaxes the memory requirement. In fact, the dimensionality of our data set is so large that BIRCH cannot run or does not give a meaningful clustering (it generates a single cluster) in many cases without generalization.
- As we mentioned earlier, BIRCH is more appropriate for the clustering of sessions than the leader algorithm in [21] and the k-means algorithm in [18].

It should be noted, though, that the generalization is independent of the clustering algorithm. Our approach can be easily adapted to use other clustering algorithms.

Our contributions can be summarized as follows.

- A generalization-based clustering method for clustering sessions is introduced.
- A simple and intuitive method to construct the page hierarchy is developed.
- Experiments have been conducted to understand the effectiveness of the generalization-based clustering method.

The use of attribute-oriented induction for conceptual clustering which has been mentioned in [11], deals with nominal data. To the best of our knowledge, attribute-oriented induction has not been used for clustering numerical data or sessions. In our approach, it is employed to deal with sessions which contain numerical data, the elapsed times on the pages.

3 Session Identification

A Web user may visit a Web site from time to time and spend an arbitrary amount of time between consecutive visits. To deal with the unpredictable nature of Web browsing and make the problem tractable, a concept, *session*, is introduced to represent an episode of interaction between a user and a Web server. We cluster the sessions instead of the users' entire history. This can be justified because our goal is to understand the usage of the Web and different sessions of a user may correspond to the visits of the user with different purposes on mind. Concepts similar to session have been proposed in other studies [6, 19, 16, 15, 18].

A session consists of the pages accessed and the times spent on these pages by a user in the episode. The boundary of a session is determined by a threshold, *max_idle_time*. If a user stays inactive (not requesting any page) for a period longer than *max_idle_time*, subsequent page requests are considered to be in another episode, thus another session. The threshold *max_idle_time* may be updated according to some statistics, such as the ratio of time over page size, to reflect individual's differences in browsing habits. The use of *max_idle_time* could avoid extremely long sessions and non-stop sessions. For example, a user may leave for lunch in the middle of browsing and totally forget about it. When the person comes back hours or days later, a new session will be generated.

Sessions are identified by grouping consecutive pages requested by the same user together. The data in a Web server log is transformed into a set of sessions in the form of (*session-id,* $\{\langle$*page-id, time*$\rangle\}$), where *session-id* and *page-id* are unique IDs assigned to sessions and pages. For example, a session (sid_0, p_0, 10, p_1, 30, p_2, 20) indicates a user spent 10 seconds on page p_0, 30 seconds on page p_1, and 20 seconds on page p_2.

The Web server log is scanned to identify sessions. A session is created when a new IP address is met in the log. Subsequent requests from the same IP

address are added to the same 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.

Two other 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 [6] and is discarded. If a particular session contains less than *min_page* pages, the session is assumed to be noise and is removed.

The time spent on a page is estimated to be the difference between the page's request date/time and the date/time of the next request from the same client. The time spent on the last page of each session is approximated by the average time of all other pages in the session.

The algorithm for identifying sessions is outlined as follows.

1. While there are more records in the Web server log file, do the following steps.
2. Read the next record in the Web server log file.
3. Parse the record to get IP address, URL of the page, and time and date of the request.
4. If the page is a background image file, it is discarded. The image files can be detected by looking at their file name extensions, e.g, .gif, .jpeg, etc. [20]
5. Decide the session according to the IP address.
6. If the elapsed time from last request is within *max_idle_time*, the page is appended into the session.
7. 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 output.

In addition, two tables, an IP table which maps session-ids to IP addresses and a URL table which maps page-ids to URLs, are generated. Moreover, statistics, such as the mean and variance of the time over page size ratio, can be collected as basis for tuning *max_idle_time*.

4 Generalization-based Clustering of Sessions

The sessions obtained as described in Section 3 will be generalized, which is explained in Section 4.1, and then clustered, which is discussed in Section 4.2.

4.1 Generalization of Sessions

As described in Section 3, a session consists of a set of (page-id, time) pairs. Such a sparse representation would give a lot of problems for clustering algorithms, especially for BIRCH whose core data structures are of a fixed size. A straightforward solution is to represent the sessions on all pages, padding zeros for pages not in a session. However, this works poorly when the total number of pages is large.

Moreover, the objective of clustering is to finds groups with similar access patterns, which do not necessary correspond to page level. Groups that are not obvious at page-level may emerge when considered at a higher level, as explained in Section 2.

By analyzing the pages, we realize that they are not randomly scattered, but are organized into a hierarchical structure, called a page hierarchy. A *page hierarchy* is a partial ordering of the Web pages, in which a leaf node represents a Web page corresponding to a file in the server. A non-leaf node in a page hierarchy represents a Web page corresponding to a directory in the server. A link from a parent node to a child node represents the consisting-of relationship between the corresponding pages. A page hierarchy for some pages in the UMR Web server is shown in Fig. 1. For example, a graduate electrical engineering course page is in the graduate course page which in turn is in the registrar office's page.

To distinguish the two kinds of pages, a page represented by a leaf node is called a *simple page*; and a page represented by a non-leaf node is called a *general page*. For example, the page hierarchy in Fig. 1 has four simple pages and three general pages.

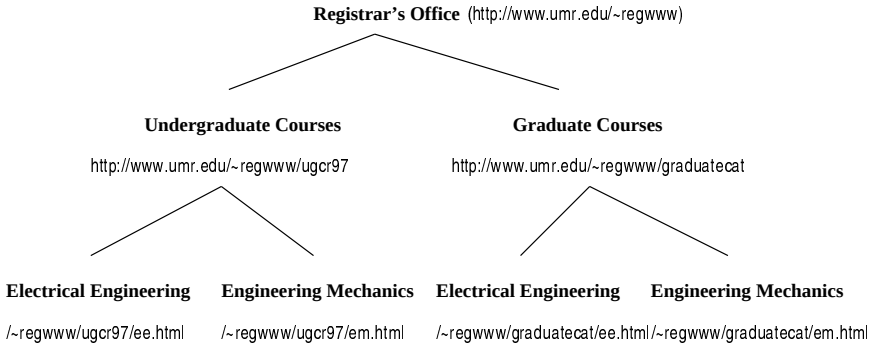

Fig. 1. An example of page hierarchy.

The page hierarchy can be created automatically based on the URLs of the pages. For example, the page hierarchy in Fig. 1 can be constructed from the four simple pages below (the mnemonic names of the general pages are optional). The root of the hierarchy which is the home page of UMR is not shown in Fig. 1 for simplicity.

```
http://www.umr.edu/~regwww/ugcr97/ee.html
http://www.umr.edu/~regwww/ugcr97/em.html
http://www.umr.edu/~regwww/graduatecat/ee.html
http://www.umr.edu/~regwww/graduatecat/em.html
```

The steps to construct the page hierarchy are outlined as follows.

1. The page hierarchy is initialized with only the root which represents the home page of the Web server.
2. For each URL in the URL table generated as described in Section 3, if it does not already exist in the page hierarchy:
   (a) A node for the page is created. Next, the URL is parsed and for each prefix which is a legal URL, a node for the general page is created if it does not exist in the page hierarchy.
   (b) For every pair of nodes in which one's URL is the longest prefix of the other's, a link is added if it is not already in the page hierarchy.

For example, for a URL `http://www.umr.edu/~regwww/ugcr97/ee.html`, nodes for itself, its first prefix `http://www.umr.edu/~regwww/ugcr97/`, and its second prefix `http://www.umr.edu/~regwww/` will be created, if they are not already in the page hierarchy. 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.

The sessions are generalized using attribute-oriented induction [9]. In attribute-oriented induction, the simple page in each session is replaced by its corresponding general page in the page hierarchy. Duplicate pages are then removed with their times added together. The session is said to be generalized and a session so obtained is called a generalized session. For example, for a session of simple pages in Fig. 1, `((Undergraduate Electrical Engineering Courses, 25), (Undergraduate Engineering Mechanics Courses, 48), (Graduate Electrical Engineering Courses, 32), (Graduate Engineering Mechanics Courses, 19))`, attribute-oriented induction will generalize it to a generalized session `((Undergraduate Courses, 73), (Graduate Courses, 51))` at level 2.

Since the number of general pages is much smaller than that of simple pages, the generalization of the sessions greatly reduces the dimensionality. As a result, a generalized session can then be represented by a vector, (*session-id*, $t_1, t_2, \cdots, t_n$), where t_i is the total time the user spent on the i-th general page and its descendents. Note the page-ids of these general pages are not included in the vector because all sessions are based on the same set of general pages.

Another advantage of the generalization of sessions using the page hierarchy is that the resulting data representation can accommodate updates on the Web site, such as addition and deletion of pages, as long as the higher level structure is stable.

4.2 Clustering of Generalized Sessions

The generalized sessions are clustered using BIRCH [24] which can be summarized as follows.

BIRCH builds a Clustering Feature (CF) tree as the result of clustering by incrementally inserting objects (represented by vectors) into the CF tree. A CF tree is a multidimensional structure, like a B+ tree, in which a nonleaf node stores B (called branching factor) entries of (CF_i, *pointer_to_child_i*), and a leaf

node stores B entries of (CF_i), where CF_i is a CF vector. A CF vector is a triple containing the number, the linear sum, and the square sum, of the vectors in the subtree rooted at a child.

When a new vector 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, such as Euclidean distance. 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, the object is put into an empty entry in the leaf node. In the later case, if there is no empty entry left in the leaf node, it is split into two. In case there is a split, an empty entry in the parent node is used to record the new leaf node. The parent node is split in a similar way if there is no empty entry and if this goes up to the root, the tree is one level deeper. When the tree grows too large to be held in memory, the threshold T is enlarged and the current tree is converted into a new tree by inserting all leaf node entries of the current tree into the new tree, which is guaranteed to be smaller.

The generalized sessions are read by BIRCH and inserted into an initially empty CF tree one by one. The resulting tree is a hierarchical clustering of the generalized sessions. The general pages in the clusters can be interpreted by referring to the page hierarchy. A minor change is made in BIRCH to increase its node size so that it can accommodate high dimensional data.

5 Experiments

The algorithms have been implemented and tested on a data set collected from UMR's Web server log (`http://www.umr.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.

Several test sets are used in our experiments, which are subsets of the data set, as summarized in Table 1. The thresholds *max_idle_time*, *min_time*, and *min_page* are set to 30 minutes, 1 second, and 2 pages, respectively. The *max_idle_time* is set according to common practices[6]. The *min_time* and *min_page* are set to include as many pages and sessions as possible. Our experiments show similar results for other settings. The branching factor B in BIRCH is set to 3. The threshold T in BIRCH is initialized to 0 such that each entry holds one vector (generalized session).

5.1 Effectiveness of Generalization

The effects of generalization on dimensionality and cluster quality are examined. We have tested our approach on subsets of the Web site. A subset of the Web site is specified by restricting pages in a session to be in a sub-hierarchy of the page hierarchy. Pages that are not in the sub-hierarchy are ignored. This reduces the dimensionality and cardinality of the generalized sessions, thus enabling us to examine the clusters in detail and to cluster sessions without generalization.

Table 1. Test sets

test set	no_of_records	no_of_distinct_pages	no_of_distinct_hosts
50k	50,000	5,731	3,694
100k	100,000	8,168	6,304
200k	200,000	12,191	11,473
300k	300,000	15,201	16,498
400k	400,000	17,574	21,153
500k	500,000	21,308	26,107

Three sub-hierarchies are selected in various domains. The root of the RO sub-hierarchy is the home page of the Registrar's Office (`~regwww`). The root of the MAEM sub-hierarchy is the home page of the Department of Mechanical Engineering, Aerospace Engineering, and Engineering Mechanics (`~maem`). The root of the HELP sub-hierarchy is the home page of the Help Desk of Computing Services (`helpdesk`). The sub-hierarchies are summarized in Table 2. The 500k test set is used and the number of sessions for each sub-hierarchy is reported in Table 2. The sessions are generalized to level 2 which is the level just below the root.

Table 2. Sub-hierarchies of the page hierarchy.

sub-hierarchy	total_nodes	leaf nodes	no_levels	no_sessions
RO	137	134	3	663
MAEM	102	92	4	132
HELP	101	43	5	243

The dimensionality of the sessions before and after the generalization is shown in Fig. 2. It is obvious from the figure that the generalization significantly reduces the dimensionality. In the MAEM case, it is reduced by an order of magnitude.

To examine the effects of generalization on clustering quality, the leaf clusters (leaf nodes in the resulting CF tree) are examined, as they are good representatives of the final clustering.

The number of leaf clusters with generalization are compared with that without generalization in Fig. 3. In the HELP and the RO cases, we get fewer leaf clusters as expected, since clusters are merged after generalization. In the MAEM case, we get more leaf clusters, probably because the data is more diverse and BIRCH mislabels some clusters. In all three cases, the differences are within 10%. This is because the number of leaf clusters is dominated by the number of sessions, which is the same with or without generalization.

It is more interesting to look at the number of nontrivial leaf clusters. A nontrivial leaf cluster is a leaf cluster that contains more than one session and has at least one common page among the sessions in the cluster. Fig. 4 shows

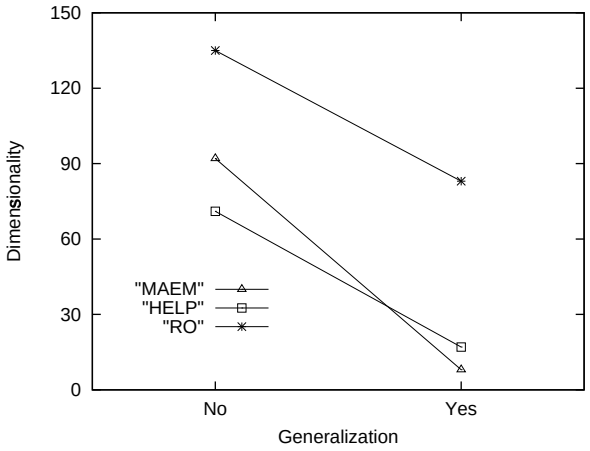

Fig. 2. Dimensionality of the sessions.

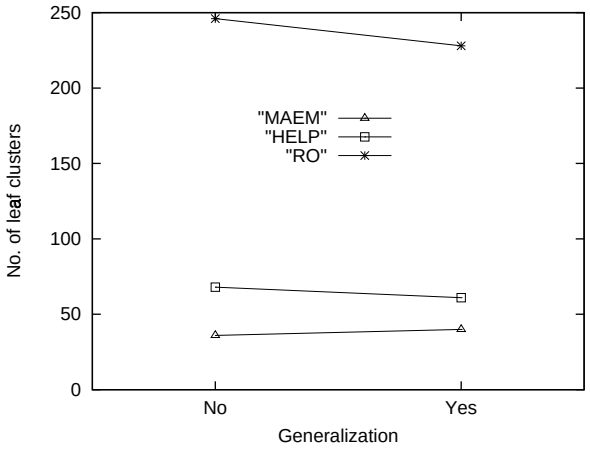

Fig. 3. Number of leaf clusters.

the percentage of total leaf clusters that are nontrivial. In all three cases, the generalization increases the percentages by more than 10. In the HELP case, the generalization increases the percentage of nontrivial leaf clusters by almost 20. This clearly demonstrates that generalization improves the quality of clustering.

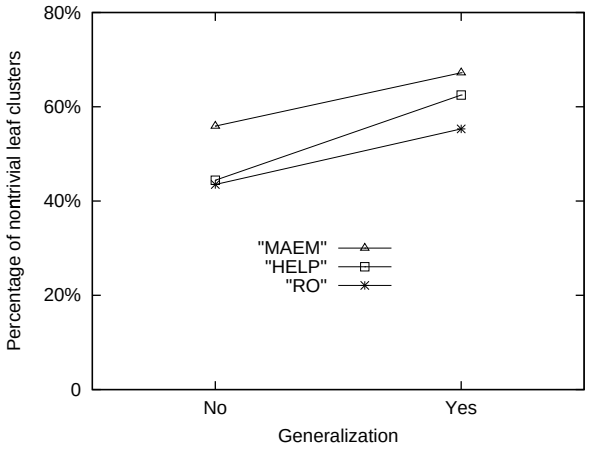

Fig. 4. Number of nontrivial leaf clusters.

To study the effects of different generalization levels, the HELP sub-hierarchy is used because it has the largest number of levels. Experiments are performed at all levels except the level 1 (root) which is meaningless. The number of leaf clusters, number of nontrivial leaf clusters, and dimensionality, are shown in Fig. 5. The higher we generalize, the more nontrivial leaf clusters we find. However, it is possible some of them are false clusters created by generalization. In general, the generalization level should be selected to the highest level which still keeps the logic of page hierarchy. It is also a good idea to try several generalization levels.

5.2 Scale Up of the Method

The execution times in session identification/generalization and clustering for the test sets are shown in Fig. 6. All the sessions are generalized to level 2. The page hierarchy is the same for all the test sets. The generalized sessions have a dimensionality of 1,628 except for the 50k test set which is 1,194. Basically, all general pages at level 2, or their descendents, are accessed except in the case of 50k. The time in session identification/generalization is linear to the number of records in the test sets. This is not surprising as the session generalization is linear to the number of sessions which is found to be linear to the number of records in the test sets. The time in clustering is almost linear to the number of records

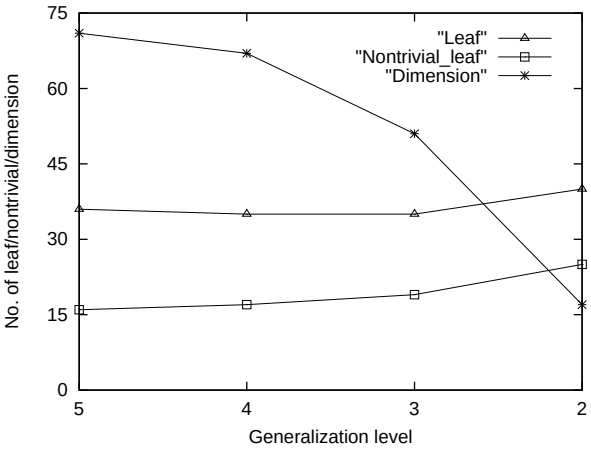

Fig. 5. Effects of generalization levels.

in the test sets except for the 50k test set which is much less when compared with the others. This is because the 50k test set has a smaller dimensionality which makes the distance computation in BIRCH less expensive.

This demonstrates that our approach scales up well for large data sets. The results also confirm the analysis on BIRCH [24] and attribute-oriented induction [11].

5.3 Cluster Analysis

A preliminary analysis of the clustering has been performed to verify that the clusters make sense. The resulting clusters are analyzed by examining the general pages in them. Most clusters in the hierarchical clustering contain only a few general pages which are considered as the general interest of the group. However, because of the complexity of the task and our limited manpower, we have not done sophisticated analysis. The clustering will be passed on to the webmasters of the University for more thorough analysis, who are more familiar with the Web pages.

In each of the three sub-hierarchies we tested, we find some interesting clusters that are not in the original sessions. For example, in the MAEM sub-hierarchy, there are three groups who spent 5-8, 15-25, and 30+ minutes, respectively, on the personal home pages of the faculty and staff in the Department. This is not found in the original sessions because the sessions consist mostly of different home pages. The general page that represents them is actually the most visited general page. Most of the users are internal, that is, from UMR. Based on the finding, the department may, for example, provide help to the faculty and staff for developing their home pages.

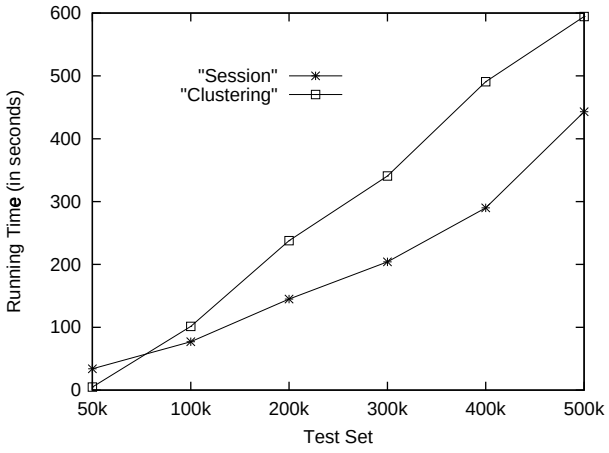

Fig. 6. Execution time.

In the HELP sub-hierarchy, generalization leads to the finding of a large group who look for help about the Web, instead of small groups who are interested in specific topics about the Web. This justifies the arrangement of the helpdesk Web page which has a prominent item on *help on WWW/Internet*, separating from other networking and application topics.

In the RO sub-hierarchy, there is a group who has browsed pages on the course catalog and pages on courses offered in the Fall/Summer 1997 semesters (the log was for April 1997). It would be convenient to link the two together so that a student can check the course availability when making a plan. Another group looks at pages on transcripts such as how to get a transcript in addition to the course offerings. It can be guessed these users are students who are going to apply for jobs. A link to long-distance learning opportunities may be helpful to some of them.

Sometimes, the pages in a cluster are very general and we can look at the simple pages in the data set to find out more. For example, there is a cluster containing a single general page on course offerings. A detailed analysis reveals a group interested in mechanical engineering courses and another in economics courses.

It can be summarized that the method we proposed in this paper is effective in finding interesting clusters and that it scales up well for large data sets. In particular, the generalization not only greatly reduces the dimensionality of the sessions, but also improves the quality of the clustering. A few meaningful clusters which are not in the original sessions have been found in our data set.

6 Discussion

The related issues in session identification and generalization are addressed in this section.

6.1 Session Identification

Although a lot of progress has been made in session identification, it is still a difficult problem. The difficulties in session identification are discussed below.

1. It is sometimes hard to identify unique users based just on server logs. One problem is that many Web servers only record the IP address of clients, which can be shared by more than one user. This does not pose a serious threat if the session boundary can be clearly decided since each session is usually contributed by a single user.
   A more serious problem arises when some Internet routers, proxy servers, and ISPs hide the actual IP address of clients by a random or anonymous IP address. The solutions proposed so far include cookies, user registration, and heuristics based on the clients' browser type and links among pages [6]. However, none of these provides a fail-proof solution.
   A somewhat related issue is to identify concurrent sessions from the same user. This can happen, for example, when a user opens multiple windows of a browser. The activities from all windows will be represented as one session in our approach. This could cause a problem if each window corresponds to a task in the user's mind, and thus needs to be represented as a session. Heuristics such as referal pages [6] may help if the windows follow different paths.
2. In our approach, the time spent on a page is estimated by subtracting the times of two consecutive requests. The time spent on the last page is estimated using the average time of all other pages. However, the actual time spent on each page is almost always different from the estimation because of network traffic, server load, user reading speed, etc. A user profiler is proposed in [18] 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.
   A possible sidewalk solution is to discard the time information all together. The sessions can be simply represented by the pages in them. Each page now becomes a binary variable. Our preliminary analysis is that the clusters found will be similar to those found with time information. More experiments need to be conducted to make a conclusive claim.
3. A lot of Web browsers have caching functions so that the server log may not reflect the actual history of browsing. If a cached page has been accessed in the session, the page will be in the session, but its time may be inaccurate. Some proxy servers also cache pages for different users, thus the cached

pages are not present in many sessions. One way to deal with it is to disable caching by expiring a page right after it is fetched. This, however, defeats the purpose of caching. Some heuristics used to detect unique users can also be used to identify the missing pages [6].

6.2 Generalization

Two important factors in generalization are the generalization level and the page hierarchy, which are discussed below.

1. In the generalization of sessions, a level is given or implied from a threshold to determine the general pages which are going to replace the simple pages. This level plays an important role in forming the generalized sessions as well as the final clustering, as demonstrated in our experiments. If the level is set too high, over-generalization may occur in which too many details are lost and the validity of the clusters may be in question. On the other hand, if the level is set too low, the clustering algorithm may not find the clusters existing only at higher levels. Besides, the dimensionality may be too large. We report an initial experience in our data set, but experiments on other data sets are needed to gain more insights on the issue.
2. The construction of the page hierarchy based on the URLs of pages implies that the underlying page organization reflects the semantics of the pages. In case this cannot be assumed, the page hierarchy should be constructed according to the semantics of the pages, e.g., by using a document clustering [17, 23] or categorization method [16]. A semi-automatic approach with the help of the webmaster is another alternative. Our approach can be easily extended to use page hierarchies so constructed.

In all, the generalization of the sessions reveals clusters which are more general and broad. The goal is to find high-level patterns without losing important details. However, the generalization will lose some information and may group sessions which do not belong to the same cluster. In some extreme cases where clustering on the original sessions is needed, the generalization level should be set to positive infinity so the sessions are not generalized.

7 Conclusion

The clustering of the Web usage sessions based on access patterns has been studied. A generalization-based clustering method is proposed which employs attribute-oriented induction in clustering to deal with the high-dimensional data in Web usage mining. Our experiments on a large real data set show that the method is efficient and practical for Web mining applications.

It is found that even after attribute-oriented induction, the dimensionality of generalized sessions may still be very large. One possible solution is to use a subspace clustering algorithm which partitions the vector space for clustering [1].

Another is to dynamically adjust the page hierarchy [10] to further reduce the dimensionality at higher levels.

The clustering algorithm used in our experiments, BIRCH, seems to degrade when the dimensionality increases. There are lots of singleton leaf clusters in the final clustering for almost all the test sets. Currently several alternative clustering algorithms are under consideration. We are also working on the improvement of the BIRCH algorithm.

A natural extension of our method is to combine user registration information, such as age, income level, address, etc, with their access patterns in clustering. It would be interesting to see how each can contribute to the quality of clustering.

An important topic for future research is the use of clustering in applications such as education and e-commerce. A possible direction could be how to design Web pages that are attractive to targeted groups.

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 and Dan Uetrecht for the data set. Mario Creado's help during the experiments is greatly appreciated.

References

1. R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Seattle, Washington, 1998.
2. J. C. Bezdek and S. K. Pal. *Fuzzy Models for Pattern Recognition.* IEEE Press, 1992.
3. J. Borges and M. Levene. Mining association rules in hypertext databases. In *Proc. 1998 Int'l Conf. on Data Mining and Knowledge Discovery (KDD'98)*, pages 149–153, August 1998.
4. A. Büchner and M. Mulvenna. Discovering internet marketing intelligence through online analytical web usage mining. *SIGMOD Record*, 27, 1998.
5. 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.
6. R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Journal of Knowledge and Information Systems*, 1, 1999.
7. R. Cooley, B. Mobasher, and J. Srivastava. Web mining: Information and pattern discovery on the world wide web. In *Proc. Int. Conf. on Tools with Artificial Intelligence*, pages 558–567, Newport Beach, CA, 1999.
8. O. Etzioni. The world-wide web: Quangmire or gold mine? *Communications of ACM*, 39:65–68, 1996.
9. 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.

10. 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.
11. J. Han and Y. Fu. Exploration of the power of attribute-oriented induction in data mining. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 399–421. AAAI/MIT Press, 1996.
12. A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Printice Hall, 1988.
13. L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis.* John Wiley & Sons, 1990.
14. 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.
15. B. Mobasher, N. Jain, S. Han, and J. Srivastava. *Web Mining: Pattern Discovery from World Wide Web Transcations.* Technical Report, University of Minnesota, avialable at ftp://ftp.cs.umn.edu/users/kumar/webmining.ps., 1996.
16. 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, avialable at ftp://ftp.cs.umn.edu/users/kumar/web-wits.ps., 1997.
17. M. Perkowitz and O. Etzioni. Adaptive web pages: Automatically synthesizing web pages. In *Proc. 15th National Conf. on Artificial Intelligence (AAAI/IAAI'98)*, pages 727–732, Madison, Wisconsin, July, 1998.
18. 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.
19. M. Spiliopoulou and L. Faulstich. Wum: A web utilization miner. In *Proc. EDBT Workshop WebDB'98*, Valencia, Spain, 1998.
20. 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.
21. 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.
22. O. R. Zaïane, X. Xin, and J. Han. Discovering web access patterns and trends by applying olap and data mining technology on web logs. In *Proc. Advances in Digital Libraries*, pages 19–29, 1998.
23. 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.
24. 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.