On Mining Web Access Logs

Anupam Joshi
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County, Baltimore, MD 21250
joshi@cs.umbc.edu
Raghu Krishnapuram
Department of Mathematical and Computer Sciences
Colorado School of Mines, Golden, CO 80401
rkrishna@mines.edu

Abstract

The proliferation of information on the world wide web has made the personalization of this information space a necessity. One possible approach to web personalization is to mine typical user profiles from the vast amount of historical data stored in access logs. In the absence of any *a priori* knowledge, unsupervised classification or clustering methods seem to be ideally suited to analyze the semi-structured log data of user accesses. In this paper, we define the notion of a "user session", as well as a dissimilarity measure between two web sessions that captures the organization of a web site. To extract a user access profile, we cluster the user sessions based on the pair-wise dissimilarities using a robust fuzzy clustering algorithm that we have developed. We report the results of experiments with our algorithm and show that this leads to extraction of interesting user profiles. We also show that it outperforms association rule based approaches for this task.

1 Introduction

The proliferation of information on the world wide web has made the personalization of this information space a necessity. This means that a user's interaction with the web information space should be tailored based on information about him/her. For example, a person in Switzerland searching for ski resorts is more likely to be interested in the Alps, whereas a person in Colorado would likely be interested in the Rockies. Personalization can either be done via information brokers (e.g. web search engines), or in an *end to end* manner by making web sites adaptive. Initial work in this area has basically focused on creating broker entities, often called recommender systems. One of the earliest such systems was the Firefly system [1] which attempted to provide CDs that best match a user's professed interests. More recently, systems such as PHOAKS [4] and our own W^3IQ[2, 3] have sought to use cooperative information retrieval techniques for personalization.

End–End personalization is predicated on adaptive web sites[15, 16], which change the information returned in response to a user request based on the user. Very primitive forms of this can be seen in sites that ask the users to provide some basic information (address, phone, keywords indicating interest), and then tailor their information content (and especially ads) based on things like zip code, area code and demographic profile. However, in general the appearance of a particular page, including links on it, can also be changed when web sites are adaptive. Perhaps the earliest work along similar lines was the Webwatcher project[5] at CMU. It highlights hyperlinks in a page based on the declared interests and the path traversal of a user as well as the path traversals of previous users with similar interests. There is also a recent body of work[18, 17] which seeks to transform the web into a more structured, database like entity. In particular, Han et al.[17] create a MOLAP based warehouse from web logs, and allow users to perform analytic queries. The also seek to discover time dependent patterns in the access logs[21].

Mining typical user profiles from the vast amount of historical data stored in server or access logs is a possible approach to personalization that has been recently proposed[28, 7, 20], and some initial work done. In [7], associations and sequential patterns between web transactions are discovered based on the Apriori algorithm [8]. The logs are first split into sessions (transactions), and then the apriori algorithm used to discover associations between sessions. However, in creating sessions, an assumption is made that the identity of the remote user is logged by the web server. Except for rare instances when the server is so configured and the remote site runs *identd* in a mode that permits plaintext transfer of ids, this assumption is clearly not valid. Chen et. al.[20] also use association rule algorithms (FS and SS) to find associations between user sessions. They define a

session (traversal pattern in their nomenclature) to be a set of *maximal forward references*, in other words, a sequence of web page accesses by a user in which s/he does not revisit an already visited page. The claim is that a backward reference is mostly for ease of navigation. However, that is not necessarily the case – users may seek to revisit a page to read more, or clarify what they had read in light of new information on a subsequent page. Also like [7] they assume that user ids are known.

Note that most existing efforts to mine web logs have relied on association rule type techniques. These can be inadequate for extracting profiles from web log data. First, they are not resilient to the noise typically found in the logs due to a wide variety of reasons inherent to web browsing and logging. There is a significant percentage of time (sometimes as large as 20-30 percent) that a user is simply "browsing" the web site and does not follow his normal access pattern. For example, a user who typically goes to CNN's site for sports news will also visit their (say) politics and national news sections every so often. Moreover, the noise contamination rate and the scale of the data is rarely known in advance.

Further, the data involved in web mining lend themselves better to a "fuzzy" approach which allows for degrees of similarity between entities. In particular, association rule techniques assume that each item is distinct, so any two items are either the same, or not. This creates a problem when we apply association rules to user sessions, which have as their elements the URLs visited in the session. Consider for example three sessions with one URL visited each (http://www.anyu.edu/courses/mycourse/hw.html), (http://www.anyu.edu/courses/mycourse/proj.html), and (http://www.anyu.edu/academics/admission.html). Since each session has a distinct URL, association rule techniques will not group session 1 and 2 into the same "large" itemset, even though it is fairly clear to a human observer from the context (i.e. structure of the web site) that they should be grouped together. This is principally because as defined, association rule algorithms cannot handle graded notions of similarity between itemsets. *Han et al.* [23] have suggested creating an attribute hierarchy, merging together attributes at its various levels. However, the hierarchy needs to be explicitly created and items merged (by the user) before the association rule algorithms can be run. As we shall show later, our approach has this hierarchical notion built in and does not need user intervention.

We propose to use unsupervised clustering methods to analyze the semi-structured log data of user accesses by categorizing them into classes of user sessions. The URLs in each session then represent a typical traversal pattern – i.e. they are often visited together. This information can be used in a variety of ways, including the creation of adaptive web sites. At the very minimum, this information can be used by the site designer to reorganize the site to better convey the information to the user.

Categories in most web mining tasks are rarely well separated. In particular, some sessions likely belong to more than one group to different degrees. The class partition is best described by fuzzy memberships, particularly along the overlapping borders. Also, it is necessary for the clustering process to work with relational[1] data. As opposed to object data when the data elements represent points in some n-dimensional space and the distances between them are Minkowski norms, relational data means that only pairwise distance between the data elements can be described. In particular, it is not obvious how to map two objects web sessions into numerical features in a manner that makes (Minkowski) distances between them meaningful. This immediately rules out the use of fast clustering algorithms developed by the data mining community such as CLARANS[27] and Birch[26], which only deal with object data.

2 Clustering Access Logs

Sessionizing

The access log for a given web server consists of a record of all files accessed by users. Each log entry consists of :(i) User's IP address, (ii) Access time, (iii) Request method ("GET", "POST", $\cdots$), etc), (iv) URL of the page accessed, (v) Prototcol (typically HTTP/1.0), (vi) Return code, (vii) Number of bytes transmitted. We filter out some of these entries. These include entries that: (i) result in any error, (ii) use a request method other than "GET", or (iii) record accesses to image files (.gif, .jpeg, , $\cdots$, etc), which are typically embedded in other pages and are only transmitted to the user's machine as a by product of the access to a certain web page which has already been logged. While error entries contain potentially useful information, they do not serve any purpose with regards to finding traversal patterns.

Analogous to [7], the individual log entries are grouped into user sessions using a perl script which is a modification of [22]. A user session is defined as a sequence of temporally compact accesses by a user. Since web servers do not typically log usernames (unless *identd* is used), we define a user session as accesses from the same IP address such that the duration of time elapsed between any two consecutive accesses in the session is within a pre-specified threshold. Each URL in the site is assigned a unique number $j \in \{1, \ldots, N_U\}$, where N_U is the total number of valid URLs.

[1] Note that this term is used in its statistical sense, not as used by the database community

Thus, the i^{th} user session is encoded as an N_U-dimensional binary attribute vector $\mathbf{s}^{(i)}$ where $s_j^{(i)}$ is 1 if the user accessed the j^{th} URL during the i^{th} session, and 0 otherwise.

The ensemble of all N_S sessions extracted from the server log file is denoted by $\mathcal{S}$. Note that our scheme will map one user's multiple sessions to multiple user sessions. However, this is not of concern since our attempt is to extract "typical user session profiles". If we assume that the majority of a user's sessions follow a similar profile then clearly no difference is made. On the other hand, this notion of multiple user sessions enables us to better capture the situation when the same user displays a few different access patterns on this site. This approach will not work as well when multiple users from the same machine are accessing the site at the same time. However, this is likely a rare phenomenon given the proliferation of Desktops. Web caches cause another problem for our technique (like for all other related systems). We assume though that by appropriate use of the No cache pragma in HTTP/1.1, this problem can be avoided.

Computing The Dissimilarity Matrix

In the following paragraphs, we introduce the similarity measures between two user-sessions, $\mathbf{s}^{(k)}$ and $\mathbf{s}^{(m)}$, which we have recently proposed[24]. The measures attempt to incorporates both the structure of the site, as well as the URLs involved. We first consider the simple case where all URLs accessed in the sessions are assumed to be to be totally distinct. Then, we can simply use the cosine of the angle between $\mathbf{s}^{(k)}$ and $\mathbf{s}^{(l)}$ as a measure ($M_{1,kl}$) of similarity. This simply measures the number of common URLs accessed during the two sessions relative to the total number of URLs accessed in both sessions. It has the desirable properties that $M_{1,kk} = 1$, $M_{1,kl} = M_{1,lk}$, and $M_{1,kl} > 0, \forall k \neq l$. The problem with this similarity measure is that like the association rule based approaches, it ignores the similarity between URLs (as described in the introduction section). Moreover, cosine type measures tend to best use when the binary vectors are symmetric (i.e. not visiting a URL would be as significant as visiting one in terms of grouping sessions).

One possible approach to estimate similarity of URLs is to analyze their contents. However this itself is an open area of work in IR, and tends to be computationally expensive. This leads us to define a similarity measure on the structural URL level. We model the web site as a tree with the nodes representing different URLs – essentially the directory structure rooted at the server's *document root*, with links (such as redirects and aliases) explicitly brought in. Similarity between two URLs is assessed by measuring the overlap in the paths from the root of the tree to the corresponding nodes. Hence, we define the similarity between the i^{th} and j^{th} URLs as

$$S_u(i,j) = \min\left(1, \frac{|(p_i \cap p_j|}{\max(1, \max(|p_i|, |p_j|) - 1)}\right) \quad (1)$$

where p_i denotes the path traversed from the root node to the node corresponding to the i^{th} URL, and $|p_i|$ indicates the length of this path. Now the similarity between sessions is defined by measuring the "similar" URLs visited in the two sessions relative to the total number of URLs visited:

$$M_{2,kl} = \frac{\sum_{i=1}^{N_U}\sum_{j=1}^{N_U} s_i^{(k)} s_j^{(l)} S_u(i,j)}{\sum_{i=1}^{N_U} s_i^{(k)} \sum_{j=1}^{N_U} s_j^{(l)}} \quad (2)$$

For the special case when all the URLs accessed during session $s^{(k)}$ have zero similarity with the URLs accessed during session $s^{(l)}$, i.e., $S_u(i,j) = 0$ if $i \neq j$, $M_{2,kl}$ reduces to

$$M_{2,kl} = \frac{\sum_{i=1}^{N_U} s_i^{(k)} s_i^{(l)}}{\sum_{i=1}^{N_U} s_i^{(k)} \sum_{j=1}^{N_U} s_j^{(l)}}$$

and when the two sessions are identical, this value further simplifies to

$$M_{2,kk} = \frac{1}{\sum_{i=1}^{N_U} s_i^{(k)}}$$

which can be considerably small depending on the number of URLs accessed. This is unintuitive, because ideally the similarity should be maximal for two identical sessions. Besides identical sessions, this similarity will generally be underestimated for session pairs that share some identical URLs while the the unshared URLs have low similarity. In general for such sessions where the URL similarities are low, $M_{1,kl}$ provides a higher and more accurate session similarity measure. On the other hand, when the URL similarities are high, $M_{2,kl}$ is higher and more accurate. Therefore, we use [24] the maximum of M_1 and M_2 as our similarity measure. For the purpose of relational clustering, this similarity is mapped to the dissimilarity measure $d_s^2(k,l) = (1 - M_{kl})^2$. This dissimilarity measure satisfies the desirable properties: $d_s^2(k,k) = 0$, $d_s^2(k,l) >= 0, \forall k,l$, and $d_s^2(k,l) = d_s^2(l,k), \forall k,l$. We note here that our dissimilarity measure is not a *metric*. In particular, unlike a metric distance it is possible for two distinct sessions to have zero dissimilarity. This occurs whenever $\sum_{i=1}^{N_U}\sum_{j=1}^{N_U} s_i^{(k)} s_j^{(l)} S_u(i,j) = \sum_{i=1}^{N_U} s_i^{(k)} \sum_{j=1}^{N_U} s_j^{(l)}$, or equivalently $\sum_{j=1}^{N_U} s_i^{(k)} s_j^{(l)} S_u(i,j) = s_i^{(k)} \sum_{j=1}^{N_U} s_j^{(l)}$ for all $i = 1, \ldots, N_U$.

This is particularly true if the URL level similarities are 1 for all the URLs accessed in the two sessions. A typical example consists of the sessions {/courses/cecs345} and {/courses/cecs345/syllabus.html}. This property is actually desirable for our application, because we consider

these two sessions to fit the same profile. The session dissimilarity measure also violates the triangular inequality for metric distances in some cases. For instance, the dissimilarity between the sessions {/courses/cecs345/syllabus} and {/courses/cecs345} is zero. So is the dissimilarity between {/courses/cecs345} and {/courses/cecs401}. However, the dissimilarity between {/courses/cecs345/
syllabus} and {/courses/cecs401} is not zero (it is $1/4$). This illustrates another desirable property for profiling sessions which is that the dissimilarity becomes more stringent as the accessed URLs get farther from the root because the amount of specificity in user accesses increases correspondingly.

Clustering

As has been described earlier, clustering of sessions requires algorithms that can accept graded notions of similarity and overlap between clusters, and deal with relational data. Moreover, the algorithms need to be able to handle noise in the data. We have recently proposed a robust fuzzy clustering algorithm[25] which we use here.

Let $X = \{\mathbf{x}_i | i = 1, \ldots, n\}$ be a set of n objects. Let $r(\mathbf{x}_i, \mathbf{x}_j)$ denote the dissimilarity between object $\mathbf{x}_i$ and object $\mathbf{x}_j$. Let $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\}, \mathbf{v}_i \in X$ represent a subset of X with cardinality c, i.e., $\mathbf{V}$ is a c-subset of X. Let X_c represent the set of all c-subsets $\mathbf{V}$ of X. Each $\mathbf{V}$ represents a particular choice of prototypes for the c clusters in which we seek to partition the data. The Robust Fuzzy Medoids Algorithm (RFCMdd) minimizes the objective function:

$$J_m(\mathbf{V}; X) = \sum_{i=1}^{n} \sum_{i=1}^{c} u_{ij}^m \; r(\mathbf{x}_j, \mathbf{v}_i), \tag{3}$$

where the minimization is performed over all $\mathbf{V}$ in X_c. In (3), u_{ij} represents the fuzzy membership of $\mathbf{x}_j$ in cluster i. The membership u_{ij} can be defined heuristically in many different ways. We use the Fuzzy c-Means [10] membership model given by:

$$u_{ij} = \frac{\left(\frac{1}{r(\mathbf{x}_j, \mathbf{v}_i)}\right)^{1/(m-1)}}{\sum_{k=1}^{c} \left(\frac{1}{r(\mathbf{x}_j, \mathbf{v}_k)}\right)^{1/(m-1)}}, \tag{4}$$

where $m \in [1, \infty)$ is the "fuzzifier". The fuzzifier influences the degree of membership of a point in the cluster. This generates a fuzzy partition of the data set X in the sense that the sum of the memberships of an object $\mathbf{x}_j$ across all classes is equal to 1. Since u_{ij} is a function of the dissimilarities $r(\mathbf{x}_j, \mathbf{v}_k)$, it can be eliminated from (3), and this is the reason J_m is shown as a function of $\mathbf{V}$ alone.

Substituting the expression for u_{ij} in (4) into (3), we obtain:

$$J_m(\mathbf{V}; \mathbf{X}) = \sum_{i=1}^{n} \left(\sum_{i=1}^{c} (r(\mathbf{x}_j, \mathbf{v}_i))^{1/(1-m)} \right)^{1-m} = \sum_{j=1}^{n} h_j, \tag{5}$$

where

$$h_j = \left(\sum_{i=1}^{c} (r(\mathbf{x}_j, \mathbf{v}_i))^{1/(1-m)} \right)^{1-m} \tag{6}$$

is $1/c$ times the harmonic mean of the dissimilarities $\{r(\mathbf{x}_j, \mathbf{v}_i)) : i = 1, \ldots, c\}$ when $c = 2$. The objective function for the Robust Fuzzy c Medoids (RFCMdd) algorithm is obtained by modifying (5) as follows:

$$J_m^T(\mathbf{V}; \mathbf{X}) = \sum_{k=1}^{s} h_{k:n}. \tag{7}$$

However, the objective function in (7) cannot be minimized via the alternating optimization technique, because the necessary conditions cannot be derived by differentiating it with respect to the medoids. (Note that the solution space is discrete). Thus, strictly speaking, an exhaustive search over X_c needs to be used. However, following Fu's [12] heuristic algorithm for a crisp version of (3), we describe a fuzzy algorithm that minimizes (7).

In (7) $h_{k:n}$ represents the k-th item when $h_j, j = 1, \ldots, n$, are arranged in ascending order, and $s < n$. The value of s is chosen depending on how many objects we would like to disregard in the clustering process. This allows the clustering algorithm to ignore outlier objects while minimizing the objective function. For example, when $s = n/2$, 50% of the objects are not considered in the clustering process, and the objective function is minimized when we pick c medoids in such a way that the sum of the harmonic-mean dissimilarities of 50% of the objects is as small as possible.

The quadratic complexity of the algorithm arises because when looking to update the medoid of a cluster, we consider all n objects as candidates. In practice, the the new mediod is likely to be one that currently has a high membership in the cluster. Thus by restricting the search to say k objects with the highest membership in the cluster, the process can be made linear, i.e. $\mathcal{O}(kn)$, where k is a low integer. In that case, the complexity will be determined by the sorting operation required to find the smallest s (or equivalently the largest $n - s$) of the h_j's. This is a good result, considering that robust algorithms are typically very expensive.

The Robust Fuzzy c Medoids Algorithm (RCMdd)

Fix the number of clusters c, and the fuzzifier m;
Randomly pick initial set of medoids: $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_1, \ldots, \mathbf{v}_c\}$ from X_c;
$iter$ = 0;
Repeat
  Compute harmonic dissimilarities h_j for $j = 1, \ldots, n$ using (6);
  Sort h_j, $j = 1, \ldots, n$ to create $h_{j:n}$;
  Keep the objects corresponding to the first s $h_{j:n}$;
  Compute memberships for s objects:
  for $j = 1$ **to** s **do**
    for $i = 1$ **to** c **do**
      Compute $u_{ij:n}$ by using (4);
    endfor
  endfor
  Store the current medoids: $\mathbf{V}^{old} = \mathbf{V}$;
  Compute the new medoids:
  for $i = 1$ **to** c **do**
    $q = \underset{1 \le k \le s}{\text{argmin}} \sum_{j=1}^{s} u_{ij:n}^{m}\, r(\mathbf{x}_{k:n}, \mathbf{x}_{j:n})$
    $\mathbf{v}_i = \mathbf{x}_q$;
  endfor
  $iter = iter + 1$;
Until $\left(\mathbf{V}^{old} = \mathbf{V} \text{ or } iter = MAX\_ITER\right)$.

Notice that the algorithms as described assume that the number of clusters is known *a priori*, which is not the case here. This is a well known problem in clustering. We use a heuristic to automatically determine the number of clusters by initializing it to some large number, much larger than the expected (final) number of clusters. A SAHN type process is then used to hierarchically reduce the number of clusters. As we ascend up the hierarchy, we have to progressively increase the dissimilarity over which clusters will be merged. We note the change in this distance at each step, and assume the level at which the greatest change occurred has the right number of clusters.

3 Experimental Results

The user sessions are assigned to the closest clusters. This creates C clusters $\mathcal{X}_i = \left\{\mathbf{s}^{(k)} \in \mathcal{S} \mid d_{ik} < d_{jk}\ \forall j \neq i\right\}$, for $1 \le i \le C$.

The sessions in cluster $\mathcal{X}_i$ are summarized in a typical session "profile" vector $\mathbf{P}_i = \left(P_{i1}, \ldots, P_{i_{N_U}}\right)^t$. The components of $\mathbf{P}_i$ are URL weights which represent the number of access of a URL during the sessions of $\mathcal{X}_i$ as follows

$$P_{ij} = p\left(\mathbf{s}_j^{(k)} = 1 | \mathbf{s}^{(k)} \in \mathcal{X}_i\right) = \frac{|\mathcal{X}_{i_j}|}{|\mathcal{X}_i|}, \qquad (8)$$

where $\mathcal{X}_{i_j} = \left\{\mathbf{s}^{(k)} \in \mathcal{X}_i \mid s_j^{(k)} > 0\right\}$. The URL weights P_{ij} measure the significance of a given URL to the i^{th} profile. Besides summarizing profiles, the components of the profile vector can be used to recognize an invalid profile. This will have no strong access pattern, and all the URL weights will be low. We have created Java+JDBC based scripts to automate the process of creating typical session profiles using views in an Oracle backend.

We generated session profiles for several different logs obtained from servers at UMBC, CSM, U of Missouri etc. These logs ranged from a few hundred entries to tens of thousands of entries. Fig 1 shows the time required both by the clustering process, as well as the overall mining time (sessionizing + dissimilarity computation + clustering) for logs that represent one day to five days worth of accesses to the UMBC web server. This is also tabulated in Table 1. The clustering process revealed both obvious profiles (students enrolled in courses accessing course pages, visitors to a particular faculty's research page, visitors to UMBC's well known AgentsWeb site etc) as well as less obvious groupings. As an example, we saw that students enrolled in the UG AI course also seemed to visit the AgentsWeb pages. This could be because the AI course is often taught by Prof. Finin, who also maintains the AgentsWeb pages.

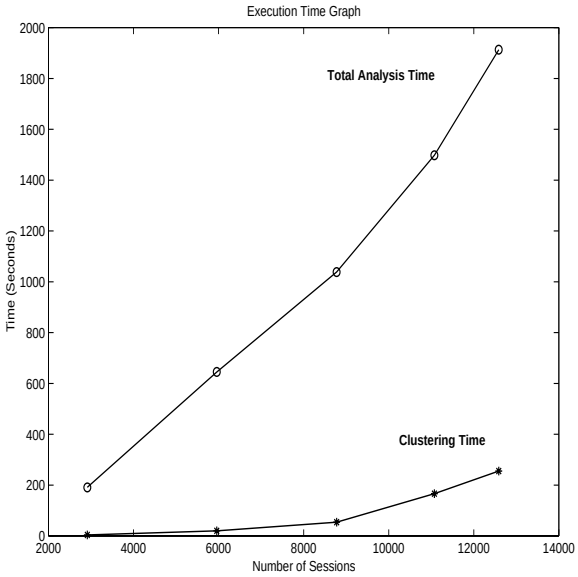

Figure 1: Execution time vs Number of Sessions in the log

As a comparison, we used a publicly available implementation of the *apriori* algorithm (http://fuzzy.cs.uni-magdeburg.de/ borgelt/) created by Christian Borgelt to generate association rules between the sessions. When a support of 10% was sought, no associations could be found. At

Days	Sessions	Accesses	Clustering Time	Total Exec Time
1	2913	53912	4	191.33
2	5956	113845	20	644.64
3	8777	162829	54	1038.38
4	11074	207223	166	1497.798
5	12586	234903	255	1912.67

Table 1: Time Measurements

lower values, a progressively larger number of rules were generated with varying confidence (50% – 80%). However, the largest itemset apriori could find, even with a support of 5% was of size 3. Note that this means that apriori could only find associations between groups of at most 3 sessions. In contrast, the clustering algorithm was able to find much larger coherent groups containing hundreds of similar sessions. We conjecture that this is because apriori cannot handle graded notions of similarity which are needed to group together similar (but not identical) sessions. The computation time needed by this implementation of apriori and our clustering algorithm were generally quite fast. Given our non-optimized implementation of RFCMdd, we were not surprised that as the log files grew larger (4-5 days worth of logs) apriori was faster. Moreoverver, the computation of the dissimilarity matrix between sessions creates an extra overhead for our approach. Note that the computation of the dissimilarity has components, specifically the overlap computation, that need to be done only once on a given site. Further, it is easily parallelizable. Thus the overhead involved in the distance matrix generation can be made acceptable, especially given that this mining process is off-line.

4 Conclusion

In this paper, we have presented a new approach for automatic discovery of user session profiles in web log data. The sessions extracted from real server access logs were clustered into typical user session profiles using a new robust fuzzy algorithm. The resulting clusters are evaluated subjectively and described by the significance of the components of a session "profile" vector which also summarizes the typical session represented by each cluster. A comparison with association rule based approach shows that the fuzzy clustering process creates better session profiles since it can group together "similar" (but not identical) sessions. In ongoing work, we are creating an apache module which will use the results of such offline analysis along with cookies to adapt a web site's content to the user accessing it. We are also creating a linear version of our clustering algorithm.

Acknowledgments

This work was partially supported by cooperative NSF awards (IIS 9801711 and IIS 9800899) to Joshi and Krishnapuram respectively, a grant from the Office of Naval Research (N00014-96-1-0439 to R. Krishnapuram), and an IBM faculty development award (to A. Joshi).

References

[1] Firefly, "http://www.firefly.com"

[2] A. Joshi, S. Weerawarana, and E. Houstis, "On disconnected browsing of distributed information," *Proc. Seventh IEEE Intl. Workshop on Research Issues in Data Engineering (RIDE)*, pp. 101-108, 1997.

[3] A. Joshi, C. Punyapu, P. Karnam, "Personalization and Asynchronicity to Support Mobile Web Access", *Proc. Workshop on Web Information and Data Management*, ACM 7^{th} Intl. Conf. on Information and Knowledge Management, November 1998.

[4] L. Terveen, W. Hill, and B. Amento, "PHOKAS - Asystem for sharing recommendations," *Comm. ACM*, **40**:3, 1997.

[5] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell, "WebWatcher: A learning apprentice for the world wide web," *AAAI Spring Symposium on Information Gathering from Heterogenous, Distributed Environments*, March, 1995.

[6] C. Shahabi, A. M. Zarkesh, J. Abidi and V. Shah, "Knowledge discovery from user's web-page navigation," *Proc. Seventh IEEE Intl. Workshop on Research Issues in Data Engineering (RIDE)*, pp. 20-29, 1997.

[7] R. Cooley, B. Mobasher, and J. Srivastava, "Web mining: Information and Pattern discovery on the World Wide Web," *Proc. IEEE Intl. Conf. Tools with AI*, Dec, 1997.

[8] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *Proc. of the 20th VLDB Conference*, pp. 487-499, Santiago, Chile, 1994.

[9] U. Fayad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, ed. *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, 1996.

[10] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

[11] H. Frigui and R. Krishnapuram, "Clustering by competitive agglomeration," *Pattern Recognition*, vol. 30, No. 7, pp. 1109-1119, 1997.

[12] K. S. Fu, *Syntactic Methods in Pattern Recognition*, Academic Press, New York, 1974.

[13] R. J. Hathaway, J. W. Davenport and J. C. Bezdek, "Relational duals of the c-means algorithms," *Pattern Recognition*, vol. 22, pp. 205-212, 1989.

[14] R. J. Hathaway and J. C. Bezdek, "NERF c-Means: Non-Euclidean relational fuzzy clustering," *Pattern Recognition*, vol. 27, No. 3, pp. 429-437, 1994.

[15] M. Perkowitz and O. Etzioni, "Adaptive Web sites: an AI Challenge" *Proc. Intl. Joint Conf. on AI*, 1997.

[16] M. Perkowitz and O. Etzioni, “Adaptive Web sites: Automatically Synthesizing Web Pages” *Proc. AAAI 98*, 1998.

[17] O.Zaiane and J. Han, “ WebML: Querying the World-Wide Web for Resources and Knowledge” *Proc. Workshop on Web Information and Data Management*, ACM 7^{th} Intl. Conf. on Information and Knowledge Management, November 1998.

[18] G. Arocena and A. Mendelzon, “WebOQL: Restructuring Documents, Databases, and Webs”, *Proc. IEEE Intl. Conf. Data Engineering ’98*, Orlando, February 1998

[19] S. Sen and R. N. Davé, “Clustering of Relational Data Containing Noise and Outliers,” *Proceedings of FUZZIEEE*, Anchorage, Alaska, May 1998, pp. 1411-1416.

[20] M.S. Chen, J.-S. Park and P. S. Yu, “Efficient Data Mining for Path Traversal Patterns,” *IEEE Trans. on Knowledge and Data Engineering*, Vol. 10, No. 2,pp. 209-221, April 1998.

[21] O.R. Zaiane, M. Xin, and J. Han, “Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs”, *Proc. Advances in Digital Libraries Conf. (ADL’98)*, Santa Barbara, CA, April 1998, pp. 19-29.

[22] Mark Nottingham, “Follow: A session based Log analyzing tool” , *http://www.pobox.com/˜mnot/follow/* .

[23] J. Han, “Data Mining”, in J. Urban and P. Dasgupta (eds.), *Encyclopedia of Distributed Computing*, Kluwer Academic Publishers, 1999.

[24] O. Nasraoui, H. Frigui, A. Joshi, and R. Krishnapuram, “Mining Web Access Logs Using Relational Competitive Fuzzy Clustering”, in Proc. *Eight International Fuzzy Systems Association World Congress - IFSA 99*, Taipei, August 99.

[25] Krishnapuram, R., Joshi, A. and Yi, L., A Fuzzy Relative of the k-Medoids Algorithm with Application to Web Document and Snippet Clustering, in Proc. *IEEE Intl. Conf. Fuzzy Systems - FUZZIEEE99*, Korea, 1999.

[26] Zhang, T., Ramakrishnan, R. and Livny, M., BIRCH: A New Data Clustering Algorithm and its Applications, *Data Mining and Knowledge Discovery Journal*, **1:**2, 1997.

[27] R. T. Ng and J. Han, Efficient and Effective Clustering Methods for Spatial Data Mining, *Proc. 20th VLDB Conference*, pp. 144-155, 1994.

[28] A. Joshi, and R. Krishnapuram, “Robust Fuzzy Clustering Methods to Support Web Mining”, *Proc. Workshop in Data Mining and knowledge Discovery*, SIGMOD, pp. 15-1 – 15-8, 1998.

[29] R. Krishnapuram and J. M. Keller, “A Possibilistic Approach to Clustering”, *IEEE Trans. Fuzzy Systems*, **1:**2, pp 98–110, 1993.

[30] R. Krishnapuram and J. M. Keller, “The Possibilistic c-Means Algorithm: Insights and Recommendations”, *IEEE Trans. Fuzzy Systems*, **4:**3, pp 385-393, 1996.

[31] R. N. Davé and R. Krishnapuram, “Robust Clustering Methods: A Unified View”, *IEEE Trans. Fuzzy Systems*, **5:**2, pp 270–293, 1997.

[32] J. Kim, R. Krishnapuram and R. N. Davé, “Application of the Least Trimmed Squares Technique to Prototype-Based Clustering”, *Pattern Recognition Letters*, **17**, pp 633–641, 1996.