On Mining Web Access Logs

Anupam Joshi, Karuna Joshi
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County, Baltimore, MD 21250
{joshi, kjoshi1}@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. The standard K-Means algorithm was used to cluster users' traversal paths in [6] . However, it is not clear how the similarity measure was devised and whether the clusters are meaningful. 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.

It is important to mention that so far, most efforts have relied on relatively simple techniques which can be inadequate for real user profile data since they are not resilient to the noise typically found in user traversal patterns. Web mining involves data that is mildly to severely corrupted with noise. outliers and incomplete data can easily occur in the data set due to a wide variety of reasons inherent to web browsing and logging. Moreover, the noise contamination rate and the scale of the data is rarely known in advance. For example, consider the situation where we are analyzing log entries to discover typical information access patterns. Clearly, there is a significant percentage of time (sometimes as large as 20-30 percent) that a user is simply "browsing" the web and does not follow any particular 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. Hence, there is a need for robust methods that are free of any assumptions about the noise contamination rate and scale in this task.

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 varietyand 2 into the same "large" itemset, even though it is fairly clear to a human observer from the context (and 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. We note that some researchers[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.

In the absence of any *a priori* knowledge about the possible patterns, unsupervised classification or clustering methods seem to be ideally suited 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 of way. 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. More importantly, it can be used by software to make the web site itself dynamic and adaptive. In this work, we define the notion of a "user session" as being a temporally compact sequence of web accesses by a user. We define a new distance measure between two web sessions that captures the organization of a web site and similarities between URLs. This organizational information is inferred directly from the URLs.

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 [10], particularly along the overlapping borders. Also, it is necessary for the clustering process to work with relational[1] data because it is intuitively and conceptually easier to describe the relation or similarity between two objects (i.e web sessions) than to map them to numerical features in a manner that makes (Minkowski) distances between them meaningful. This problem is particularly acute when the data contains non-numeric fields, as do most web mining tasks. Hence, clustering the user sessions should be tackled by exploiting inter-session similarities within a relational framework. This immediately rules out the use of fast algorithms developed by the data mining community such as CLARANS[27] and Birch[26], which only deal with object data.

The rest of the paper is organized as follows, where the sections follow different phases of the knowledge discovery process [9] of categorization of user profiles. In Section 2, we define the similarity between user sessions. In Section 3, we describe the robust fuzzy clustering algorithm. In Section 4, we define quantitative measures to help us in the interpretation and evaluation of the results of mining the access log data, and present our experimental results, as well as comparisons with association rule based techniques. We conclude with a discussion of our ongoing work.

2 Sessionizing access log data

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. First, we filter out log entries that are not germane for our task. These include entries that: (i) result in any error (indicated by the error code), (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.

Next, 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. Thus, the i^{th} user session is encoded as an N_U-dimensional binary attribute vector $\mathbf{s}^{(i)}$ with the property

$$s_j^{(i)} = \begin{cases} 1 & \text{if the user accessed the } j^{th} \text{ URL during the } i^{th} \text{ session} \\ 0 & \text{otherwise} \end{cases}$$

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

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

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 works in this area). We assume though that by appropriate use of the No cache pragma in HTTP/1.1, this problem can be avoided.

2.1 Computing The Dissimilarity Matrix

In the absence of any a priori knowledge, an unsupervised classification or clustering method seems to be ideally suited to partition the user sessions. There are two major classes of clustering techniques, those that work with "object data" or feature vectors, and those that work on "relational data" (similarities or dissimilarities between the data). Even though the first class of clustering algorithms is more popular and has received a lot of attention, it is not suitable for clustering user sessions as explained earlier.

We chose the relational approach to clustering since our data (sessions) are not numeric in nature. This approach requires the definition and computation of the dissimilarity/similarity between all session pairs (i.e., the relation matrix) prior to the clustering process. 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 the individual attributes or URLs accessed in the sessions are totally independent and the structure of the site is ignored. Then, we can simply use the cosine of the angle between $\mathbf{s}^{(k)}$ and $\mathbf{s}^{(l)}$ as a measure of similarity

$$M_{1,kl} = \frac{\sum_{i=1}^{Nu} \mathbf{s}_i^{(k)} \mathbf{s}_i^{(l)}}{\sqrt{\sum_{i=1}^{Nu} \mathbf{s}_i^{(k)}} \sqrt{\sum_{i=1}^{Nu} \mathbf{s}_\mathbf{i}^{(\mathbf{l})}}} \tag{1}$$

It can be seen that $M_{1,kl}$ simply measures the number of identical URLs accessed during the two sessions relative to the 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 it completely ignores the hierarchical organization of the web site, which will adversely affect the ability to capture correct profiles. For example, the session pair {/courses/cmsc201} and {/courses/cmsc341}, as well as the session pair {/courses/cmsc341} and {/research/grants} will receive a 0 similarity score according to S_1. However, it is evident that the first two sessions are more similar than the second two, because both users in the first sessions seem to be interested in courses. Similarly, one would expect the sessions {/courses/cmsc341/projects/proj1} to be more similar to {/courses/cmsc341/projects} than to {/courses/cmsc421} because there is more overlap between the URLs in the first two sessions along the directory hierarchy tree. This leads us to define a similarity measure on the structural URL level that will be used in the computation of the similarity at the session level.

We model the web site as a tree with the nodes representing different URLs. The nodes are essentially the directory structure rooted at the server's *document root*, with links (such as redirects and aliases) explicitly brought in. An edge connects one node to another if the URL corresponding to the latter is hierarchically located under that of the former, for example {/courses} and {/courses/cecs345}. The root of the tree (the node with no incoming edges) corresponds to the document root (/) of the server. Taking into account the syntactic representation of two URLs, their similarity is assessed by comparing the location of their corresponding nodes on the tree. This is done by comparing the paths from the root of the tree to the two nodes. Hence, we define the "syntactic" similarity between the i^{th} and j^{th} URLs as

$$S_u(i,j) = \min\left(1, \frac{|(p_i \cap p_j|}{\max(1, \max(|p_i|, |p_j|) - 1)}\right) \tag{2}$$

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 or the number of edges included in the path. Note that this similarity which lies in $[0,1]$ basically measures the amount of overlap between the paths of the two URLs. Now the similarity on the session level which incorporates the syntactic URL similarities is defined by correlating all the URL attributes and their similarities in two sessions as follows

$$M_{2,kl} = \frac{\sum_{i=1}^{N_U} \sum_{j=1}^{N_U} s_i^{(k)} s_j^{(l)} S_u(i,j)}{\sum_{i=1}^{N_U} s_i^{(k)} \sum_{j=1}^{N_U} s_j^{(l)}} \tag{3}$$

Unlike M_1, this similarity uses soft URL level similarities that vary between 0 and 1 (depending on how similar they are), instead of the hard similarities 0 or 1 (depending on whether they are different or identical respectively). 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 means that this similarity measure will be rather unintuitive, because ideally the similarity should be maximal for two identical sessions. Besides identical sessions, this similarity will generally be underestimated for session pairs who share some identical URLs while the rest of the unshared URLs have low syntactic similarity. In general for such sessions where the syntactic URL similarities are low, $M_{1,kl}$ provides a higher and more accurate session similarity. On the other hand, when the syntactic URL similarities are high, $M_{2,kl}$ is higher and more accurate. Therefore, we define [24] a new similarity between two sessions that takes advantage of the desirable properties of M_1 and M_2 as follows

$$M_{kl} = \max(M_{1,kl}, M_{2,kl}) \tag{4}$$

For the purpose of relational clustering, this similarity is mapped to the dissimilarity measure $d_s^2(k,l) = (1 - M_{kl})^2$. Note that squaring the complement of the similarity has the effect of amplifying the difference between similar and different sessions. This dissimilarity measure satisfies the desirable properties: $d_s^2(k,k) = 0$, $d_s^2(k,l) >= 0, \forall k,l$, and $d_s^2(k,l) = d_s^2(l,k), \forall k,l$. However, 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. Hence, the proposed dissimilarity measure fits our subjective criteria of session similarity.

3 Algorithms for Robust Fuzzy 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 therefore chosen to use two new algorithms that we have devised for web mining tasks[25].

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 Fuzzy Medoids Algorithm (FCMdd) minimizes the objective function:

$$J_m(\mathbf{V}; X) = \sum_{i=1}^{n} \sum_{i=1}^{c} u_{ij}^m \, r(\mathbf{x}_j, \mathbf{v}_i), \tag{5}$$

where the minimization is performed over all $\mathbf{V}$ in X_c. In (5), u_{ij} represents the fuzzy [10], or possibilistic [29] [14] membership of $\mathbf{x}_j$ in cluster i. The membership u_{ij} can be defined heuristically in many different ways. For example, we can use the Fuzzy c-Means [10] membership model given by:

$$u_{ij} = \frac{\left(\frac{1}{r(\mathbf{x}_j, \mathbf{v}_i)}\right)^{1/(m-1)}}{\sum_{k=1}^{c} \left(\frac{1}{r(\mathbf{x}_j, \mathbf{v}_k)}\right)^{1/(m-1)}}, \tag{6}$$

where $m \in [1, \infty)$ is the "fuzzifier". Another possibility is to use

$$u_{ij}^m = \frac{\exp\{-\beta r(\mathbf{x}_j, \mathbf{v}_i)\}}{\sum_{k=1}^{c} \exp\{-\beta r(\mathbf{x}_j, \mathbf{v}_k)\}}. \tag{7}$$

The above equations generate 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. If we desire possibilistic memberships [29], we could use

$$u_{ij} = \frac{1}{1 + \frac{r(\mathbf{x}_j, \mathbf{v}_k)}{\eta_i}}, \tag{8}$$

or [30]

$$u_{ij} = \exp\left(-\frac{r(\mathbf{x}_j, \mathbf{v}_i)}{\eta_i}\right). \tag{9}$$

Since u_{ij} is a function of the dissimilarities $r(\mathbf{x}_j, \mathbf{v}_k)$, it can be eliminated from (5), and this is the reason J_m is shown as a function of $\mathbf{V}$ alone. When (5) is minimized, the $\mathbf{V}$ corresponding to the solution does generate a fuzzy or possibilistic partition via an equation such as (6). However, the objective function in (5) 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 (5), we describe a fuzzy algorithm that minimizes (5).

The Fuzzy c-Medoids Algorithm (FCMdd)

Fix the number of clusters c;
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 memberships:
    for $i = 1$ **to** c **do**
        for $j = 1$ **to** n **do**
            Compute u_{ij} by using (6), (7),(8) or (9).
        endfor
    endfor
    Store the current medoids: $\mathbf{V}^{old} = \mathbf{V}$;
    Compute the new medoids:
    for $i = 1$ to c **do**
        $q = \underset{1 \le k \le n}{\text{argmin}} \sum_{j=1}^{n} u_{ij}^m \, r(\mathbf{x}_k, \mathbf{x}_j)$
        $\mathbf{v}_i = \mathbf{x}_q$;
    endfor
    $iter = iter + 1$;
Until $\left(\mathbf{V}^{old} = \mathbf{V} \text{ or } iter = MAX\_ITER\right)$.

Note that 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 (which can be identified with complexity $\mathcal{O}(kn)$), the process can be made linear, i.e. $\mathcal{O}(kn)$, where k is a low integer.

It is well-known that algorithms that minimize a Least-Squares type objective function are not robust[31]. In other words, a single outlier object could lead to a very unintuitive clustering result. To overcome this problem, we have developed a variation of of FCMdd that is is based on the Least Trimmed Squares idea [32].

To design an objective function for a robust version of FCMdd based on the Least Trimmed Squares idea, we use the membership function in (6). Substituting the expression for u_{ij} in (6) into (5), we obtain:

$$J_m(\mathbf{V}; \mathbf{X}) = \sum_{i=1}^{n} \left(\sum_{i=1}^{c} (r(\mathbf{x}_j, \mathbf{v}_i))^{1/(1-m)} \right)^{1-m} = \sum_{j=1}^{n} h_j, \tag{10}$$

where

$$h_j = \left(\sum_{i=1}^{c} (r(\mathbf{x}_j, \mathbf{v}_i))^{1/(1-m)} \right)^{1-m} \tag{11}$$

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 f or the Fuzzy c Trimmed Medoids (FCTMdd) algorithm is obtained by modifying (10) as follows:

$$J_m^T(\mathbf{V}; \mathbf{X}) = \sum_{k=1}^{s} h_{k:n}. \tag{12}$$

The Fuzzy c Trimmed Medoids Algorithm (FCTMdd)

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 (11);
    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 (6);
        endfor
    endfor
    Store the current medoids: $\mathbf{V}^{old} = \mathbf{V}$;
    Compute the new medoids:
    for $i = 1$ **to** c **do**
        $q = \underset{1 \leq k \leq s}{\text{argmin}} \sum_{j=1}^{s} u_{ij:n}^m \, r(\mathbf{x}_{k:n}, \mathbf{x}_{j:n})$
        $\mathbf{v}_i = \mathbf{x}_q$;
    endfor
    $iter = iter + 1$;
Until $\left(\mathbf{V}^{old} = \mathbf{V} \text{ or } iter = MAX\_ITER\right)$.

In (12) $h_{k:n}$ 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 objective function in (12) cannot be minimized easily. However, we can design the heuristic algorithm given above. Again, we caution the reader that the above algorithms can converge to a local minimum. It is good to try many random initializations to increase the reliability of the results. Interestingly the worst-case complexity of this algorithm still remains O(n^2), and the complexity of the medoid update can be made linear as in the case of FCMdd. 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.

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. A heuristic is used to automatically determine the number of clusters. after 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.

4 Experimental Results

4.1 Measures for Evaluation of Results

We interpret the results of clustering the user session relational data are using the following quantitative measures. First, 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 $1 \leq i \leq C$.

After the assignment of user sessions to the automatically determined number (C) of clusters, 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

$$P_{ij} = p\left(\mathbf{s}_j^{(k)} = 1 | \mathbf{s}^{(k)} \in \mathcal{X}_i\right) = \frac{|\mathcal{X}_{i_j}|}{|\mathcal{X}_i|}, \tag{13}$$

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 which has no strong or frequent access pattern. For such a profile, all the URL weights will be low. Several classical cluster validity measures can be used to assess the goodness of the partition such as intra and inter cluster distances.

4.2 Preprocessing and Database Creation

To generate the clusters for the web logs, we first generated the sessions data. Next, we generated session clusters by applying the FCMdd and FCTMdd algorithms described earlier. The cluster file listed each session and the cluster that it belongs to. Using the sessions, URLs in sessions and the clustering information, we populated an Oracle database that would help us in analyzing our clusters.

The database dealing with sessions consists of three tables. The URL table stores the URL description along with the Unique ID (URL_NO) that is generated by the program. The Session Table contains data pertaining to the Domain that identifies the session, the cluster number to which the session belongs and the frequency of the domain. SessionNo is the unique identifier for each session. We know that many URLs (say N) can be accessed in a single session and also one URL can belong in multiple sessions (say N). To incorporate the N:N relationship between URLs and Sessions, we split the relation ship into a 1:N relation by introducing a new table SES_URL table. This table contains the primary identifiers of both the Session and URL tables, which constitute the composite primary key of the table. This table was populated with the output from the clustering program that contained a listing of all Sessions and the URL numbers associated with each session.

Next, we proceeded to create three views from the tables that we had populated above. X_i View displays the cardinality (total number of sessions) of each cluster obtained. X_{ij} View displays each URL in the log along with Cluster number and the total number of sessions in the cluster that contain the URL. The third view, Degree View, was created from the two views above. It displays each URL in the log along with Cluster number, X_i, X_{ij} and the Degree Measure, X_{ij}/X_i (13).

4.3 Experiment Results

We generated clusters using both the algorithms 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. We illustrate the result from two of them here. For one study, log entries of the access log pertaining to one of the author's homepage over a period of two days were used. In another study, log entries of UMBC CSEE server over a several hour period in the morning were used. The cluster numbers displayed are simply labels assigned by the program.

While analyzing, we did not consider clusters that had less than 3 user sessions as relevant. In all the clusters {/url} and {/url/} were regarded the same and counted only once. In the CSEE Logs, URLs {/courses/undergraduate/CMSCcourse_no} and {/courses/undergraduate/course_no} point to the same page and were counted only once. Separate tables were created in the Oracle database for each log studied. The Degree View for each log was used in analyzing the results. Tables below tabulate the clusters that were found for the experiments. Observations made from these experiments are also listed below. Note that while the clusters produced often group together pages dealing with similar content, the algorithms do capture cases where the same traversal patterns cover pages with different content. For example, cluster 10 in CSEE pages for FCTMdd which captures access to a variety of course pages.

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 create association rules between the sessions. When a support of 10% was sought, no associations could be found. At lower values, a progressively larger number of rules were generated with fairly high confidence (¿80%). However, the largest itemset apriori could find, even with a support of 2% was of size 5. Note that this means that apriori could only find associations between groups of at most 5 sessions. In contrast, the clustering algorithm was able to find much larger coherent group of sessions. As explained in the introduction, this is expected since apriori cannot handle graded notions of similarity which are needed to group together similar (but not the same) sessions. The computation time needed by this implementation of apriori and our clustering algorithm were generally similar, and for sessions of several thousand entries, less than a second of CPU time was taken on a moderately loaded multiprocessor SGI machine. However, the computation of the dissimilarity matrix between sessions creates an extra overhead for our approach.

4.3.1 CSEE server Logs analysis

The following summarizes observations made on clustering results presented in tables 1 and 2. Note that in order to make the tables fit into a page and avoid visual clutter, we have used a single entry *url/** to represnt the fact that the traversal path includes the url and others in its subdirectories in the document space. Correspondingly, in the "degree" entry of the table, we combine the URL weights of all the URLs so grouped together. This explains why the value is greater than 1 for some of the entries.

FCTMdd Algorithm

- Clusters 1 (*/˜sli2/cube*),14(*/˜sli2/plot*) and 24(*/˜sli2/tetris*)correspond to users interested in the computer games in a user's (sli2) page.
- Clusters 0 (201 Course), 6(Lecture 12 of 201) represent users who want to access the CMSC 201 course pages. Specifically Cluster 0 is made up of users who wish to access 201 course page in general, while Cluster 6 consists of user sessions that were only accessing the web pages of Lecture 12 of this course.

- Clusters 8(CMSC331), 10(Courses), 20(CMSC104) and 12(CMSC421) represent users who want to access the pages of various courses offered by the department.
- Cluster 23 (/kqml/mail/kqml/1997) contains user sessions that accessed Mail archives listed in the KQML page.
- Cluster 4 (Agents) contains user sessions that accessed the Agent project pages.
- Clusters 2 (/˜graddir/CSEE) and 9 (CSEE Graduate) contains user sessions that accessed the Graduate Admissions pages.
- Clusters 5(/˜thurston), 11(/˜squire), 13(/˜kalpakis), 21 (/˜lomanaco) correspond to user sessions that accessed individual users home pages.
- Clusters 3, 15, 16,18,19, 22 have too small a cardinality to be relevent.
- Other clusters did not represent any conclusive group of URLs and have low URL weights for all URLs.

FCMdd Algorithm

- Clusters 0 (/˜sli2/), 16 (/˜sli2/cube) and 17 (/˜sli2/plot) correspond to users interested in the computer games in a user's (sli2) page.
- Clusters 1 (Courses Page), 2(201 Course), 3(Lecture 12 of 201) and 4(courses) represent users who want to access the CMSC 201 course pages. Specifically Cluster 2 is made up of users who wish to access 201 course page in general, while Cluster 1 represents hits to the lecture pages of this course and Cluster 3 consists of user sessions that were only accessing the web pages of Lecture 12 of this course. Cluster 4 also contained accesses to the CMSC 104 course and the main /courses directory.
- Clusters 6 (401 Course) and 4(courses) represent users who want to access the CMSC 401 course pages. Cluster 4 also contained accesses to the CMSC 201 page and the main /courses directory and Cluster 6 also contained accesses to CMSC 421 page.
- Cluster 8 (461 Course) and 4(courses) represent users who want to access the CMSC 461 course pages for the Spring session.
- Clusters 15 (/kqml/mail/kqml/1997) and 19 (/kqml/mail/) contains user sessions that accessed Mail archives listed in the KQML page.
- Clusters 14 (/agents/kse , /agents/web/) and 20 (/agents/) contains user sessions that accessed the Agent project pages.
- Clusters 10 (/˜squire), 21(/˜kalpakis), 23(/˜thurston) and 24(/˜mikeg, /˜hchen4) correspond to user sessions that accessed individual users home pages. However, these clusters contain only 5 (or less) user sessions.
- Clusters 5, 7, 11, 12, 13, 18 and 22 have too small a cardinality to be included in the study.

From the above two experimental studies we observe, that though both the algorithms generated almost the same number of clusters, FCTMdd algorithm generated more compact clusters for the same logs. For example, cluster 1 in the FCMdd experiment result was a grouping of a variety of URL groups, whereas the FCTMdd results have them grouped into separate clusters. Similarly, clusters 15 and 19 in FCMdd represent traversal patterns on KQML related pages, with some other component. FCTMdd groups together all KQML related traversals into cluster 23, and moves the other accesses (to oracle help pages, pages for user lomonaco) into a separate cluster 21.

4.3.2 Author's web page Logs analysis

The following summarizes observations made on clustering results presented in tables 3 and 4.
FCTMdd Algorithm

- Cluster 0 (DBrowsing Project) corresponds to users who are interested in the DBrowsing project.
- Cluster 1 (General Browser) corresponds to users who are general browsers and are not looking for any specific information.

FCMdd Algorithm

- Cluster 0 (General Browser) corresponds to users who are general browsers and are not looking for any specific information.
- Cluster 1 (Web Mining) represents users who want to access the Web Mining Project pages.
- Cluster 2 (Courses) corresponds to users who access mainly the course page. They also navigate to research and publications sections and traverse the main page quite often.
- Cluster 3 (CMSC491 course) corresponds to users who want to access the CMSC 491 course page.

From the above two experimental studies we observe, that FCTMdd algorithm generates only two clusters compared to the four by FCMdd. Thus it picked up the two major traversal paths - general Browsers who look at the home page, course page and research pages; and browsers only interested in the authors research on mobile computing. FCMdd was not able to handle noise as effectively, and ended up splitting into more clusters.

5 Conclusion

In this paper, we have presented a new approach for automatic discovery of user session profiles in web log data. We defined the notion of a "user session" as being a temporally compact sequence of web accesses by a user. A new similarity measure to analyze session profiles is presented which captures both the individual URLs in a profile as well as the structure of the site. The sessions extracted from real server access logs were clustered into typical user session profiles using two new fuzzy algorithms with desirable properties. The resulting clusters are evaluated subjectively and described by the significance of the components of a session "profile" vector which also summarizes the typical sessions in 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 a system which will use the results of such offline analysis along with cookies to adapt a web sites index page to the user accessing it.

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", to be presented at the *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, to be presented at *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.

Table 1: CSEE Logs Analysis using FCTMdd Algorithm

Cluster	**Cardinal**	**# URLs**	**URLs**	**Degree**
0 - CMSC201	28	155	{/courses/undergraduate/CMSC201/Spring99/*} {/agents/*}	2.46 1.9
1 - Sli2 cube	13	38	{/˜sli2/cube/*}	14.07
2 - graddir/CSEE	8	17	{/˜graddir/CSEE/*} {/˜ebert/*} {/}	1 0.75 0.625
_3_	_1_	_6_	_{/%7Eirao1/*}_	
4-Agents	10	17	{/agents/*} Other URLs	2.2 < 0.3
5-˜thurston	6	7	{/˜thurston/*}	1.5
6- CMSC 201	7	25	{/courses/undergraduate/201/Spring99/lectures/*} {/courses/undergraduate/201/Spring99/*}	1.8 3.2
7 - NONE	4	28	{/courses/undergraduate/CMSC201/Spring99/*} {/kqml/mail/}	2 1.25
8 - CMSC 331	8	23	{/courses/undergraduate/CMSC331/Spring99/} {/˜khu1/}	1.87 1.25
9 - graduate	3	15	{/˜graddir/CSEE/graduate/} OR {/˜graddir/}	4
10 - CMSC 104	11	33	{/courses/undergraduate/CMSC104/} Other course pages	1.1 < 0.4
11 - /s̃quire/	3	13	{/s̃quire/} {/courses/undergraduate/341/Spring99/}	4.33 1
12 - CMSC 421	3	19	{/courses/undergraduate/CMSC421/Spring99/elm/}	5.66
13 - /˜kalpakis	3	6	{/˜kalpakis/441sp99/} {/˜kalpakis/} or {/˜kalpakis}	3 0.67
14 - /˜sli2/plot/	6	40	{/˜sli2/plot/} Other URLs	6 < 0.8
_15_	_2_	_7_	_{/˜finin/*}, {/cikm}_	
_16_	_2_	_19_	_{/courses/undergraduate/CMSC461/*}_	
17 - NONE	9	24	{/agents/*} Other URLs	1 <0.5
_18_	_1_	_9_	_{/cikm/1994/iia/papers/*}_	_6_
_19_	_2_	_33_		
20 - CMSC 104	4	17	{/courses/undergraduate/CMSC104/Spring99/*}	6.5
21 - /˜lomanaco	29	330	{/˜lomonaco/*} {/%7Esli2/cube/*} {/help/oracle8/server803/*}, Others	3.34 0.65 0.41
_22_	_1_	_1_		
23 - KQML Mail	4	266	{/kqml/mail/kqml/1997/*} {/kqml/papers/*} {/kqml/software/kats/*}	53.75 5.75 1.75
24 - tetris	20	12	{/˜sli2/tetris/*}, {/˜sli2/*}	2.55
25 - NONE	5	12	{/courses/undergraduate/341/Spring99/abaumg1/*} Other URLs	0.6 <0.4

Table 2: CSEE Logs Analysis using FCMdd Algorithm

Cluster	**Cardinal**	**URLs**	**URLs**	**Deg**
0 - /~sli2/	30	32	{/~sli2/directory.htm} {/~sli2/tetris/*} {/~brostrom/} or {/~brostrom/431/*}	0.567 1.03 0.15
1 - Courses, Graduate Program, Agents	22	121	{/courses/undergraduate/CMSC201/Spring99/*} {/~graddir/CSEE/*} {/courses/undergraduate/CMSC341/Spring99/*} {/agents/*} {/agentslist/} or {/agentslist/archive/*}	1.12 0.77 0.135 1.84 0.495
2 - CMSC 201	6	28	{/courses/undergraduate/201/Spring99/lectures/*} /~tchen/*	5.177 0.635
3 - Lecture 12 of CMSC201	6	35	{/courses/undergraduate/201/Spring99/lectures/lec12/*} {/courses/undergraduate/201/Spring99/lectures/*} Other URLs	2.34 0.668 < 0.5
4 - Courses 104, 201	10	37	{/courses/} or {/courses/*} {/courses/undergraduate/201/Spring99/*} {/courses/undergraduate/104/Spring99/*}	1.3 1.4 0.9
5	*2*	*5*	*/~squire/*, /~dasgupta/ , /~elm*	
6 - 104, 421	8	32	{/courses/undergraduate/CMSC104/Spring99/*} {/courses/undergraduate/CMSC421/Spring99/*}	2 1.25
7	*1*	*7*	*/courses/undergraduate/104/*	
8 - CMSC 461, 331 Oracle help, /~khu1	18	61	{/courses/undergraduate/CMSC461/Spring99/*} {/~khu1/*} {/courses/undergraduate/CMSC331/Spring99/*} {/help/oracle8/*}	0.83 0.55 0.5 0.33
9	3	27	{/courses/undergraduate/CMSC201/Spring99/*}	
10 - /~squire/	5	17	{/~squire/*} {/agents/news/*}	1.2 0.8
11	*1*	*2*	*/~yan/**	
12	*1*	*3*	*/~thurston/**	
13	*2*	*2*		
14 - agents	4	14	{/agents/*}	3.75
15 - KQML Mail	31	619	{/kqml/mail/kqml/1997/*} {/kqml/papers/*} and {/kqml/*} {/~lomonaco/*} {/courses/undergraduate/*} {/help/oracle8/*}	6.93 1.1 3.32 2.26 0.58
16 - /~sli2/cube	11	31	{/~sli2/cube/*} {/~sli2/cube/Cube*} (classes, java files)	7.5 8.1
17 /~sli2/plot/	4	50	{/~sli2/plot/*} (classes, java files) {/courses/undergraduate/CMSC104/Spring99/*}	8.25 2.75
18	*2*	*9*	*/courses/undergraduate/104/**	
19 - kqml mail	6	27	{/kqml/mail/*} or {/agents/kqml/mail/*} Other URLs	1.16 < 0.334
20 - agents	7	10	{/agents/} or {/agents/*}	1.86
21 - /~kalpakis	3	6	{/~kalpakis/441-sp99/*} or {/~kalpakis/}	3.34
22	*1*	*2*	*/agents/news/ , /pub/agents/*	
23 - /~thurston	5	5	{/~thurston/*}	1.4
24 - /~mikeg, /~hchen4	4	10	{/~mikeg/*} {/~hchen4/*}	1 1

Table 3: Author's web page Logs Analysis using FCTMdd Algorithm

Cluster	**Cardinal**	**# URLs**	**URLs**	**Degree**
0 - dbrowse	3	4	{ /~ajoshi/dbrowse/*}	2.33
1 - General	138	39	{ /~ajoshi/courses/cmsc491w/*} {/~ajoshi/} { /~ajoshi/dbrowse/*} { /~ajoshi/web-mine/*}	1.14 0.62 0.32 0.25

Table 4: Author's web page Logs Analysis using FCMdd Algorithm

Cluster	**Cardinal**	**# URLs**	**URLs**	**Degree**
0 - General Browser	42	15	{/~ajoshi/} { /~ajoshi/dbrowse/*} { /~ajoshi/web-mine/*} Other URLs	0.833 0.62 0.21 < 0.2
1 - Web Mining	8	13	{ /~ajoshi/web-mine/*} { /~ajoshi/dbrowse/*}	1.375 0.625
2 - Courses	69	33	{ /~ajoshi/courses/cmsc491w/*} {/~ajoshi/} { /~ajoshi/course.html} { /~ajoshi/resch/*} (publications) { /~ajoshi/dbrowse/*} Other URLs	1.30 0.725 0.377 0.33 0.32 < 0.3
3 - CMSC491 course	22	4	{/~ajoshi/courses/cmsc491w/*} Other URLs	1.9 < 0.04