Technical Report TR-CS-00-05

On Creating Adaptive Web Servers Using Weblog Mining

Tapan Kamdar and Anupam Joshi

Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
1000 Hilltop Circle
Baltimore, MD 21250

20 November 2000

On Creating Adaptive Web Servers Using Weblog Mining

Tapan Kamdar and Anupam Joshi
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County, Baltimore, MD 21250
{kamdar, joshi}@cs.umbc.edu

Abstract

Personalization of content returned from a web site is an important problem in general, and affects e-commerce and e-services in particular. Targeting appropriate information or products to the end user can significantly change (for the better) the users experience on a web site. 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 are ideally suited to analyze the semi-structured log data of user accesses by examining user sessions. User access profiles are generated by clustering user sessions on the basis of pair-wise dissimilarities using a robust fuzzy clustering algorithm.We present a system that mines the logs to get profiles and uses them to automatically generate a web page containing URLs the user might be interested in. We also evaluate the efficacy of sessionizing the information with and without the use of cookies.

1 Introduction

The WWW is a distributed and growing collection of large amount of information and is becoming the apocryphal *vox populi*. This is causing an information overload for users, where they spend a significant portion of their time looking for appropriate information. *Personalization* is thus, a prime requirement for the evolving web infrastructure.

Personalization can either be done via information brokers or in an *end to end* manner by making web sites adaptive. Instances of information brokers include search engines, infomediaries and recommender systems. Commercial infomediaries like Yodlee.com provide e-Personalization solutions that enable consumers to access all their personal online accounts with one-click. Recommender systems are based on the principle of broker entities, which looks for information relevant to a user. One of the earliest such systems was Firefly [1], which attempted to provide CDs that best match a user's professed interests. More recently, systems such as PHOAKS [23] and our own W^3IQ[13, 12] have sought to use cooperative information retrieval techniques for personalization.

End–end personalization is an attractive proposition because of the fact that it cuts down on the network traffic and saves valuable bandwidth. It is predicated on adaptive web sites[18, 19], which change the information returned in response to a request based on the user. Very primitive forms of this can be seen in sites which ask the users to provide some basic declarative information such as address, phone number, and keywords indicating interest, and then tailor their information content (and especially ads) based on zip code, area code, demographic profile, etc.. For e.g. Yahoo creates personalized web pages by asking an individual to sign into his profile for the first time. It asks her to select channels of information she would like to see on her personal page. Yahoo saves the profile under the user's name and uses a cookie to identify the user to display her personal page.

However, in general the appearance of a particular page, including links on it, can also be changed when web sites are adaptive. For example, Perkins *et al.* define operations such as promotions/demotion, highlighting and linking that could be done on static pages to create content tailored for a specific user dynamically. The proposed idea is based on factors such as correlations between two pages. They create a matrix of correlations between pages and use this matrix to decide if the pages are similar. Perhaps the earliest work along similar lines was the Webwatcher project at CMU[4]. It highlights hyperlinks in a page based on the declared interests and the path traversal pattern of a user as well as the path traversal patterns of previous users with similar interests.

Such an approach could generate results which would be totally different than the actual interests of the user. Mining typical user profiles from the vast amount of historical data stored in server or access logs is a possible approach to personalization. It has been proposed in[11], and some initial work has also been done. In [8], associations and sequential patterns between web transactions are discovered based on the Apriori algorithm [2]. 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 plain text transfer of ids, this assumption is clearly not valid. Chen et. al.[7] 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. Also like [8] they assume that user ids are known. Another approach to observing path traversal and clustering based on that data is advanced by Shahabi et al. [22]. The basic approach there is to define a path similarity measure for a given Web site. Then, the logged data about user's paths is clustered to aggregate users into groups using a simple K-means algorithm. However, it is not clear how the similarity metric is devised, and whether it can produce meaningful clusters.

There is also a recent body of work [5, 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. They also seek to discover time dependent patterns in the access logs [24]. Another idea which is gaining momentum is of learning about users and customers by tracking and analyzing their *clickstreams*, which is of great importance in e-commerce, but this has the drawback of breach of privacies issues along with it.

We have used robust fuzzy clustering methods as in [11, 20] to analyze the 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.

In this paper, we present a tool which creates adaptive web sites. The tool uses cookies and traversal patterns obtained from the web logs to generate *personalized* version of a web page for individual users.

2 Creating Personalized Web Pages using Adaptive Web Servers

For creating adaptive web sites using web log mining, one would mine the logs to obtain traversal patterns of the users. This mined information is used the next time the user visits the page to present a personalized version of the web page. This approach would contrast the approach of using information from an "interests" form (as in http://my.netscape.com/ or http://my.cnn.com/). *Interest* form generated sites create a personalized web page based on declarative information. This information is often gathered before-hand from a user (e.g. http://my.cnn.com/) asking him about his interests or by asking him to sign in using a unique user identification. Information derived from a single user profile can be used to recommend him to similar interests items on e-commerce sites (e.g. http://www.amazon.com/). These methods use declarative information or track a single user, to obtain profiles. This information, if disclosed, can be considered as disseminating private information. Hence, users would generally turn profiling off to disallow such intelligence to be gathered to track their behavior.

Our work uses a very simple way to profile users without obtaining any personal or demographic information about them. An apache module places a cookie on the users machine. This cookie is created using the *mod_usertrack* option which generates a pseudo-random unique number for a user and places it on the users machine. The logs generated do not contain user ids since *identd* is not used when a hit is registered. Hence, creating a mapping between user ids and their cookie information is not possible. When a user visits a web page, he is viewed as one belonging to a group of users with similar traversal patterns. The object of this project is to view the user not as an individual, but as a part of a group of users. This method has the advantage of not revealing any private profile information, but at the same time generating a *personalized* page for the user.

The idea behind creating an adaptive web site for the CS Department at UMBC being to model the page as http://my.cs.umbc.edu/ without asking for any user information. For creating an adaptive web server, access logs were collected.

3 Mining Web Logs

In this section, we briefly describe the process of mining web logs. Our prior work [11, 20] contains specific information about the algorithm and the complete process.

3.1 Sessionizing Access Logs

Web server access logs consist of records of all file operations generated by all users accessing the web pages. Each log entry consists of User's IP address, Access date and time, Request method, URL of the page accessed, Protocol used, Return code, Cookie belonging to the user and other fields as described in the Apache Log Format [3]. We filter out error entries, request methods other than "GET" and accesses to image files (.gif, .jpeg, , $\cdots$, etc). Image entries 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. Error entries contain potentially useful information, but they do not serve any purpose with regards to finding traversal patterns.

The individual log entries are grouped into user sessions using a Perl script which is a modification of the *follow* utility [16]. A user session is defined as a sequence of temporally compact accesses by a user. Since web servers do not typically log user names (unless *identd* is used), we define a user session using two different approaches. In our prior work [20] , we defined 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. In this paper, we present an alternate approach to obtain user sessions by using cookies. Cookies are unique to a user and are placed on the users machine, the first time the user accesses the particular web server. On subsequent access the client presents the cookie to the server as a part of the HTTP request. A session is defined as a set of accesses to a web site by a particular user, identified by a cookie, such that time difference between any two consecutive accesses in the session is within a pre-specified time duration. No mapping between a user and a cookie is maintained, hence keeping the users identification secret. 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 $\mathbf{s}^{(i)}$ 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 the scheme using users IP address will map one user's multiple sessions to multiple user sessions and the scheme using Cookies will map one user's multiple sessions to one user session. Usage of cookies reflects a user session profile more clearly than the usage of IP addresses. In the case of different users logging into the same machine and accessing the same pages and in the case of user requests coming in from a proxy server, the cookie approach will clearly differentiate between multiple users while the IP approach will consider them as coming from a single user.

Web caches could cause another problem for our technique (like for all other related systems). With the appropriate use of the No cache pragma in HTTP/1.1, this problem can be been avoided.

3.2 Creating the Dissimilarity Matrix

In the following paragraphs, we introduce the similarity measures between two user-sessions, $\mathbf{s}^{(k)}$ and $\mathbf{s}^{(m)}$. The measures attempt to incorporate both the structure of the site, as well as the URLs involved.If the URLs accessed are completely dissimilar, 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 would give us a count of the common URLs between the two 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 measure is that it totally ignores the similarity between the URLs and hence does not facilitate identifying user travel pattern analysis on a web site. The cosine measure would classify two sessions as similar only if all the pages accessed were exactly same. However, sessions are similar when the pages accessed between them are similar and not necessarily same. Hence, the cosine measure should be used only when the URLs are totally different.

The most obvious approach to compute similarity between URLs is to analyze their content. Analyzing content is an open research problem in Information Retrieval and existing methods are computationally very expensive. Since time is a very important factor in such analysis, this approach is ruled out. Hence we define a similarity measure for URLs on the basis of the structure of the URL. 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*. 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

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

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:

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

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 a value 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 unshared URLs have low similarity. In general for such sessions where the URL similarities are low, the cosine measure $M_{1,kl}$ is higher and more accurate. On the other hand, when the URL similarities are high, $M_{2,kl}$ is higher and more accurate. Therefore, we use [15] 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 $d_s^2(k,l) = d_s^2(l,k), \forall k,l$. 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 {/471/lectures/} and {/471/lectures/lectures.html}. This property is actually desirable for our application, because we consider these two sessions to fit the same profile. The space in which sessions are distributed is not Euclidean which is proved by the fact that the session dissimilarity measure violates the triangular inequality for metric distances in some cases. For instance, the dissimilarity between the sessions {/471/lectures/lecture01} and {/471/lectures} is zero. So is the dissimilarity between {/471/lectures/} and {/471/notes/}. However, the dissimilarity between {/471/lectures/lecture01} and {/471/notes/} is $1/4$ i.e. non-zero. Hence, dissimilarity between sessions becomes more glaring as the accessed URLs move farther from the root causing the amount of specificity in user accesses to increase.

3.3 Clustering the sessions

To cluster the sessions, we use the Fuzzy c Medoids Algorithm (FCMdd) which we have proposed in [20]. We present a brief description of the algorithm below. Let $X = \{\mathbf{x}_i | i = 1, 2, \ldots, n\}$ 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. The Fuzzy Medoids Algorithm (FCMdd) minimizes:

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

where the minimization is performed over all $\mathbf{V}$ in X^c. In (3), u_{ij} represents the fuzzy [6], or possibilistic [14, 10] membership of $\mathbf{x}_j$ in cluster i. The membership u_{ij} is defined by using the

FCM [6] 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 $m \in [1, 2, \infty)$ is the "fuzzifier".

Above equations generate a fuzzy partition of X in the sense that the sum of the memberships of an object $\mathbf{x}_j$ across 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). This is the reason J_m is shown as a function of $\mathbf{V}$ alone. When (3) is minimized, the $\mathbf{V}$ corresponding to the solution generates a fuzzy or possible partition via an equation such as (4). However, (3) cannot be minimized via the alternating optimization technique, because the necessary conditions cannot be derived by differentiating it with respect to the medoids, since solution space is discrete. Hence, an exhaustive search over X^c is required. However, following Fu's [9] heuristic algorithm for a crisp version of (3),we describe the following fuzzy algorithm (FCMdd) that minimizes (3).

The Fuzzy c-Medoids Algorithm (FCMdd)

Fix the number of clusters c; Set $iter$ = 0;
Pick initial medoids $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\}$ from X^c;
Repeat
    Compute memberships u_{ij} for $i = 1, 2, \ldots, c$ and $j = 1, 2, \ldots, n$, by using (4); *(A)*
    Store the current medoids: $\mathbf{V}^{old} = \mathbf{V}$;
    Compute the new medoids $\mathbf{v}_i$ for $i = 1, 2, \ldots, c$:
        $q = \underset{1 \le k \le n}{\text{argmin}} \sum_{j=1}^{n} u_{ij}^m \, r(\mathbf{x}_k, \mathbf{x}_j); \quad \mathbf{v}_i = \mathbf{x}_q;$ *(B)*
    $iter = iter + 1$;
Until $\left(\mathbf{V}^{old} = \mathbf{V}\right.$ or $\left. iter = MAX\_ITER\right)$.

This algorithm falls in the category of Alternating Cluster Estimation [21] paradigm, and is not *guaranteed* to find the global minimum. It is advisable to try many random initializations to increase the reliability of the results. We have experimented with three different ways of initializing the medoids. The first way is to pick all the medoid candidates randomly. We call this method Initialization I. The second way is to pick the first candidate as the object that is most central to the data set, and then pick each successive one in such a way that it is most dissimilar to all the medoids that have already been picked. This makes the initial medoids evenly distributed. We refer to this procedure as Initialization II. For a given data set, the initialization produced by Initialization II is always fixed. Sometimes a bit of randomness might be desirable. In the third initialization strategy, we add randomness by picking the first medoid candidate randomly. The rest of the medoids are selected the same way as in Initialization II. We call this method Initialization III. The computational complexity of Initialization I, II and III are $\mathcal{O}(c)$, $\mathcal{O}(nc^2)$ and $\mathcal{O}(nc^2)$ respectively. We found that both Initialization II and III work well in practice. The fuzzifier m in FCMdd determines the degree of fuzziness of the resulting clusters. Since the medoid always has a membership of 1 in the cluster, raising its membership to the power m has no effect. Thus, when m is high, the mobility of the medoids may be lost, because all memberships become very small except the one corresponding to the current medoid. For this reason, a value between 1 and 1.5 for m is recommended.

It can be seen from step (B) of FCMdd that the complexity of the algorithm is $\mathcal{O}(n^2)$, where n is the number of input objects. However, this is too expensive for most Web mining applications. To overcome this problem, we can modify step (B) of FCMdd so that it examines only a subset of objects while updating the medoid for cluster i. The subset we choose is the set of p objects in X that correspond to the top p highest membership values in cluster i. We denote this subset by $X_{(p)i}$. The subsets $X_{(p)i}$, $i = 1, 2, \dots, c$, can be identified during the membership updating step, i.e., in step (A) of the algorithm. This increases the complexity of step (A) to $\mathcal{O}(ncp)$. However, the complexity of step (B) is reduced to $\mathcal{O}(ncp)$. Therefore, the overall complexity is linear in the number of objects. The value of p should be proportional to the dimensionality of the data, if known.

4 Personalizing Tool

In this section, we present a tool which generates a personalized page for a user coming to a particular web site. The module is written in Mod Perl and it runs on Apache HTTP Server 1.2.5. Apache was selected because of its excellent performance and great flexibility. Given the prototype version of the module, it is currently not used as the default CS page for UMBC, but it exists as another URL which users can access to use their specially generated page. The module exists at the url *http://abhayankar.cs.umbc.edu:8080/mycs* .

Cookies were generated using the *mod_usertrack* module. The server generates a pseudo-random unique number and places it as a cookie for the CS site on the users machine. The cookie duration can be set as required. One can set the cookie to never expire and hence merge new traversal patterns of the user with his previous traversal pattern. Another approach would be to set the cookie duration to some time and when it expires, generate a new cookie for the user and find traversal patterns using this cookie. The choice of the expire date thus involves a tradeoff between showing traversal patterns of a user over his lifetime and retaining clusters with cookies which are no longer valid.

The cookie's expire date in our experiments was set to two months. This was a very large period in comparison to the duration of our logs (i.e. 2 days). Hence we did not face the problem of multiple cookies existing for a user in the same log. This does not pose a problem to our approach as we can set the cookie to never expire.

Every hit on the CS machine registers an entry on the server logs along with the cookie information. The duration of the logs selected for the purpose of evaluation was two days. Logs for five two-day periods were used in our experiments. Each log contains an average of two hundred and fifty thousand entries and more than two thousand user sessions.

Unique session numbers were generated for separate sessions. Session file consisting of the session number and the cookie related to the particular session were generated. Clusters were generated for the web logs using the cookie value as a unique identifier. The cluster numbers generated are just labels and convey no other information. A cluster file consisting of the mapping of the unique session number and the cluster it belonged to was generated.

An apache module generates a web page on the fly, for a user visiting to the web page. The user's cookie is examined to determine whether his traversal pattern is included in a particular session present in the clustered results. If the user cookie is not found, he is shown the default CS page, otherwise further processing is carried out to generate his individual page. From the session file, the various sessions to which the user cookie belongs to , are determined. For each session , corresponding clusters are obtained from the cluster file.

The web page generated has the same header and footer as the default CS page. Each link shown on the generated page contains the page title and a snippet about the page. A snippet is created by stripping of the tags of the content between the *body* tag and presenting the first few lines of information contained in the page. A snippet is supposed to provide some basic information about a page, assuming that the first few lines generally contain the introduction. The personalized web page also consists of a link to the default CS page, in case the user is interested in accessing something other than his normally used pages.

In figure 1, the user falls into the category of people accessing the Oracle help web pages on the CS server and hence shows all links related to those pages accessed on the CS server.

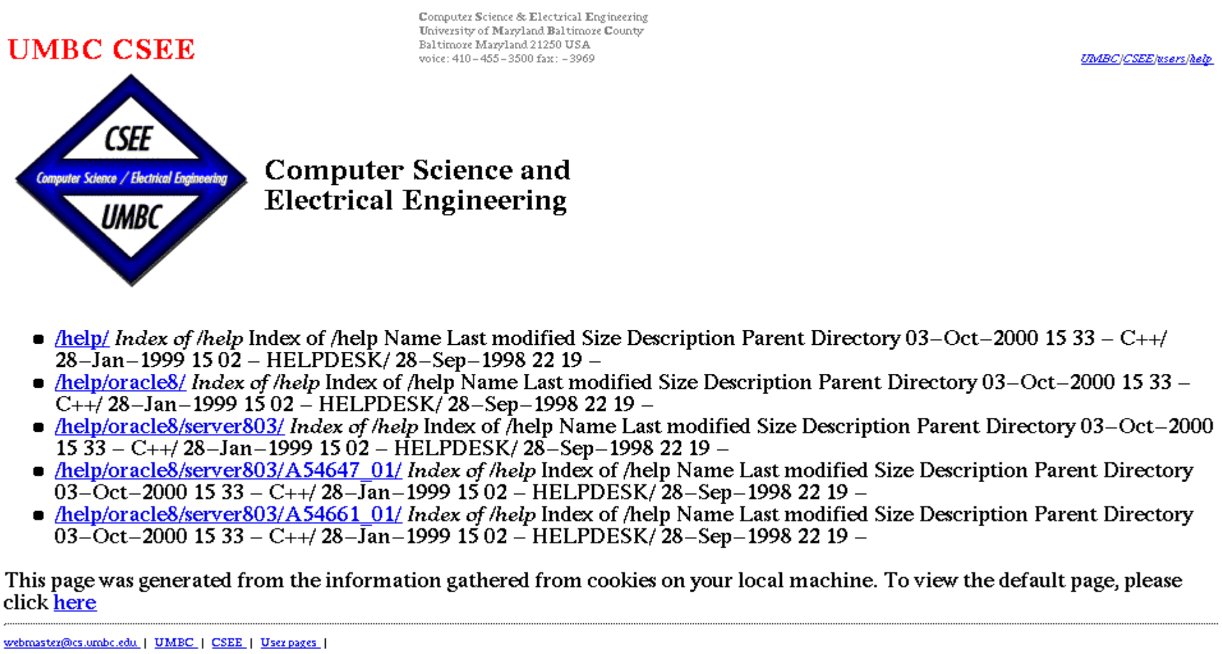

Figure 1: Adaptive Web page for a CS user

Figure 2 shows the default CS page if the user cookie value doesn't exist in the sessions clustered.

5 Experimental Results

Clusters were generated for 5 sets of logs obtained from servers at the Computer Science Department at UMBC. Cookies had been set on the server, so that every log entry also contained the cookie information. All the logs were sessionised using two different parsers, one which took the users IP address as the key and the other which took the cookie as the key. The clustering algorithm was run on both these sets of parsed logs and clusters were obtained.

In order to judge the quality of the clusters, two measures Intra-cluster Distance and inter-cluster Distance were used. Intra-cluster Distance is defined as the average distance of a session in a cluster from all the other sessions in the same cluster. Inter-Cluster Distance is defined as the average distance of all sessions in a cluster from sessions in the remaining clusters. Intra and inter Cluster Distances are normalized to value between 0 and 1. Ideally, one would like the intra-cluster Distance to be minimum i.e. 0 and the inter cluster distance to be maximum i.e. 1.

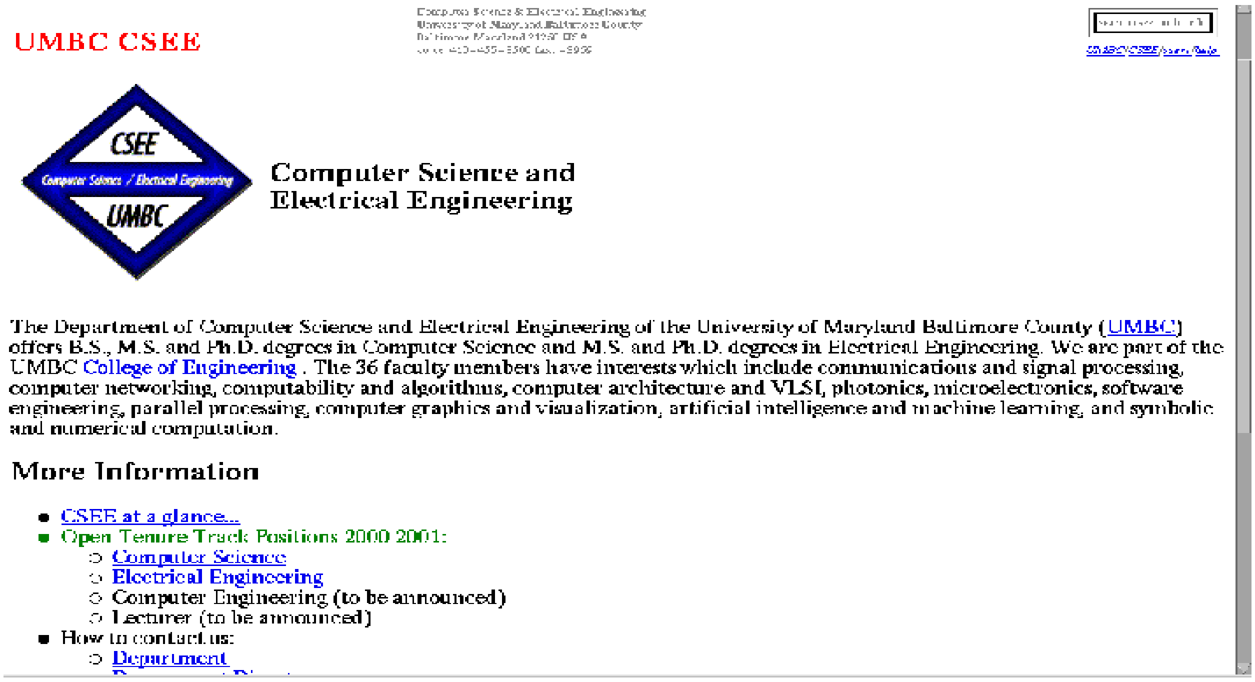

Figure 2: Default CS Web page

Intra and Inter Cluster distances between the clusters obtained from the cookie sessionised logs and IP sessionised logs were computed to judge the quality of the clusters obtained. The Inter-cluster distances were not affected by the selection of cookie over IP addresses as the key. This suggests that the clusters were still as distant from each other as they were distant when IP addresses were used as the key. The Intra-cluster distances were on an average 20% smaller than the intra-cluster distances obtained for the logs with IP addresses as key. Hence, the clusters obtained with the cookies were more compact and dense than their counterparts. Table 1 shows the results hold for all logs sessionised.

The number of sessions in the logs sessionised using cookies is significantly lesser than the number of sessions obtained with the logs sessionised using IP addresses. Table 2 shows the results for all five sets of logs. Since the number of sessions obtained using cookies is less, the time and space required in generating the dissimilarity matrix also reduces. implying that the time $\mathcal{O}(ncp)$ required for clustering the sessions also reduces.

The labels for the clusters are just representative for their content and do not infer any information. Clusters are labeled as numbers to make the analysis task simpler. Hence, it does not infer that Cluster *i* in one data set is similar to Cluster *i* in another data set.

Another interesting observation is that the intra-cluster distance of similar clusters is significantly lesser in the clusters obtained from the cookie parsed logs than the clusters in the IP parsed logs e.g. For the */courses/* cluster 22 in the cookie parsed logs, the intra-cluster distance in 0.511 while for the similar cluster 26 of the IP parsed logs, the intra-cluster distance is 0.553 . Hence, similar clusters are comparatively densely packed in the cookie clusters than in the IP clusters. Also, the clusters created using cookies have less outliers than the clusters created using IP addresses. Hence, clusters created using cookies are more cohesive than the clusters created using IP addresses.

Table 3 shows that nearly half the number of clusters obtained in both approaches are common to each other. From Table 1, we can observe that among similar clusters, the intra-cluster distance

Logs	Without Cookie		With Cookie	
Distance	**Intra-Cluster**	**Inter-Cluster**	**Intra-Cluster**	**Inter-Cluster**
Oct 1-2	0.480	0.993	0.399	0.996
Oct 3-4	0.418	0.995	0.340	0.995
Oct 5-6	0.441	0.994	0.315	0.997
Oct 7-8	0.441	0.995	0.322	0.997
Oct 9-10	0.403	0.995	0.335	0.997

Table 1: Intra and Inter Cluster Distance Measurements

Sessions	Without Cookie	With Cookie
Oct 1-2	5960	5212
Oct 3-4	6400	5569
Oct 5-6	4937	4254
Oct 7-8	4989	4195
Oct 9-10	7929	7061

Table 2: Sessions obtained without and with use of cookies

Clusters	Without Cookie	With Cookie	Similar Clusters
Oct 1-2	21	17	9
Oct 3-4	20	24	9
Oct 5-6	24	19	14
Oct 7-8	19	17	13
Oct 9-10	24	20	11

Table 3: Meaningful Cluseters obtained without and with use of cookies

between the clusters obtained using cookies is lesser than the intra-cluster distance obtained by using IP addresses. This contributes to the overall decrease in average intra-cluster distance. Also, clusters with higher intra-cluster distance in the IP sessionised logs tend to fall off in the clustering process when logs are sessionised using cookies. Hence, this decreases the intra-cluster distance for clusters obtained by using cookies even further resulting in about 20% improvement in intra-cluster distance and causing clusters obtained to be more compact.

For the Clusters generated using the Cookies for sessionizing

- Cluster 1 (/˜sletsc1/) and Cluster 5(/˜mikeg/) correspond to user pages accessed frequently by other users.
- Cluster 2 (/˜sli2) corresponds to users interested in Java based computer games in a user's page.
- Cluster 4 (/˜tbogar1/), Cluster 6 (/˜ebedwe/), Cluster 8 (/˜ebert/), Cluster 10 (/˜squire/), Cluster 12 (/˜sherman/), Cluster 15 (/˜pusquel/) , Cluster 17 (/˜lomanaco/) and Cluster 20 (/˜evans/) represent user sessions that accessed faculty pages.

- Cluster 7 (/˜ugrad/) represents user sessions that accessed the undergraduate students pages.
- Cluster 9 (/www/) represents user sessions that accessed an older version of the CSEE Web pages.
- Cluster 11 (/help/oracle/) represents user sessions that accessed the help pages of oracle8. These are likely to be from students enrolled in the undergraduate and graduate database courses, both of which use oracle.
- Cluster 13 (/461/), Cluster 19 (/471/) and Cluster 22 (/courses/) represent user sessions that accessed the pages belonging to a particular class offered at UMBC.
- Cluster 14 (/kqml/) contain user sessions that accessed the pages maintained by the Agents group at UMBC about KQML.
- Cluster 16 (/cikm/) represents the user sessions accessing the CIKM conference pages.
- Clusters 0, 3 and 24 have cardinalities that are too small to be included in the study.

For the Clusters generated without using the Cookies for sessionizing

- Clusters 1, 2, 4, 5, 8, 9, 11, 12, 14, 15, 18, 20, 23, 25 and 26 are similar to cluster obtained with the use of cookies with correspondence between them given by Table 5.
- Cluster 10 (/˜thurston/), Cluster 13 (/˜stephens/), luster 16 (/˜kalpakis/) and Cluster 19 (/˜chang/) represent user sessions that accessed faculty pages.
- Cluster 17 (/331/) represent user sessions that accessed the pages belonging to a particular class offered at UMBC.
- Cluster 21 (/agentslist/) and Cluster 22 (/robots.txt/) contain user sessions that accessed the pages maintained by the Agents group at UMBC.
- Clusters 0, 3 and 21 have cardinalities that are too small to be included in the study.

6 Conclusions

In this paper, we present an approach to personalize the web space by generating pages dynamically based on off-line clustering of web logs. Web logs were sessionised using cookies and this approach was found to generate clusters with lesser intra-cluster distance than the clusters generated by clustering using IP addresses. These adaptive web pages can reduce network traffic by giving a user his required pages without him looking around for it. Such an approach also lays low the issue of security since no personal or declarative information is obtained from a user to present "personalized" pages. In ongoing work, we are examining the scalability of the system and ways to mine logs in an incremental manner.

References

[1] Firefly. *http://www.firefly.com.*

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference*, pages 487–499, Santiago, Chile, 1994.

[3] Apache.org. Mod_log_common module for logfile structure. http://www.apache.org/docs-1.2/mod/mod_log_common.html.

[4] R. Armstrong, T. Joachims D. Freitag, and T. Mitchell. Webwatcher: A learning apprentice for the World Wide Web. In *Proceedings of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, pages 6–13, Stanford, CA, March 1995.

[5] G. Arocena and A. Mendelz. Weboql: Restructuring documents, databases, and web. In *Proc. IEEE Intl. Conf. Data Engineering '98*. IEEE Press, 1998.

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

[7] M.S. Chen, J.-S. Park, and P. S. Yu. Efficient data mining for path traversal patterns. *IEEE Trans. Knowledge and Data Engineering*, 10(2):209–221, April 1998.

[8] R. Cooley, B. Mobasher, and J. Srivastav. Web Mining: Information and pattern discovery on the World Wide Web. In *Proc. IEEE Intl. Conf. Tools with AI*, pages 558–567, Newport Beach, CA, 1997.

[9] K. S. Fu. *Syntactic Pattern Recognition and Applications*. Academic Press, San Diego, CA, 1982.

[10] R. J. Hathaway and J. C. Bezdek. Switching regression models and fuzzy clustering. *IEEE Transactions on Fuzzy Systems*, 1(3):195–204, 1993.

[11] A. Joshi and R. Krishnapuram. Robust fuzzy clustering methods to support web mining. In *Workshop in Data Mining and knowledge Discovery SIGMOD*, volume 15, pages 1–8, 1998.

[12] A. Joshi, C. Punyapu, and P. Karnam. Personalization and asynchronicity to support mobile web access. In *Proc. Workshop on Web Information and Data Management, 7^{th} Intl. Conf. on Information and Knowledge Management*, November 1998.

[13] A. Joshi, S. Weerawarana, and E. Houstis. On disconnected browsing of distributed information. In *Proceedings of IEEE Intl. Workshop on Research Issues in Data Engineering (RIDE)*, pages 101–108, Birmingham, UK, 1997.

[14] R. Krishnapuram and J. M. Keller. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98–110, 1993.

[15] O. Nasraoui, H. Frigui, A. Joshi, and R. Krishnapuram. Mining web access logs using relational competitive fuzzy clustering. *Eight International Fuzzy Systems Association World Congress, Taipei*, August 1999.

[16] Mark Nottingham. Follow: A session based log analyzing tool. http://www.pobox.com/~mnot/follow/.

[17] O.Zaiane and J. Han. Webml: Querying the world-wide web for resources and knowledge. In *Proc. Workshop on Web Information and Data Management,* 7^{th} *Intl. Conf. on Information and Knowledge Management*, 1998.

[18] M. Perkowitz and O. Etzioni. Adaptive web sites: an ai challenge. In *Proc. Intl. Joint Conf. on AI – IJCAI97*, 1997.

[19] M. Perkowitz and O. Etzioni. Adaptive web sites: Automatically synthesizing web pages. In *Proc. AAAI 98*, 1998.

[20] Krishnapuram R. and Joshi A. On mining web acceess logs. In *Workshop on Research Issues in Data Mining and Knowledge Discovery SIGMOD*, pages 63–69, 2000.

[21] T. A. Runkler and J. C. Bezdek. ACE: A tool for clustering and rule extraction. *IEEE Transactions on Fuzzy Systems*, 1999.

[22] Cyrus Shahabi, Adil Faisal, Farnoush Banaei Kashani, and Jabed Faruque. Insite: A tool for interpreting users interaction with a web space. In *VLDB*, 2000.

[23] L. Terveen, W. Hill, and B. Amento. PHOAKS - a system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, 1997.

[24] O.R. Zaiane, M. 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 Conf. (ADL'98)*, pages 19–29, 1998.

Table 4: CSEE Logs Analysis using Linear FCMdd Algorithm and Cookies

Cluster	**Cardinality**	**URLs**	**URLs**	**Deg**
0 -	832	2106		≤ 0.5
1 - /~sletsc1/	19	56	{/~sletsc1/*}	5.211
			{/~sletsc1/DWC/*}	5.000
2 -/~sli2/	211	108	{/~sli2/*}	17.336
			{/~sli2/cube/*}	14.303
			{/~sli2/plot*}	1.265
4 - /~tbogar1/	32	28	{/~tbogar1/*}	3.313
			{/~tbogar1/dlance.html/*}	0.8125
5 - /~mikeg/	75	24	{/~mikeg/*}	6.093
			{/~mikeg/medmcat.html/*}	0.5867
6 -/~ebedwe/	9	19	{/~ebedwe/*}	3.444
			{/~ebedwe/evolution/*}	0.888
			{/~ebedwe/skin/*}	1.666
7 -/~ugrad/	79	195	{/~ugrad/*}	4.557
			{/courses/*}	2.481
			{/courses/undergraduate/*}	1.962
8 - /~ebert/	93	187	{/~ebert/*}	3.71
			{/~ebert/435/*}	1.688
9 -/www/	95	252	{/www/*}	1.453
			{/www/graduate/*}	0.947
10 - /~squire/	95	216	{/~squire/*}	3.842
			{/~squire/download/*}	1.000
11 - /help/oracle8/	411	995	{/help/*}	5.183
			{/help/oracle8/*}	4.526
			{/help/oracle8/server803*}	3.284
			{/help/oracle8/server803/A54657_01/*}	0.541
12 - /~sherman/	61	202	{/~sherman/*}	4.443
			{/~sherman/Chess/*}	1.295
			{/~sherman/Courses/*}	2.639
13 -/461/	14	278	{/461/*}	24.929
			{/461/spring98/*}	24.143
			{/461/spring98/lectures/*}	24.143
			{/461/spring98/lectures/lecture1/*}	11.714
14 - /kqml/	69	238	{/kqml/*}	3.913
			{/abir/*}	1.449
			{/abir/ad198/*}	1.449
			{/kqml/papers/*}	1.118
15 -/~plusquel/	106	197	{/~plusquel/*}	3.217
			{/~plusquel/611/*}	0.915
			{/~plusquel/vlsi/*}	1.236
16 - /cikm/	58	51	{/cikm/*}	2.534
			{/cikm/2000/*}	1.034
17 -/~lomanaco/	107	119	{/~lomanaco/*}	2.776
			{/~lomanaco/f00/*}	0.944
18	*1*	*2*		
19 -/471/	42	458	{/471/*}	15.143
			{/471/current/*}	3.595
			{/471/notes/*}	7.095

Table 5: CSEE Logs Analysis using Linear FCMdd Algorithm and Cookies

Cluster	Cardinality	URLs	URLs	Deg
20 -/~evans/	30	60	{/~evans/*}	3.400
21	*4*	*4*		
22 -/courses/	1810	3157	{/courses/*}	6.945
			{/courses/undergraduate/*}	6.267
			{/courses/undergraduate/201/*}	3.787
			{/courses/undergraduate/201/fall00/*}	3.432
			{/courses/undergraduate/201/fall00/lectures/*}	2.261

Table 6: CSEE Logs Analysis using Linear FCMdd Algorithm and without Cookies

Cluster	Cardinality	URLs	URLs	Deg
0 -	888			≤ 0.5
1 -/~evans/	33	176	{/~evans/*}	3.545
2 - /~sherman/	172	433	{/~sherman/*}	1.203
			{/~squire/*}	1.686
3	*5*	*47*	/~linglan/*	2
4 - /cikm/	79	73	{/cikm/*}	2.746
			{/cikm/2000/*}	1.342
5 - /~sletsc1/	22	63	{/~sletsc1/*}	4.227
			{/~sletsc1/DWC/*}	4.045
6 - /2000/	29	22	{/2000/*}	2.379
7 - /%7Esli2/	12	21	{/%7Esli2/*}	3.666
			{/%7Esli2/cube/*}	3.666
8 - /~mikeg/	100	122	{/~mikeg/*}	4/67
			{/~mikeg/medschool.html/*}	0.62
9 -/471/	67	645	{/471/*}	11.119
			{/471/current/*}	2.523
			{/471/notes/*}	5.627
10 - /~thurston/	47	257	{/~thurston/*}	2.446
			{/~sherman/*}	1.255
11 - /~tbogar1/	36	45	{/~tbogar1/*}	3.583
			{/~tbogar1/dlance.html/*}	0.75
12 - /help/oracle8/	441	2325	{/help/*}	8.857
			{/help/oracle8/*}	8.17
			{/help/oracle8/server803*}	5.487
13 - /~stephens/	48	2000	{/~stephens/*}	4.625
			{/~stephens/NUM/*}	0.917
14 - /~ebert/	119	331	{/~ebert/*}	3.605
			{/~ebert/435/*}	1.294
15 -/~ugrad/	81	377	{/~ugrad/*}	4.704
			{/courses/*}	3.049
			{/courses/undergraduate/*}	2.481

Table 7: CSEE Logs Analysis using Linear FCMdd Algorithm and without Cookies

Cluster	**Cardinality**	**URLs**	**URLs**	**Deg**
16 - /~kalpakis/	27	114	{/~kalpakis/*}	3.629
			{/~kalpakis/Courses/*}	2.555
			{/~kalpakis/Courses/441/*}	2.148
17 -/331/	8	19	{/331/*}	3.375
			{/331/current/*}	1.125
			{/331/fall00/*}	1.25
18 -/~lomanaco/	74	204	{/~lomanaco/*}	4.689
			{/~lomanaco/f00/*}	1.311
19 - /~chang/	26	151	{/~chang/*}	2.962
			{/~chang/cs441.f99/*}	1.038
20 - /kqml/	156	1057	{/kqml/*}	3.673
			{/kqml/mail/*}	1.282
			{/kqml/mail/kqml/*}	1.179
21 - /agentslist/	52	158	{/agentslist/*}	1.327
			{/~jklabraou/*}	1.192
22 - /robots.txt/	147	458	{/robots.txt/*}	1.000
23 -/461/	216	4357	{/461/*}	2.102
			{/461/spring98/*}	1.431
			{/agentslist/*}	2.873
24	*4*	*11*		
25 -/~sli2/	261	163	{/~sli2/*}	14.199
			{/~sli2/cube/*}	11.762
26 -/courses/	1787	15542	{/courses/*}	10.077
			{/courses/undergraduate/*}	9.523
			{/courses/undergraduate/201/*}	4.222
			{/courses/undergraduate/201/fall00/*}	3.422
			{/courses/undergraduate/201/fall00/lectures/*}	2.222

Cluster	With Cookie		Without Cookie	
	Cluster Number	Intra-Cluster Distance	Cluster Number	Intra-Cluster Distance
/~sletsc1/	1	0.044	5	0.191
/~sli2/	2	0.088	26	0.553
/~tbogar1/	4	0.108	11	0.153
/~mikeg/	5	0.010	8	0.194
/~ugrad/	7	0.276	15	0.355
/~ebert/	8	0.280	14	0.366
/~squire/	10	0.316	2	0.801
/help/oracle8/	11	0.264	12	0.314
/~sherman/	12	0.538	2	0.801
/461/	13	0.261	23	0.919
/kqml/	14	0.044	10	0.647
/cikm/	16	0.223	4	0.272
/~lomanaco/	17	0.370	18	0.443
/courses/	22	0.511	26	0.553

Table 8: Common Clusters between Logs sessionised with and without Cookies

Cluster	Intra-Cluster	Inter-Cluster
0	0.986	0.999
1	0.044	0.999
2	0.088	0.999
4	0.108	0.988
5	0.010	0.999
6	0.229	0.997
7	0.276	0.997
8	0.280	0.996
9	0.716	0.991
10	0.316	0.997
11	0.264	0.997
12	0.538	0.996
13	0.261	0.999
14	0.377	0.999
15	0.381	0.999
16	0.223	0.999
17	0.370	0.997
19	0.383	0.999
20	0.052	0.999
21	0.206	0.997
22	0.511	0.987
Average	0.315	0.997

Table 9: Intra and Inter Cluster Distance with Cookies

Cluster	Intra-Cluster	Inter-Cluster
0	0.974	0.997
1	0.051	0.999
2	0.801	0.995
3	0.895	0.987
4	0.272	0.990
5	0.191	0.998
6	0.131	0.991
7	0.095	0.998
8	0.194	0.996
9	0.592	0.997
10	0.127	0.998
11	0.153	0.999
12	0.314	0.998
13	0.400	0.996
14	0.366	0.997
15	0.355	0.992
16	0.565	0.997
17	0.344	0.998
18	0.443	0.995
19	0.677	0.995
20	0.647	0.993
21	0.775	0.995
22	0.460	0.998
23	0.919	0.983
24	0.513	0.998
25	0.115	0.998
26	0.553	0.976
Average	0.441	0.994

Table 10: Intra and Inter Cluster Distance without Cookies