Web User Clustering and Its Application to Prefetching Using ART Neural Networks

Santosh K. Rangarajan[$], VirV. Phoha[$], Kiran Balagani[$], S. S. Iyengar[*], Rastko Selmic[£]

[$]Department of Computer Science
[£]Department of Electrical Engineering
Louisiana Tech University
Email: phoha@latech.edu

[*]Department of Computer Science
Louisiana State University
Email: iyengar@bit.csc.lsu.edu

Abstract

In this paper, we present a novel approach to group users according to their Web access patterns. Our technique for grouping users is based on the ART1 neural network. We compare the quality of clustering of our ART1 based clustering technique with that of the K-Means clustering algorithm in terms of inter-cluster and intra-cluster distances. Our results show that the average inter-cluster distance of the clusters formed by K-Means algorithm varies from 12.66 to 24.20, while the average inter-cluster distance of clusters formed by our ART1 based clustering technique is almost constant (approximately 18.01), which indicates the high quality of clusters formed by our approach. We present a prefetching scheme in which we apply our clustering technique to group users and then prefetch their requests according to the prototype vector of each group. Our prefetching scheme has prediction accuracy as high as 97.78%.

Keywords

Web usage mining, clustering, Adaptive Resonance Theory (ART), ART1 neural network, K-Means algorithm, prefetching

1. Introduction

An important attribute contributing to the popularity of a Web site is the degree of personalization it offers when presenting its services to users. However, improving the level of user personalization by reorganizing the entire Web site structure according to the interests of each user increases the number of computations at the Web server hosting the Web site. One solution to avoid this problem is to group users based on their Web interests, and then organize the structure of the Web site in a manner suitable to the Web needs of different groups. It is difficult to group users according to their Web interests mainly because of two reasons: (1) users' interests are diverse and, (2) users' interests change with time. Web access logs serve as a substantial source of information about users' Web access patterns. Properly exploited, the Web access logs can be used to analyze and discover useful information about users' interests with the site.

In this paper, we present an ART1 based clustering algorithm to group users according to their Web access patterns [1]. The ART1 [2] is a modified version of ART

[3] for clustering binary vectors. The advantage of using the ART1 algorithm to group users is that it adapts to the change in users' Web access patterns over time without losing information about their previous Web access patterns. In our ART1 based clustering approach, each cluster of users is represented by a prototype vector that is a generalized representation of URLs frequently accessed by all the members of that cluster. One can control the degree of similarity between the members of each cluster by changing the value of the vigilance parameter. In our work, we analyze the clusters formed by using the ART1 technique by varying the vigilance parameter ρ between the values 0.1 and 0.5. We compare the performance of ART1 clustering technique with that of the traditional K-Means clustering algorithm in terms of average inter-cluster and average intra-cluster distances. Our results show that, while there is not much difference between the inter-cluster distances obtained by using both the algorithms, there is a considerable difference in the intra-cluster distances. For the clusters formed using the K-Means algorithm, the intra-cluster distance varied from 12.66 to 24.20. For the clusters formed by our approach, the intra-cluster distance remained constant (approximately 18.01), which indicates that the variance within each cluster is uniform. We present a prefetching scheme in which we use our ART1 based clustering algorithm to cluster users based on their access patterns. Our prefetching scheme predicts future requests according to the prototype vector of each cluster. The overall architecture of our clustering and prefetching technique is illustrated in Figure 1.

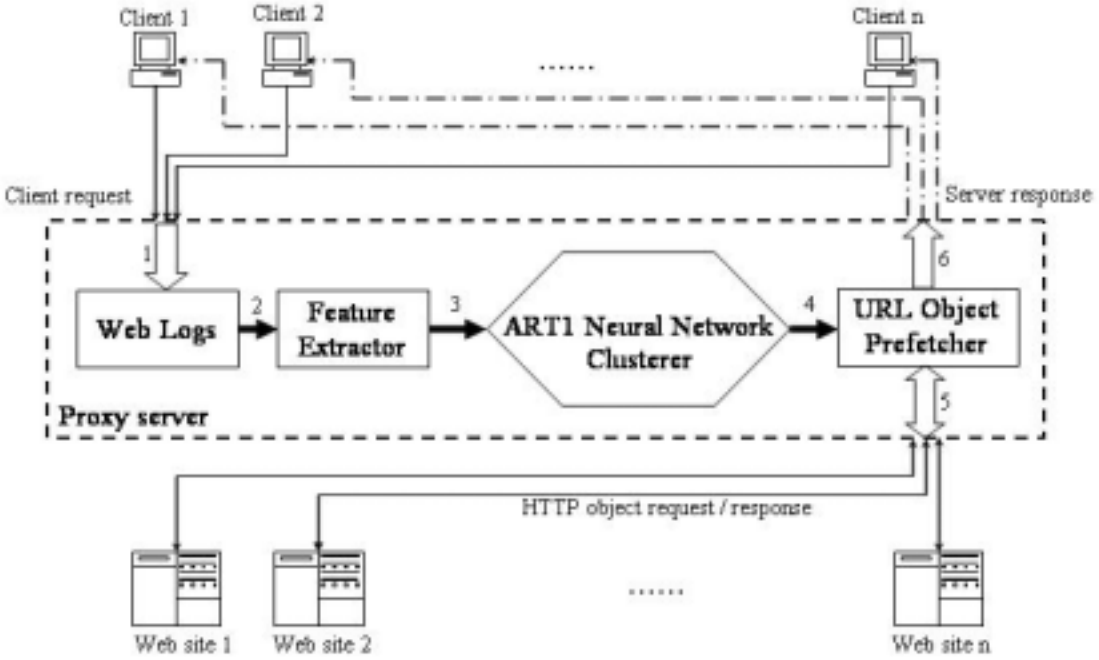

Figure 1. Architecture of our clustering and prefetching scheme. (1) Each client's request is recorded in the proxy server's Web log file, (2) the feature extractor extracts the features as discussed in Section 3.2, (3) the feature vector of each client is the input to our offline ART1 based clusterer, (4) the ART1 based clusterer identifies the group to which the client belongs and returns the prototype vector of that group, (5) the prefetcher requests all URL objects represented by the prototype vector, and (6) the proxy server responds to the clients with prefetched URL objects.

The rest of the paper is organized as follows. In Section 2, we briefly review the research done in the areas of Web user clustering and prefetching. In Section 3, we explain our implementation of the ART1 based clustering method and discuss its application for prefetching users' requests. In Section 4, we present the results of our work. We conclude our work in Section 5.

2. Related Work

In this section, we discuss significant work in the areas of Web user clustering and prefetching.

2.1. Related research in clustering Web users

Clustering users based on their Web access patterns is an active area of research in Web usage mining. R. Cooley et al. [4] propose a taxonomy of Web Mining and present various research issues, techniques and future directions in this field. Phoha et al. use competitive neural networks and data mining techniques to develop schemes for fast allocation of Web pages [5]. M. N. Garofalakis et al. [6] review popular data mining techniques and algorithms for discovering Web, hypertext, and hyperlink structure. Y. Fu et al. [7] present a generalization based clustering approach, which combines attribute oriented induction, and BIRCH [8] to generate hierarchical clustering of Web users based on their access patterns. I. Cadez et al. [9] use first-order Markov models to cluster users according to the order in which they request Web pages. The Expectation Maximization algorithm is then used to learn the mixture of first-order Markov models that represent each cluster. G. Paliouras et al. [10] analyze the performance of three clustering algorithms (1) Autoclass, (2) Self Organizing Maps and (3) Cluster Mining for constructing community models for the users of large Websites. Xie and Phoha [11] apply the concept of mass distribution in Dempster-Shafer's theory and propose a belief function similarity measure, which adds to the clustering algorithm the ability to handle uncertainty among Web users' navigation behavior.

Although the algorithms and techniques discussed in this section succeed in grouping the users' according to their diverse interests, they lack the ability to adapt to the change in users' Web interests over time.

2.2. Related research in prefetching

Prefetching means fetching the URL objects before the users request them. For a prefetching scheme to be effective there should be an efficient method to predict users' requests. An efficient prefetching scheme effectively reduces the user perceived Web latency. However, an inefficient prefetching technique causes wastage of network resources by increasing the Web traffic over the network, which in turn increases Web latency.

Loon and Bhargavan [12] present an approach for prefetching URLs based on users' profiles. Each user's profile is represented by a weighted directed graph in which the nodes represent URLs and the edges represent the access paths. The weight of a node represents the frequency of access of URLs and the weight of an edge represents the frequency of access of one URL after another. This weighted directed graph is used to predict the user's request. Ibrahim and Xu [13] present a context specific prefetching technique, which uses an artificial neural network for predicting users' requests. Li Fan et

al. [14] investigate an approach to reduce Web latency, by prefetching between caching proxies and browsers. Their technique uses the Prediction by Partial Matching (PPM) algorithm for prefetching. The prediction accuracy of PPM ranges from 40% to 73%, and generates an extra traffic ranging from 1-15%. Evangelos and Chronaki [15] present a simple and effective Top-10 approach for prefetching. In their approach, the ten most popular Web pages are prefetched. The authors show that the Top-10 approach accurately predicts 60% of the future requests. Padmanabhan and Mogul [16] present a prefetching scheme in which the server computes the likelihood that a particular Web page will be accessed and conveys this information to the client. The client program then decides whether to prefetch the Web page. The prediction is based on a dependency graph similar to the one used in [12]. The authors conclude that their methodology results in substantial reduction in Web latency, but increases the traffic on the network. Tian, Choi, and Phoha [17] present an intelligent and adaptive neural network predictor, which uses the back propagation learning rule to learn the changing access patterns of pages in a Web site.

Most of the research discussed in prefetching concentrates on prefetching individual users' requests according to their previous access patterns. Although these methods are efficient for prefetching, they may considerably overload the network with unnecessary traffic when prefetching for a large number of users. To reduce such an effect of prefetching, we present a prefetching scheme that uses ART1 clustering technique to prefetch requests for a large community of users instead of prefetching individual users' requests.

3. Methodology

This section describes preprocessing of Web logs, extraction of feature vectors, and clustering users using adaptive resonance theory, and our prefetching scheme.

3.1. Preprocessing the Web logs

For testing our approach, we use the Web log files provided by NASA, spanning the timeframe from July 1, 1995 through July 15, 1995. The Web logs contain HTTP requests to NASA Kennedy space center's WWW server [18]. The raw data from the log file is in the following form:

< host name, timestamp, requested URL, HTTP reply code, bytes sent in reply >

The "host name" field of each Web log contains the identity of the host making a request to the NASA WWW server. Each host represents a large community of organizationally related users. We preprocess the log files by filtering the log files to contain access patterns for 70 hosts whose requests constitute the majority of the Web logs. We then clean the Web logs to contain the requests for URLs that were frequently requested by the 70 selected hosts (the remaining hosts were removed because the number of requests generated by each of them was insufficient to be considered for grouping). There are 200 such URLs with their frequency ranging from 32 to 3435. These URLs contributed to 114,290 hits.

3.2. Extraction of feature vectors

The base vector $B = \{URL_1, URL_2, \ldots URL_{200}\}$ represents the access pattern of the hosts. For each host *H*, we form a binary pattern vector P_H, which is an instance of the base vector. The pattern vector P_H is formed by mapping the frequency of access of each element "URL_i" in B to binary values. The pattern vector P_H is of the form $P_H = \{P_1, P_2 \ldots P_{200}\}$ where P_H is the pattern vector of host *H*, $1 \le H \le 70$ and P_i is an element of P_H having a value of either zero or one. The pattern vector P_H is the input vector to the ART1 clustering algorithm. A description of the procedure to form the pattern vector follows.

For each pattern vector, P_H *such that* $H = 1$ *to* 70 *//There are* 70 *input pattern vectors* P_H
//where H stands for Host-Id

For each element P_i *in pattern vector* P_H, $i = 1$ *to* 200 *// Size of the pattern vector* P_H *is* 200
//where i stands for URL-Id

$$P_i = \begin{cases} 1 & \text{if } URL_i \text{ is requested by the host 2 or more times} \\ 0 & \text{if } URL_i \text{ is requested by the host less than 2 times} \end{cases}$$

End
End

3.3. Clustering users using Adaptive Resonance Theory

For clustering the hosts (each host represents a large community of organizationally related users, e.g., all requests with the host-name www.latech.edu represent the requests made by the students and faculty members of Louisiana Tech University), we adapt ART1 algorithm to our situation, which is more suitable for clustering binary vectors. ART1 consists of an Attentional subsystem and an Orientation subsystem. The Attentional subsystem consists of Comparison layer F_1, Recognition layer F_2, and Control gains G_1 and G_2. F_1 and F_2 layers are fully connected with top-down weights and bottom-up weights. The Orientation subsystem consists of the vigilance parameter ρ. The input pattern vectors $P_{H=1 \text{ to } 70}$ are presented at the F_1 layer. Each input pattern presented at the F_1 layer activates a node (winner) in the F_2 layer. The F_2 layer reads out the top-down expectation to F_1, where the winner is compared with the input vector. The vigilance parameter determines the mismatch that is to be tolerated when assigning each host to a cluster. If the match between the winner and the input vector is within the tolerance, the top-down weights corresponding to the winner are modified. If a mismatch occurs, F_1 layer sends a reset burst to F_2, which shuts off the current node, and chooses another uncommitted node. Once the network stabilizes, the top-down weights corresponding to each node in the F_2 layer represent the prototype vector for that node. Our architecture of ART1 based network for clustering user communities (illustrated in Figure 2) consists of 200 nodes in the F_1 layer, with each node presented with the binary value 0 or 1. The pattern vector P_H, which represents the access pattern of each host *H* is presented at the F_1

layer. The F_2 layer consists of a variable number of nodes corresponding to the number of clusters.

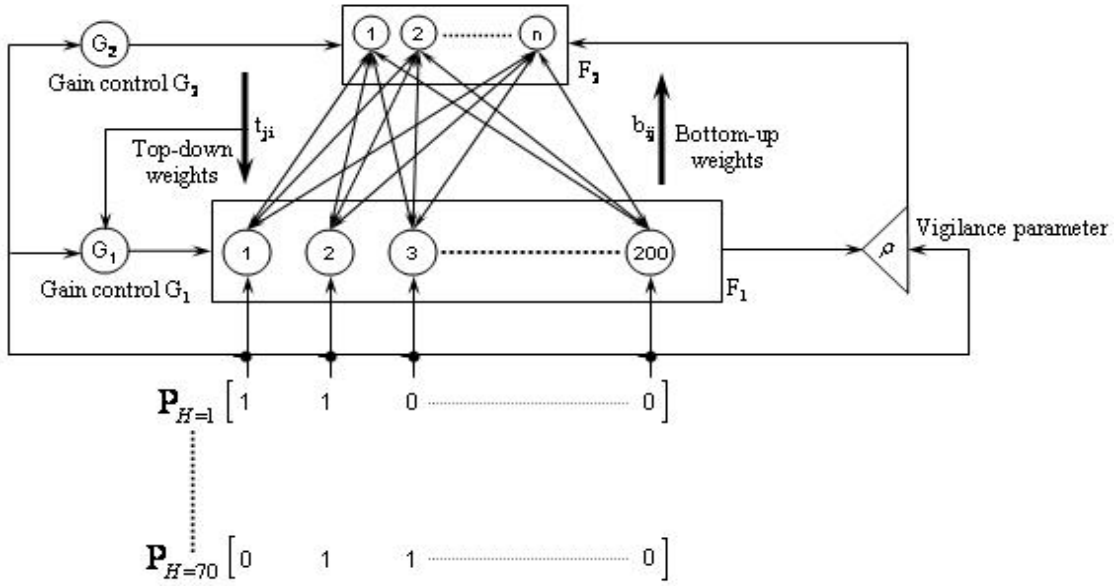

Figure 2. Architecture of our ART1 neural network based clusterer. The pattern vector P_H, which represents the access patterns of the host *H* is the input to the Comparison layer F_1. The vigilance parameter determines the degree of mismatch that is to be tolerated. The nodes at the Recognition layer F_2 represent the clusters formed. Once the network stabilizes, the top-down weights corresponding to each node in F_2 represent the prototype vector for that node.

The procedure for clustering hosts using the ART1 algorithm follows.

Procedure: *ART1_Clustering (An array of input vectors P [], vigilance parameter value)*
Input:

i. *Feature vectors $P_{H=1 \text{ to } 70}$, each representing the Web access patterns of the hosts.*
ii. *The vigilance parameter value (ρ). We tested the **ART1_Clustering** algorithm by varying the ρ between the values 0.3 and 0.5.*

Output: *Clusters of hosts grouped according to the similarity determined by ρ .*

Assign values to control gains G_1 and G_2

$$G_1 = \begin{cases} 1 \, if \;\; input \;\; P_H \neq 0 \;\; and \;\; output \;\; from \;\; F_2 \;\; Layer = 0 \\ 0 \;\; Otherwise \end{cases}$$

$$G_2 = \begin{cases} 1 \, if \;\; input \;\; P_H \neq 0 \;\; and \;\; output \;\; from \;\; F_2 \; Layer = 0 \\ 0 \;\; Otherwise \end{cases}$$

Step 1: Initialization step

i. *Set nodes in F_1 layer and F_2 layer to zero*

ii. *Initialize top-down (t_{ji}) and bottom-up (b_{ij}) weights*

$t_{ji} = 1$ *and* $b_{ij} = \frac{1}{n+1}$ *, where n is the size of the input vector; (n=200)*

iii. *Initialize the vigilance parameter (ρ),* $0.3 \le \rho \le 0.5$

Step 2: Repeat steps 3-10 until all input vectors $P_{H=1 \text{ to } 70}$ are presented to the F_1 layer.

Step 3: Present randomly chosen input vector $P_H = (P_1, P_2, \cdots, P_{i=200})$ *where* $P_i = 0$ *or* 1 *at F_1.*

Step 4: Compute input 'y_j' for each node in F_2 layer using:

$$y_j = \sum_{i=1}^{200} P_i \times b_{ij}$$

Step 5: Determine k, the node in F_2 that has the largest y_k

$$y_k = \sum_{j=1}^{\text{number of nodes in } F_2} \max(y_j)$$

Step 6: Compute activation $X_k^* = (X_1^*, X_2^*, \cdots, X_{i=200}^*)$ *for the node k in F_1*

where $X_i^* = t_{ki} \times P_i$ *where* $i = 1 \ldots 200$

Step 7: Calculate the similarity between X_k^* *and input* P_H *using:*

$$\frac{\|X_k^*\|}{\|P_H\|} = \frac{\sum_{i=1}^{200} X_i^*}{\sum_{i=1}^{200} P_i}$$

Step 8: Compare the similarity calculated in Step 7 with the vigilance parameter:

if $\left(\frac{\|X_k^*\|}{\|P_H\|} > \rho \right)$

begin

Associate input P_H with node k

i. *Temporarily disable node k by setting its activation to 0*

ii. *Update top-down weights of node k*
$t_{ki}(new) = t_{ki} \times P_i$ *where* $i = 1 \ldots 200$

end

else

Step 9: Create a new node in F_2 layer

begin

i. *Create a new node l*

ii. Initialize the top-down weights 't_{li}' to the current input pattern

iii. Initialize bottom-up weights for the new node l

$$b_{il}(new) = \frac{X_i^*}{0.5 + \sum_{i=1}^{200} X_i^*} \quad where\ i = 1 \ldots 200$$

end

Step 10: Goto Step 2.
Step 11: End ***ART1-Clustering ( )*** *algorithm.*

3.4. Prefetching Scheme

Most of the techniques for prefetching discussed in Section 2.2 predict requests for a single user. Such approaches may overload the network with unnecessary Web traffic when prefetching requests for a large number of users. We use our ART1 neural network based clustering technique to prefetch requests for a community of users thereby reducing the overload on the network occurring due to prefetching individual users' requests.

Our prefetching scheme clusters the hosts using the *ART1_Clustering* algorithm. Note that each host represents a large community of organizationally related users. Therefore, by clustering hosts we mean that we are clustering a large number of users. When the ART1_Clustering algorithm stabilizes, a prototype vector is formed for each cluster. The prototype vector of each cluster gives the generalized representation of the URLs most frequently requested by all the members (hosts) of that cluster. The strategy employed in our prefetching scheme is that, whenever a host connects to the server or a proxy, the URLs in the prototype vector corresponding to the cluster to which the host belongs are prefetched. Our technique of prefetching URLs has two advantages: (1) it ensures reasonable utilization of network resources because it prefetches for a community of users instead prefetching requests for a single user and, (2) it prefetches requests with an accuracy as high as 97.778%. We measure the accuracy of our prefetching approach by predicting the URLs for each member of the clusters formed and then we verify our prediction with the access logs recorded for the next 13 days. The procedure describing our prefetching scheme follows.

Procedure: *ART1_based_Prefetching (Host-Id 'Hid' of the host requesting a URL)*
Preprocessing: *Cluster the hosts using the ART1_Clustering algorithm. Each cluster is denoted by C_n, where n is the number of clusters formed. The clusters $C_1, C_2 \ldots, C_k, \ldots, C_n$ are represented by prototype vectors. The prototype vector for the k^{th} cluster is of the form $T_k = (t_{k1}, t_{k2}, \ldots t_{k200})$ where $t_{k\,j=1\ldots200}$ are the top-down weights corresponding to the node k in layer F_2 of the ART1.*

Input: *Host-Id of the host that requests a URL.*
Output: *The array prefetched_URLs[], which contains a list of URLs that are to be prefetched for the host 'Hid'.*
Initialize $count = 0$

Step 1: for n clusters formed using ART1_Clustering algorithm
begin
Step 2: if (Hid is a member of cluster C_k)
begin
Step 3: for $j = 1$ to 200 do //size of the prototype vector T_k-
//- representing cluster C_k
begin
Step4: if $(t_{kj} = 1)$ //where t_{kj} is the j^{th} element of T_k
begin
prefetched_URLs [count]=URLi
count = count + 1
end-if- Step 4
end-for- Step 3
end-if- Step 2
end-for-Step 1
Step 5: return prefetched_URLs []
Step 6: End ART1_based_Prefetching ()

4. Results

In this section, we present the results of our work. We discuss the performance of the ART1 algorithm for clustering. We compare performance of the ART1 algorithm with that of the K-Means clustering algorithm. We present the results of our prefetching scheme in Section 4.4.

4.1. Performance of ART1 clustering technique

We vary the value of the vigilance parameter and measure the quality of clusters obtained by using our ART1 based clustering technique. To measure the quality of clusters obtained, we compute the average inter-cluster distance between the clusters and the average intra-cluster distance within each cluster.

Table 1 shows the number of clusters formed by the ART1 based clustering technique. We observed that the number of the clusters formed increases with the increase in the value of the vigilance parameter. The value of the vigilance parameter is between 0.3 and 0.5.

Table 1. Number of clusters formed by varying the value of the vigilance parameter. The value of the vigilance parameter is varied between 0.30 and 0.50.

Vigilance Parameter	Number of Clusters
0.30	20
0.35	24
0.375	28
0.40	34
0.45	38
0.475	45
0.50	52

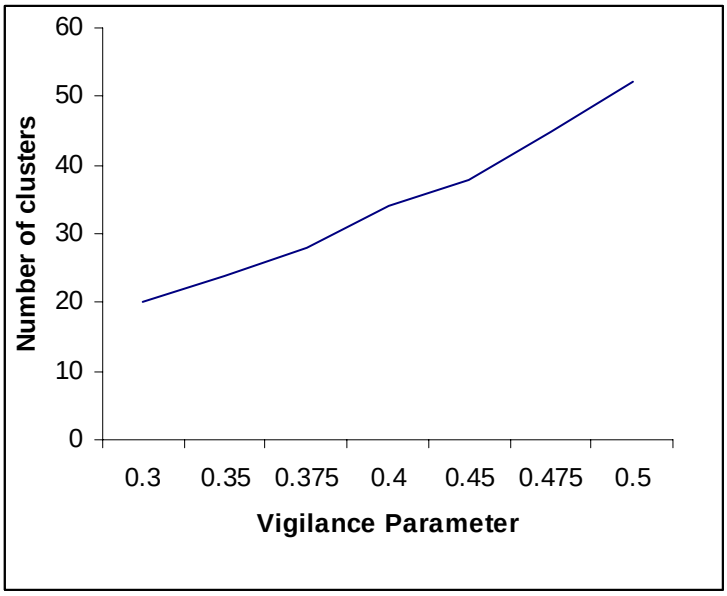

Figure 3. Increase in the number of clusters formed by increasing the vigilance parameter of the ART1 clustering technique.

The plot in Figure 3 illustrates the increase in the number of clusters formed by increasing the value of the vigilance parameter. For the clusters formed by varying the vigilance parameter, we compute the average intra-cluster and average inter-cluster distances. Table 2 gives the values of average inter-cluster distances and average intra-cluster distances by varying the value of the vigilance parameter between 0.3 and 0.5. The plot in Figure 4 illustrates the variation in average inter-cluster distance and intra-cluster distance observed by varying the vigilance parameter between the value 0.3 and 0.5.

Table 2. The average inter-cluster distances and average intra-cluster distances obtained by varying the vigilance parameter between the values 0.3 and 0.5.

Vigilance Parameter	Average Inter-cluster Distance	Average Intra-cluster Distance
0.30	54.322	18.9435
0.35	56.3186	19.7691
0.375	54.3391	18.0365
0.40	57.1537	19.1917
0.45	54.4452	20.0676
0.475	51.2412	20.4521
0.50	54.0638	19.6964

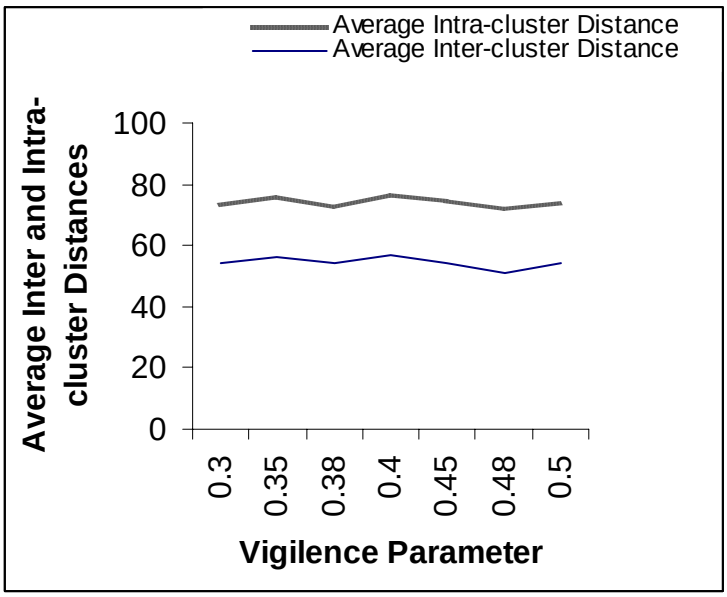

Figure 4. Variation in the values of average intra-cluster distance and average inter-cluster distance of clusters obtained by varying the value of the vigilance parameter between 0.3 and 0.5.

4.2. Performance of K-Means algorithm

The K-Means clustering algorithm is a statistical algorithm for clustering N data points into k disjoint subsets S_j containing N_j data points so as to minimize the sum-of-squares criterion

$$J = \sum_{j=1}^{k} \sum_{n \in S_j} \left\| x_n - \mu_j \right\|^2$$

where x_n is a vector representing the n^{th} data point and μ_j is the geometric centroid of the data points in S_j. In this section, we present the average inter-cluster and average intra-cluster distances obtained by using the K-Means clustering algorithm.

Table 3. The average inter-cluster distances and average intra-cluster distances obtained by varying the number of clusters.

Number of clusters	Average Inter-cluster Distance	Average Intra-cluster Distance
20	66.28	24.2
24	67.077	23.4
28	68.886	19.6687
34	69.55	18.1
38	68.82	17
45	67.6084	14.25
52	65.26	12.667

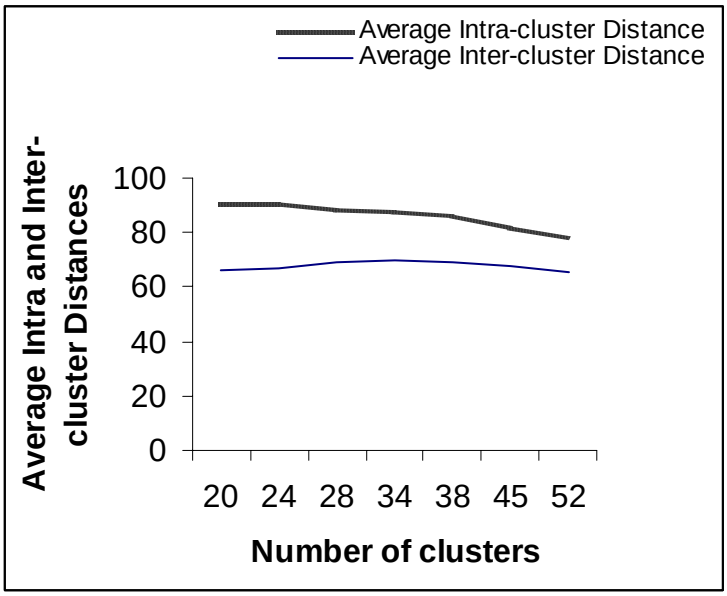

Figure 5. Variation in the average inter-cluster distance and average intra-cluster distance of clusters obtained by varying the number of clusters.

For the same number of clusters obtained by varying the vigilance parameter of the ART1 clustering technique (refer Table 1), we present the performance of the K-Means clustering algorithm in terms of average inter-cluster distance and average intra-cluster distance. Table 3 gives the average inter-cluster distance and average intra-cluster distance obtained by varying the value of K (K represents the number of clusters into which the given data is partitioned by the K-Means algorithm). The plot in Figure 5 illustrates the change in the values of the average inter-cluster distance and intra-cluster distance observed by varying the number of clusters.

4.3. Comparing performance of ART1 and K-Means

In this section, we compare the performance of ART1 clustering technique and K-Means clustering algorithm in terms of inter-cluster distances and intra-cluster distances.

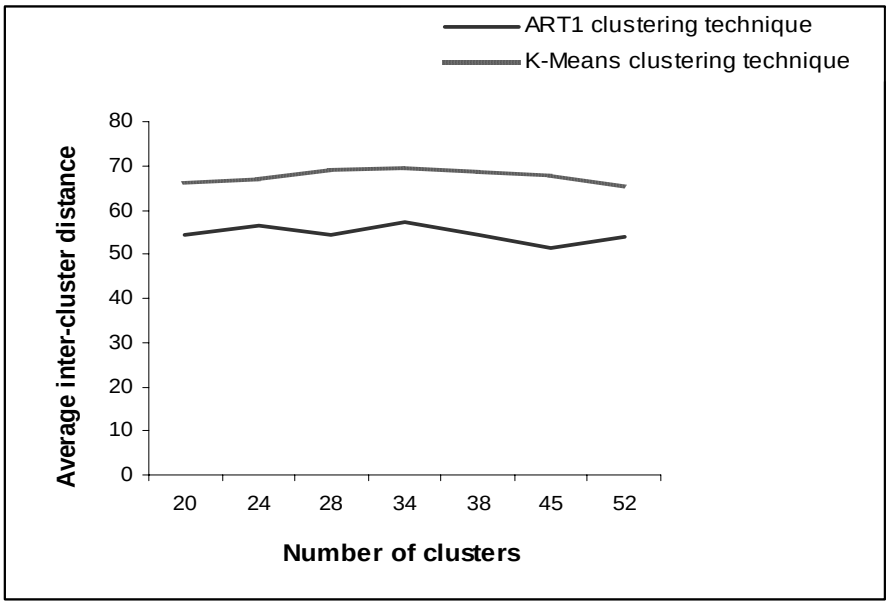

Figure 6. Variation in average inter-cluster distance of clusters formed by the ART1 clustering technique and the K-Means clustering algorithm observed by varying the number of clusters.

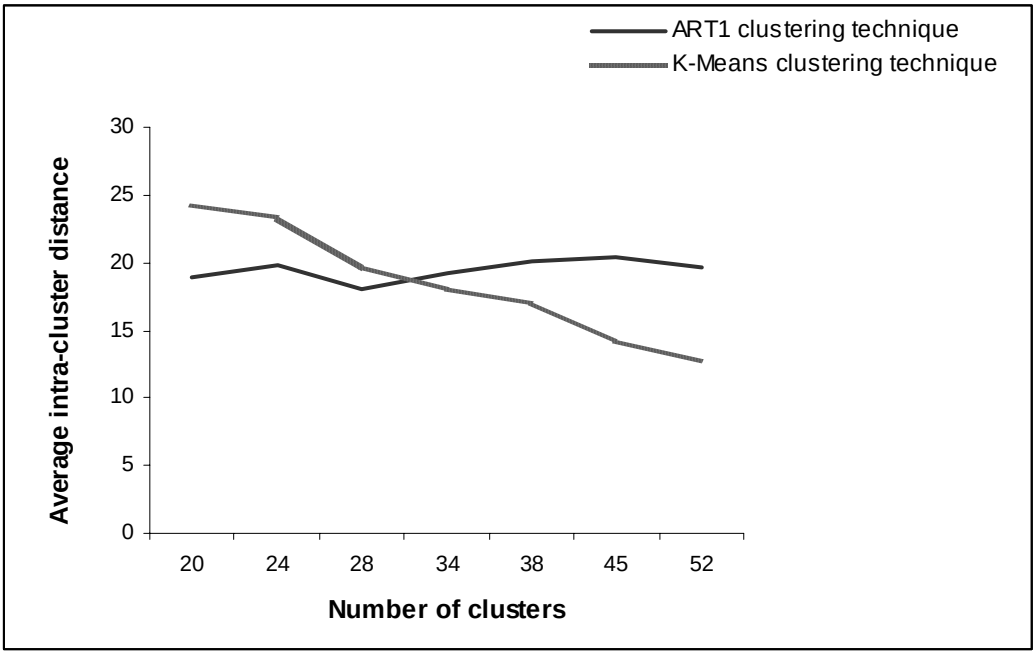

Figure 7. Variation in average intra-cluster distance of clusters formed by the ART1 clustering technique and the K-Means clustering algorithm observed by varying the number of clusters.

The plot in Figure 6 illustrates the variation in average inter-cluster distance between clusters formed by the ART1 technique and the K-Means clustering algorithm, observed by increasing the number of clusters. Notice that the average inter-cluster distances obtained by using the two algorithms vary at a steady rate indicating that there is not much difference in performance of the two algorithms in terms of inter-cluster distance. However, the plot in Figure 7 shows that the average intra-cluster distances of clusters obtained by using the K-Means clustering algorithm vary from 12.67 to 24.2 while the average intra-cluster distances of clusters obtained by using the ART1 clustering technique vary from 18.04 to 20.45. This observation indicates that variance within the clusters formed by ART1 is quite uniform compared to the variance within clusters formed by K-Means algorithm, which increases with the increase in the number of clusters.

4.4. Prefetching Results

We assess the performance of our prefetching scheme using two parameters: (1) Hits and (2) Accuracy. Hits indicate the number of URLs that are requested from the prefetched URLs, and accuracy is the ratio of hits to the number of URLs prefetched.

Table 4. Results of our ART1 based prefetching scheme. Each row in this table represents a cluster of hosts. The "Members" column shows the hosts that are clustered together. "Number of URLs Prefetched" gives the number of URLs prefetched by our prefetching scheme. "Requested URLs" gives the number of URLs requested by the hosts during the period for which we made prediction.

Members	Number of URLs Prefetched	Requested URLs		Hits	Accuracy %
0, 2, 5, 7, 8, 13, 14, 58	45	0	173	44	97.778
		2	189	43	95.556
		5	150	42	93.33
		7	160	44	97.778
		8	192	42	93.33
		13	200	43	95.556
		14	200	44	97.778
		58	122	39	86.666
3, 4, 10, 11, 18	65	3	200	60	92.30
		4	200	56	86.15
		10	168	56	86.15
		11	200	57	87.69
		18	13	8	0.123
6, 12, 15, 16	39	6	168	37	94.871
		12	181	37	94.871
		15	-	-	-
		16	126	32	82.05
1 , 9, 67	38	1	-	-	-
		9	20	35	92.12
		67	13	34	89.47

In Table 4, we show the result of our prefetching scheme. We prefetch the URLs for each host and verify the accuracy of our prefetching scheme by comparing predicted URLs with the access logs for the period of next 13 days. The results presented in Table 4 are obtained by assigning a value of 0.38 to the vigilance parameter of ART1 clustering algorithm. The prediction accuracy of our prefetching technique ranges from 82.05 to 97.78% (shown in Table 4). There has been a deviation in three cases where the hosts have not requested any URLs that were prefetched. However, the average prediction accuracy of our prefetching scheme is 92.3% (excluding the three exceptional cases in which the hosts did not request the prefetched URLs), which is considerably high.

5. Conclusion

In recent years, there has been considerable research in exploring novel methods and techniques to group users based on the information hidden in their browsing patterns. In this paper, we present our approach to group hosts (each host represents an organizationally related group of users) according to their Web request patterns. We use the ART1 clustering algorithm to cluster these communities of users. We compare the performance of the ART1 clustering with that of the K-Means clustering algorithm and show that the ART1 clustering performs better than the K-Means clustering algorithm in terms of the intra-cluster distances. We present a prefetching scheme that uses ART1 clustering. Using our prefetching scheme, were able to obtain prediction accuracy as high as 97.78 % (on NASA access logs spanning the timeframe from July 16, 1995 to July 29, 1995).

6. References

[1] Rangarajan S. K., "Unsupervised Learning Techniques for Web Domain Clustering and Its Application for Prefetching." MS Thesis, Louisiana Tech University, June 2002.

[2] Barbara M., "ART1 and Pattern Clustering." *In Proceedings of the 1988 Connectionist Models Summer 1998*, Published by M.Kaufmann, San Mateo, CA, pages 174-185, 1998.

[3] Heins, Lucien G., Tauritz, and, Daniel R., "Adaptive Resonance Theory (ART): An Introduction." Internal Report 95-35, Department of Computer Science, Leiden University, pages 174-185, The Netherlands, 1995.

[4] Cooley R., Mobasher B., and Srivatsava J., "Web Mining: Information and Pattern Discovery on the World Wide Web." *ICTAI'97*, 1997.

[5] Phoha V. V., Iyengar S.S., and Kannan R., "Faster Web Page Allocation with Neural Networks," *IEEE Internet Computing*, Vol. 6, No. 6, pp. 18-26, December 2002.

[6] Garofalakis M. N., Rastogi R., Sheshadri S., and Shim K., "Data mining and the Web: past, present and future." In *Proceedings of the second international workshop on Web information and data management, ACM*, 1999.

[7] Fu Y., Sandhu K., and Shih M., "Clustering of Web Users Based on Access Patterns." *International Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*, San Diego, CA, 1999.

[8] Zhang T., Ramakrishnan R., and Livny M., "Birch: An Efficient Data Clustering Method for Very Large Databases." In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 103-114, Montreal, Canada, June 1996.

[9] Cadez I., Heckerman D., Meek C., Smyth P., and Whire S., "Visualization of Navigation Patterns on a Website Using Model Based Clustering." Technical Report MSR-TR-00-18, Microsoft Research, March 2002.

[10] Paliouras G., Papatheodorou C., Karkaletsis V., and Spyropoulos C.D., "Clustering the Users of Large Web Sites into Communities." In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 719-726, Stanford, California, 2000.

[11] Xie Y., and Phoha V. V., "Web User Clustering from Access Log Using Belief Function." *K-CAP' 01*, British Columbia, Canada, October 2001.

[12] Loon T. S., and Bharghavan V., "Alleviating the Latency and Bandwidth problems in WWW Browsing." In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS '97)*, December 1997.
[13] Ibrahim T. I., and Xu C. Z., "Neural Nets based predictive Pre-fetching to tolerate WWW Latency". In *Proceedings of the 20th International Conference on Distributed Computing Systems, IEEE*, Taipei, Taiwan, Republic of China, April 2000.
[14] Fan L., Cao P., and Jacobson Q., "Web Prefetching between Low-Bandwidth Clients and Proxies: Potential and Performance." In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'99)*, Atlanta, GA, May 1999.
[15] Markatos E. P., and Chronaki C. E., "A Top-10 Approach to Prefetching on the Web." In *Proceedings of the Eighth Annual Conference of the Internet Society (INET'98)*, Geneva, Switzerland, July 1998.
[16] Padmanabhan V. N., and Mogul J. C., "Using Predictive Prefetching to Improve World Wide Web Latency." *ACM Computer Communication Review*, Vol. 26, No.3, page 2336, July 1996.
[17] Tian W., Choi B., and Phoha V. V, "An Adaptive Web Cache Access Predictor Using Neural Network." In *Proceedings of the15th International Conference on Industrial and Engineering. Applications of Artificial Intelligence and Expert Systems*, pages 450-459, IEA/AIE, Cairns, Australia, June 2002.
[18] NASA server logs file available at http://ita.ee.lbl.Gov/html/contrib/NASA-HTTP.html (Last accessed June 2002).