IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 22, 2021, accepted July 28, 2021, date of publication August 9, 2021, date of current version August 17, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3103804

An Improved Web Caching System With Locally Normalized User Intervals

KRISHNAKUMAR VAITHINATHAN[1], JULIAN BENADIT PERNABAS[2], (Senior Member, IEEE), LATHA PARTHIBAN[3], BHANU SHRESTHA[4], GYANENDRA PRASAD JOSHI[5], AND HYEONJOON MOON[5]

[1]Department of Computer Engineering, Karaikal Polytechnic College, Karaikal 609609, India
[2]Department of Computer Science and Engineering, School of Engineering and Technology, CHRIST (Deemed to be University), Kengeri Campus, Bangalore, Karnataka 560074, India
[3]Department of Computer Science and Engineering, Community College, Pondicherry University, Puducherry 605014, India
[4]Department of Information Contents, Kwangwoon University, Seoul 01897, South Korea
[5]Department of Computer Science and Engineering, Sejong University, Seoul 05006, South Korea

Corresponding author: Hyeonjoon Moon (hmoon@sejong.ac.kr)

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2020R1A6A1A03038540) and by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2019-0-00136, Development of AI-Convergence Technologies for Smart City Industry Productivity Innovation).

ABSTRACT Caching is one of the most promising areas in the field of future internet architecture like Information-centric Networking, Software Defined Networking, and IoT. In Web caching, most of the web content is readily available across the network, even if the webserver is not reachable. Several existing traditional caching methods and cache replacement strategies are evaluated based on the metrics like hit ratio and byte hit Ratio. However, these metrics have not been improved over the period because of the traditional caching policies. So, in this paper, we have used an intelligent function like locally normalized intervals of page visit, website duration, users' interest between user groups is proposed. These intervals are combined with multiple distance metrics like Manhattan, squared Euclidean, and 3-,4-,5-norm Minkowski. In order to obtain significant common user navigation patterns, the clustering relation between the users using different intervals and distances is thoroughly analyzed. These patterns are successfully coupled with greedy web cache replacement strategies to improve the efficiency of the proposed web cache system. Particularly for improving the caching metrics more, we used an AI-based intelligent approach like Random Forest classifier to boost the prefetch buffer performance and achieves the maximum hit rate of 0.89, 0.90, and byte hit rate of 0.87, 0.89 for Greedy Dual Size Frequency and Weighted Greedy Dual Size Frequency algorithms, respectively. Our experiments show good hit/byte hit rates than the frequently used algorithms like least recently used and least frequently used.

INDEX TERMS Locally normalized intervals, proxy, web cache, pre-fetching.

I. INTRODUCTION

Web prefetching and caching framework mitigates the load on the source servers by acting as a delegate between the web clients and servers. It attempts to identify the patterns of individual web requests, stores them, and acts as a proxy by responding to the demands of the clients in the place of servers. Over the years, researchers develop several implementations [1]–[8] of web proxy caches. A set of presenter devices [6] can be used as a proxy between the collaboration server and the clients. The requests are delegated based on the response of the server and the presenter context. The devices establish a multiple participant web session between themselves and the individual sessions. The server progressively creates responses based on the context of the presenter. These joint sessions enable each participator session to visualize the responses according to their contexts independently. In the field of web application computing [7], the proxy requests/receives the resources from the web application and injects an executable script with it and sent it to the client. Along with the requested resource, the client can also execute the script and obtain the analytic information about the

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Marozzo.

servers/resources in the network. This proxy types can be used as enterprise solutions for monitoring resources in vast networks. The scripts are used to collect various monitoring metrics from a specific application/resource pool. A reverse web proxy [9] is used to join the web application and a portal. It removes the necessity to identify information facilitating the boarding process physically. Data associated with network traffic evaluation, deep packet analysis are used to identify one or more attributes of a specific application. The data discovery engine uses those attributes to identify a set of patterns that explain the overall configuration characteristics of an application. This set of configurations are used to merge the application into the reverse proxy web portal. Web proxies help reduce the latency in displaying webpages [10]. Proxy server delegates between the browser and the origin server through the internet. When the browser requests a resource, a page structure validates the resource with respect to the time in the browser, and when the same resource is requested again, the page structure validates the cached pages to the browser and thereby reducing the time taken to render a page.

Web proxies caching is an interesting technique that improves the performance of the servers and the networks. Substituting a page in the cache needs careful inspection as it directly affects its performance. The replacement methods are categorized into two types: first, the feature measures which includes first in first out(FIFO), least recently used(LRU), least frequently used (LFU), content type, re-accessed probability of web resource, history of access or combination of one/more measures, etc. and the second is based on smart forecast/guessing techniques. Support vector machine(SVM), logistic regression, artificial neural networks, random forest, naive bayes are the most commonly used machine learning algorithms for this purpose. The prediction probability of a web page to be placed in the cache is one of the critical and essential aspects of replacement policies. The Greedy dual size frequency(GDSF) algorithm [11] uses multiple features combination coupled with SVM selects more suitable replacement pages than other traditional algorithms. The combination of pre-fetching and caching [12] improves bandwidth utilization and reduce network latencies. Preloading can be done through the clustering of web user preferences. These priorities give an overall idea of the usage patterns of different users. The big clusters indicate the higher revisiting probability of a particular web page.

Finding the best subset [13], [14] from a pool of features will always help find the optimum replacement. Different types of feature selection modules like the best first method, incremental wrapper feature selection, particle swarm optimization, J48 decision trees, and Naive bayes are successfully used in singling out the collection of features that are used to determine the best substitution in the cache. Web object frequency, web objects + internal request regularity, recency, web object distribution based on sliding window are some of the features that suit best for this purpose. Collaborative caching [15] in structured peer to peer systems combined with intelligent prediction algorithms are useful in boosting the performance of web proxy cache and produces the best accuracy than traditional algorithms like LRU, LFU, etc. Moreover, this structured system uses hap map functions to locate peers and shared data references which enable them to give less latency and load balancing properties.

Another approach includes tree augmented naive bayes (TANB) [16] algorithm gives better results and improves the performance of traditional policies like LRU and GDSF. According to the recurrent sliding window, frequency and recency of the web objects are calculated/preprocessed and fed as input to TANB to find out the page with the highest probability of revisiting in the future. The results of LRU and GDSF can also be used as input to TANB to boost their accuracies.

Weighted random indexing [17]–[19] with a data mining classifier boosts the standard policies like Greedy-Dual Size (GDS), GDSF, and Greedy Dual*(GD*). Typically, these traditional greedy policies use the fields key, frequency, size, and the inflation factor in finding the web object. When the above measures are fused with the ratio between the time difference of present and preceding request and the difference of the current reference and the last reference time of an object in the cache, there is a significant reduction in the number of replacement of objects in the cache. It implies that the number of cache hits is greater than the number of cache miss.

Web access time [20] is also one of the important features in determining the users' preferences and requirements. It is noted that the user page viewing patterns are connected with localized time, for example, in online e-commerce websites, many users share homogeneous browsing patterns during festive seasons; recommendation engines suggest their choices based on similar navigational models over the same span of time; in e-learning environments, students prefer summer months in registering for particular courses; in web caching and pre-fetching, finding users with close behavioral traits will be useful in reducing origin server load.

A. MOTIVATION

The traditional web caching plays an important role for the success of the Internet. Due to the exponential growth of the internet traffic, the data traffic and the information shared across the internet is approximately more than 34.7 million messages and the expected data rate in 2020 will be around 2.3 zettabytes. However due to the increasing volume of the data day by day, the need to find the content from the cache and retrieve the most relevant data from the cache is more challenging. However, the caching technology has been started to use in the future internet architecture like, Information centric Networking (ICN), Content Distribution Network (CDN), Named Data Networking (NDN), etc., but there are still many problems which are not addressed in the field of caching technology like allocation strategy, replacement strategy and utilization strategy. Also, over the period of time lot of research has been carried in the traditional Web caching environment, in improving the content caching, storage management and cache replacement policies. But

in general, there is no much research work on the user's access behavior by considering the frequency of the web page, Page visiting time, duration of the website based on their time and frequency. In addition, the behavior of web users access based on the time and grouping of web users based on the time has not been incorporated for the web caching technique. In addition, the impact of web users clustering based on the time has not been addressed in the caching technique. Further in most of the caching technique, the optimal replacement strategies like Weighted Greedy Dual Size frequency (WDSF) have not been incorporated with intelligent mechanism which results in better caching performance compared to other cache replacement policies. This motivates the need for the incorporating the Web user access framework based on locally normalized users and also, we have designed an integrated caching and prefetching based on the Random Forest Tree classifier (RFT) resulting in optimal cache decision by predicting the popular web content for proactive caching.

B. OBJECTIVES

The first objective of the work is to emphasize more on users level access of the intelligent framework where the cardinality of the user's access time, time frames, visiting the pages are identified to get the users access behavior with respect to time. Moreover, the various different distance measures are used for normalizing the web users' access frame work to identify the similarities in users access behavior and to capture dissimilarities measures of the page visiting and website duration arrays. This locally normalized distance measures enhance the Web users clustering performance in terms of user's access time.

The second objective of the research work is to enhance the various caching strategy by integrating the Web users' access frame work with caching algorithm. The proposed web caching algorithm Greedy Dual Size Frequency integrates with the RFT classifier to form an intelligent web caching system by predicting the popular web page contents resulting in optimal cache decision for proactive caching. This intelligent caching system increases the caching efficiency in every scenario especially in terms of prediction and hit rate and also simulation shows that modified GDSF called WGDSF outperforms with other traditional caching algorithm.

This paper is organized as follows: section.II explains the web user framework and its data structure, section.III deals with proposed interval measures, section.IV explains the dataset and components of the proposed system, and finally the section.V deals with the experimental details and results.

II. WEB USER ACCESS FRAMEWORK

Let's consider an access framework for web users where the log files record the users' navigational behaviour periodically. Here, a specific timeframe is used and is referred to a certain period on which the examination of users' access is considered. For example, in the monthly and annual logs, the timeframe of concern can be taken as one day or one month respectively. On the whole, these time selections depend on the log file intervals. The duration of the webpage indicates the total time spent by a user on that page. Also in the proposed framework, the demands of the individual application decide the regulation of the timeframes in the respective log files. Hence, the cardinality of the users(U), timeframes(T), pages(P), and websites(W) are represented as c_n, c_t, c_p, *and* c_w respectively. Here, the collection of websites(W) is given by:

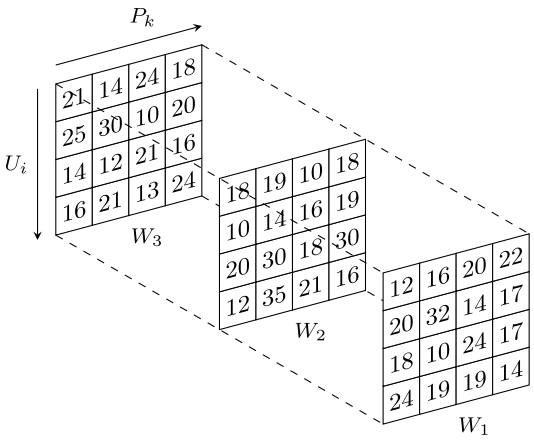


FIGURE 1. Page visitation array for 3 websites W_1, W_2, W_3 with 4 users U_1, U_2, U_3, U_4 and 4 pages P_1, P_2, P_3, P_4 respectively.

$$W = \bigcup_{i}^{c_w} \bigcup_{j}^{c_p} P_{ij} \tag{1}$$

Here, each website W_i is considered to be a set of pages P_j. The group of users is $U = u_1, \ldots, u_c$. T_{wp}, D_{wpt} represents the page visitation(size: $c_w \times c_n \times c_p$) and the webpage duration(size: $c_w \times c_n \times c_p \times c_t$) tables respectively for each user in group U and websites W. $\Delta_{U_{wp}}$ indicates the distance between users and the pages. $\Delta_{U_{wpt}}$ shows the distances between the websites with respect to individual pages, time, and users.

To understand all the characteristics of users' access time and page visitations, two different data aspects are defined for the proposed framework. The first array stores the users' visit to specific pages and the second one is saved as a multi-dimensional array and they record the users' access page time of multiple websites' pages.

The temporal users' page access relationships are given by:

1) **Page visitation array:** A page visit array is given by T^i_{wp} where $i = 1, \ldots, c_n$ represents the frequency of the independent users in accessing pages of different websites. This is a 3-dimensional array of $c_w \times c_n \times c_p$ actions. It is shown as:

$$T^{(i,q,k)}_{wp} \quad where\ 1 \leq q \leq c_w;\ 1 \leq k \leq c_p \tag{2}$$

Eqn.2 indicates the frequencies of the visits of page k by a user i in website q. T_{wp} is the organized data of all the values of $T^{(i,q,k)}_{wp}$ with size $c_n \times c_w \times c_p$.

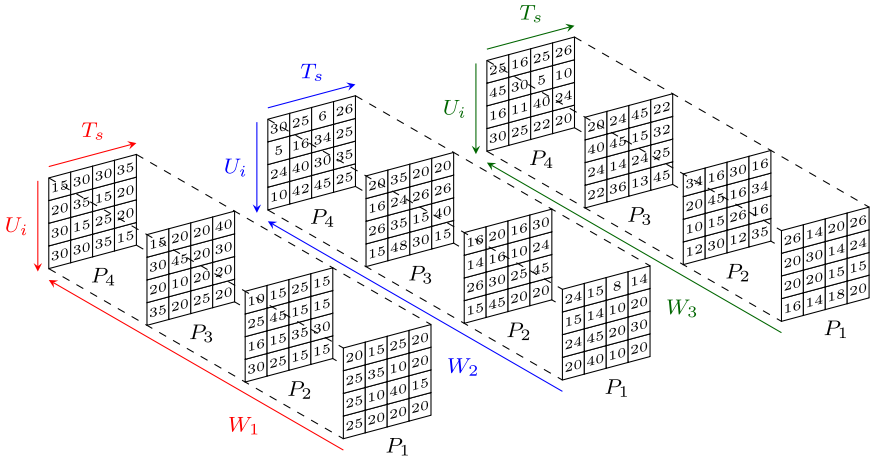


FIGURE 2. Website duration array for 3 websites W_1, W_2, W_3 with 4 users U_1, U_2, U_3, U_4, 4 pages P_1, P_2, P_3, P_4 and 4 timeframes T_1, T_2, T_3, T_4 respectively.

For example $T_p^{(2,3,4)} = 16$ identifies the user(U_2) performed 16 visits to page 4 of website 3.

2) **Website Duration array:** A website duration array is given by D_{wpt}^{i} where i represents the duration of the independent users with respect to access time of individual pages of a complete website. This is a 4-dimensional table with $c_n \times c_w \times c_p \times c_t$ elements.

$$D_{wpt}^{(i,q,r,s)} \quad where\ 1 \le q \le c_w;\ 1 \le r \le c_p; \times 1 \le s \le c_t \qquad (3)$$

Eqn.3 indicates the information of the user i visiting the r pages of q websites for the duration s. $D_{wpt}^{(i,q,:,:)}$ indicates time span details for q websites with user i. $D_{wpt}^{(i,q,r,:)}$ indicates time span information of r pages of q websites with user i. D_{wpt} is the organized data of all the values of $D_{wpt}^{(i,q,r,s)}$ with size $c_n \times c_w \times c_p \times c_t$. For example $D_{ptv}^{(2,3,2,1)} = 10$ identifies the user(U_2) who visits page 2 of website 3 for a duration 10 in the timeframe 1.

It is also noted that the values of page visit arrays(T_{wp}^{i}), has a temporal relation with the website duration arrays($D_{wpt}^{i,q,r,s}$). i.e., the number of visits in $T_{wp}^{(i,q,k)}$ is performed in an average duration of the website array which is indicated by $\frac{\sum_{j=1}^{c_t} D^{(i,q,k,j)}}{c_t}$. In summary, as the actions of the user depend upon the navigation time, the proposed structure uses the page visit and website duration information to identify the respective users' preferences of the visited pages and the locality of access time respectively. Moreover, the website arrays store the page and time aspects of multiple users on different websites simultaneously.

III. MULTIPLE FACTOR DISTANCE MEASURES

Distance measures play an important role in the clustering process. The relevance of any two patterns can be found out using the dissimilarity distances between them [21]. K-means [22] is one of the oldest and popular clustering methods and can be efficiently used with different distance metrics [23]–[25], like cosine, Manhattan, Minkowski, squared Euclidean, and Chebychev. The difference between the users is effectively captured by means of the dissimilarity measure of the page visiting and website duration arrays. Here, three types of distance measures such as Manhattan(Δ^1), P-norm Minkowski(Δ^2), squared Euclidean(Δ^3) are used. The Minkowski distances are calculated for 3-,4-,5-norms.

1) **Users' Page visit interval**: Let us consider only the page accesses, then the interval between any two users are calculated for each of the c_p pages with different websites. The interval ($\Delta_{U_{wp}}^{x}[u_a, u_b]$) between two users u_a and u_b of website w_c is calculated using the page access arrays of the two users, $T_{wp}^{(a,c,:)}$ and $T_{wp}^{(b,c,:)}$ where $u_a, u_b \in U$, x is the distances, and w_c is website respectively. The interval is represented by different distances as follows:

$$\Delta_{U_{wp}}^{1}[u_a, u_b] = \left| T_{wp}^{(a,c,:)} - T_{wp}^{(b,c,:)} \right| \qquad (4)$$

$$\Delta_{U_{wp}}^{2}[u_a, u_b] = \left(|T_{wp}^{(a,c,:)} - T_{wp}^{(b,c,:)}|^P \right)^{1/P} \qquad (5)$$

$$\Delta_{U_{wp}}^{3}[u_a, u_b] = \left\| T_{wp}^{(a,c,:)} - T_{wp}^{(b,c,:)} \right\|^2 \qquad (6)$$

2) **Users' Website duration interval**: Let us consider both the timeslot and page accesses, then the interval between any two users are calculated separately for each of the c_p pages with different websites and timeslots. This indicates that the similarity behaviour of two users is based upon their identical visits in pages and timeframes. The term $\Delta_{U_{wpt}}^{x}[u_a, u_b]$ denotes the interval between the website duration arrays $D_{wpt}^{(a,c,:,:)}$ and $D_{wpt}^{(b,c,:,:)}$ representing the users u_a and u_b. The distance metrics are calculated for the corresponding timeframes of individual pages of websites and are

given by:

$$\Delta^1_{U_{wpt}}[u_a, u_b] = \sum_{j=1}^{c_p} \left| D^{(a,c,j,:)}_{wpt} - D^{(b,c,j,:)}_{wpt} \right| \quad (7)$$

$$\Delta^2_{U_{wpt}}[u_a, u_b] = \sum_{j=1}^{c_p} \left(|D^{(a,c,j,:)}_{wpt} - D^{(b,c,j,:)}_{wpt}|^P \right)^{1/P} \quad (8)$$

$$\Delta^3_{U_{wpt}}[u_a, u_b] = \sum_{j=1}^{c_p} \left\| D^{(a,c,j,:)}_{wpt} - D^{(b,c,j,:)}_{wpt} \right\|^2 \quad (9)$$

3) **Locally Normalized Users' Page visit interval**: For a specific website(w_c), this interval($\overline{\Delta^x_{U_{wp}}}[u_a, u_b]$) between the two users($u_a$ and u_b) is calculated based on the local normalization of the page visitation frequencies($T^{(a,c,:)}_{wp}$ and $T^{(b,c,:)}_{wp}$). There are two steps to this process: first, the normalized local arrays of two users are obtained from the page visit array, it is denoted as:

$$\overline{T^{(a,c,j)}_{wp}} = \frac{T^{(a,c,j)}_{wp}}{\frac{\sum_{k=1}^{c_p}(T^{(a,c,k)}_{wp} + T^{(b,c,k)}_{wp})}{c_p * 2}} \quad where\ 1 \le j \le c_p \quad (10)$$

$$\overline{T^{(b,c,j)}_{wp}} = \frac{T^{(b,c,j)}_{wp}}{\frac{\sum_{k=1}^{c_p}(T^{(a,c,k)}_{wp} + T^{(b,c,k)}_{wp})}{c_p * 2}} \quad where\ 1 \le j \le c_p \quad (11)$$

and second, the different distance metrics are applied to the normalized array to obtain the interval, they are given by:

$$\overline{\Delta^1_{U_{wp}}}[u_a, u_b] = \left| \overline{T^{(a,c,:)}_{wp}} - \overline{T^{(b,c,:)}_{wp}} \right| \quad (12)$$

$$\overline{\Delta^2_{U_{wp}}}[u_a, u_b] = \left(|\overline{T^{(a,c,:)}_{wp}} - \overline{T^{(b,c,:)}_{wp}}|^P \right)^{1/P} \quad (13)$$

$$\overline{\Delta^3_{U_{wp}}}[u_a, u_b] = \left\| \overline{T^{(a,c,:)}_{wp}} - \overline{T^{(b,c,:)}_{wp}} \right\|^2 \quad (14)$$

4) **Locally Normalized Users' Website duration interval**: This interval($\overline{\Delta^x_{U_{wpt}}}[u_a, u_b]$) between the two users($u_a$ and u_b) is calculated based on the local normalization of the website duration arrays($D^{(a,c,:,:)}_{wpt}$ and $D^{(b,c,:,:)}_{wpt}$) for a specific website(w_c). The normalized distance arrays are denoted by:

$$\overline{D^{(a,c,j,l)}_{wpt}} = \frac{D^{(a,c,j,l)}_{wpt}}{\frac{\sum_{k=1}^{c_t}(D^{(a,c,j,k)}_{wp} + T^{(b,c,j,k)}_{wp})}{c_t * 2}} \times where\ 1 \le j \le c_p\ and\ 1 \le l \le c_t \quad (15)$$

$$\overline{D^{(b,c,j,l)}_{wpt}} = \frac{D^{(b,c,j,l)}_{wpt}}{\frac{\sum_{k=1}^{c_t}(D^{(a,c,j,k)}_{wp} + T^{(b,c,j,k)}_{wp})}{c_t * 2}} \times where\ 1 \le j \le c_p\ and\ 1 \le l \le c_t \quad (16)$$

The distance metrics for the above equations is represented by:

$$\overline{\Delta^1_{U_{wpt}}}[u_a, u_b] = \left| \overline{D^{(a,c,:,:)}_{wpt}} - \overline{D^{(b,c,:,:)}_{wpt}} \right| \quad (17)$$

$$\overline{\Delta^2_{U_{wpt}}}[u_a, u_b] = \left(|\overline{D^{(a,c,:,:)}_{wpt}} - \overline{D^{(b,c,:,:)}_{wpt}}|^P \right)^{1/P} \quad (18)$$

$$\overline{\Delta^3_{U_{wpt}}}[u_a, u_b] = \left\| \overline{D^{(a,c,:,:)}_{wpt}} - \overline{D^{(b,c,:,:)}_{wpt}} \right\|^2 \quad (19)$$

5) **Locally Normalized Users' Interest Interval**: The visit frequency and the duration are the important factors that are helpful in identifying the users' preferences and their access behaviour. Higher the frequency and the duration indicates the better interest of the user on that page. Here, the page visit and duration are 3-dimensional and 4-dimensional arrays. They cannot be directly combined. The duration array has to be merged internally so that it can be combined with the visit array. It is given by:

$$\widehat{D}^{(i,q,r)}_{wt} = \sum_{k=1}^{c_p} D^{(:,:,k,:)}_{wpt} \quad where\ 1 \le i \le c_n; \times\ 1 \le q \le c_w;\ 1 \le r \le c_t \quad (20)$$

After applying the Eqn.20, the page dimension can be safely removed from the duration array($D^{(:,:,:,:)}_{wpt}$). This interval($\overline{\Delta^x_I}[u_a, u_b]$) of the interest between the two users(u_a and u_b) is calculated based on the local normalization of the page visitation frequencies($\overline{T^{(a,c,:)}_{wp}}$ and $\overline{T^{(b,c,:)}_{wp}}$) and website duration arrays($\overline{\widehat{D}^{(a,c,:)}_{wt}}$, and $\overline{\widehat{D}^{(b,c,:)}_{wt}}$) for a specific website(w_c). The normalized frequencies and duration can be easily derived using the Eqns.(10, 11, 15, 16). The distance metrics for the interest interval is given by:

$$\overline{\Delta^1_I}[u_a, u_b] = \frac{2 \times |A_1| \times |B_1|}{|A_1| + |B_1|} \quad (21)$$

$$\overline{\Delta^2_I}[u_a, u_b] = \frac{2 \times \left(|A_1|^P\right)^{1/P} \times \left(|B_1|^P\right)^{1/P}}{\left(|A_1|^P\right)^{1/P} + \left(|B_1|^P\right)^{1/P}} \quad (22)$$

$$\overline{\Delta^3_I}[u_a, u_b] = \frac{2 \times \|A_1\|^2 \times \|B_1\|^2}{\|A_1\|^2 + \|B_1\|^2} \quad (23)$$

where

$$A_1 = \overline{T^{(a,c,:)}_{wp}} - \overline{T^{(b,c,:)}_{wp}}$$

and

$$B_1 = \overline{\widehat{D}^{(a,c,:)}_{wt}} - \overline{\widehat{D}^{(b,c,:)}_{wt}}$$

IV. DATA AND METHODS

A. DATASET

Our method is tested using the open source Websites' access log files provided by the computer science department of Boston University [26]. It consists of multiple session files of 762 users and 1,143,839 page transfer requests. From the above website logs, the log period of Nov 1994 to May 1995 is considered for this study. As the user ids in the dataset are renumbered to protect the privacy of the respective users, the data cannot be directly applied to our method. So the

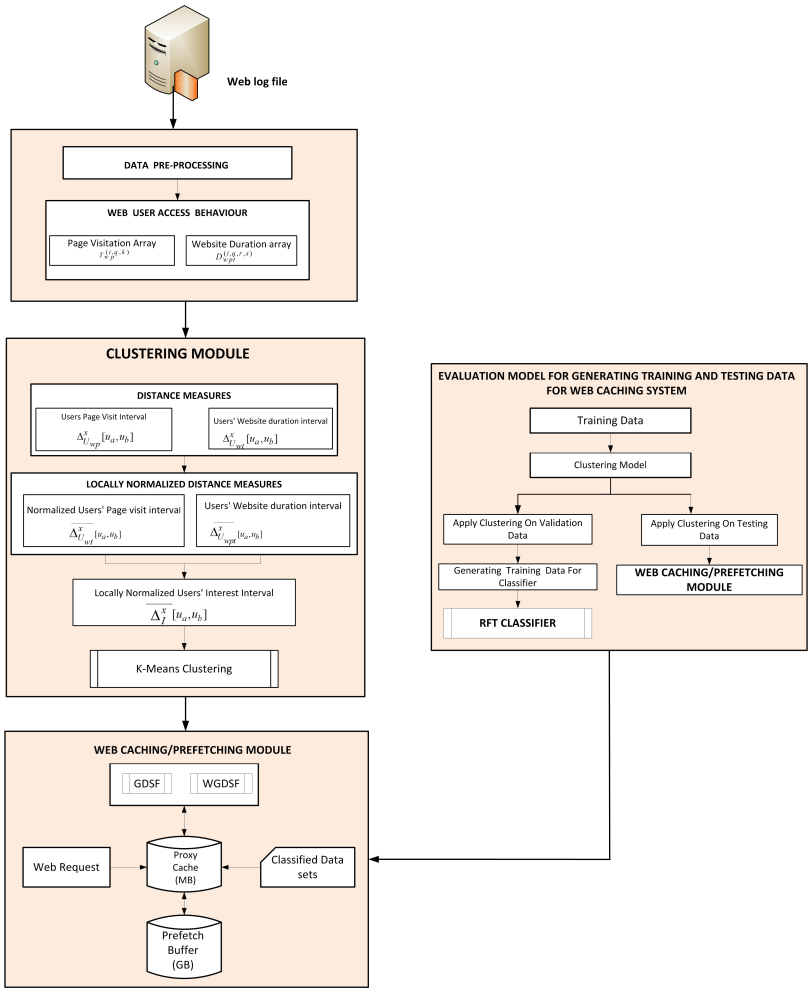


FIGURE 3. Integrated web caching/prefetching system.

pre-processing of log files is essential to further test our method.

B. PRE-PROCESSING

Data pre-processing is an important step to improve the quality of relevant information on the website logs. The log session files contain multiple information such as session starting time, machine name, time of request, user id, web URL, document size, and the file retrieval time in seconds. All the records which have the file retrieval time greater than zero are filtered out and considered for further processing. There are 5211 distinct web URLs with a total of around 211,000 entries in this collection of log files. First, unique IDs are assigned to each user($U = u_1, \ldots, u_{c_n}$). The data modeling steps for four dimension website duration arrays(D_{wpt}) of size $c_n \times c_w \times c_p \times c_t$ is as follows:

1) Distinct c_p web URLs are randomly assigned each w_q. The website group is defined in the Eqn.1.
2) From the total web URL list, each user U_i is randomly allocated with multiple website requests.
3) Different timeslots(c_t) are fixed based on the starting time of the session log files.
4) First, the total website duration without timeslot division($D_{wpt}^{(U_i,W_q,P_k)}$) is calculated as:

$$D_{wpt}^{(U_i,W_q,P_k)} = (time\ of\ request) + (file\ retrieval\ time) - (session\ starting\ time) \quad (24)$$

5) Then, the total website duration with timeslots(refer Eqn.3) $D_{wpt}^{(Users,Websites,Pages,Timeslots)}$ is obtained by extending the Eqn.24, i.e the $D_{wpt}^{(U_i,W_q,P_k)}$ is randomly splitted into c_t timeframes.

The data modeling steps for three dimension page visitation arrays(T_{wp}) of size $c_n \times c_w \times c_p$ is as follows:

1) The first two steps are the same as the above process.
2) Page visitation array($T_{wp}^{(U_i,W_q,P_k)}$) is calculated as:

$$T_{wp}^{(U_i,W_q,P_k)} = D_{wpt}^{(U_i,W_q,P_k)} / 5 \quad (25)$$

TABLE 1. Parameters of the cache buffer in web caching combined with pre-fetching algorithm.

Notation	Description
P_k	Web page
S_{P_k}	Size of the Web page
α_{proxy}	Cache buffer
$S_{\alpha_{proxy}}$	Cache buffer size
$\beta_{prefetch}$	Prefetch buffer
δ_{origin}	Origin Server
L_{Inf}	cache buffer: Inflation factor for avoiding cache pollution
$freq(P_k)$	Previous web page access frequency
$FREQ(P_k)$	Current web page access frequency
$K_{n-1}^{(P_k)}$	Previous key value of the web page(P_k)
$\Delta CP_t^{(P_k)}$	Rate of change in time between the current and previous requests for the Web page
$\beta_t^{(P_k)}$	Current time of reference of the web page
$\chi_t^{(P_k)}$	Last reference time of web page
$K_n^{(P_k)}$	Current key value of the web page
$\theta_i^{(P_k)}$	Pre-fetch time

i.e., the frequencies of visits are based on Eqn.24 and five duration units are assumed to be equal to one unit of the visit.

C. WEB CACHING TECHNIQUE

During recent years, there is an accelerated growth in the bandwidth connections and the number of end users. The performance of web servers often degrades in situations like congestion, flash crowds. Web caching combined with pre-fetching techniques are often helpful in reducing the latency and in increasing the overall performance of the network. Clustering web requests in the servers are very useful in the optimization of these techniques. Table.1 shows the parameters of the cache buffer used. The Algorithm.1 details the web caching and pre-fetching [27] algorithm used in this study.

The algorithm consists of three phases based on the location of the requested webpage P_k: First, when P_k is in the cache memory(α_{proxy}), it can be immediately retrieved and the cache-hit counter is increased along with the weight updation of P_k based on its frequency, current key value, and the time difference between old and new requests. Second, when the P_k is found in prefetch buffer($\beta_{prefetch}$), the prefetch-hit counter is increased along with the weight updation of P_k based on its probability and previous key value. Then the page is copied to prefetch buffer from cache memory. Finally, when P_k is not in both prefetch and cache buffer, the page is obtained from δ_{origin} and the P_k with the lowest current key value is replaced with Greedy cache replacement algorithm.

D. AN INTEGRATED WEB CACHING/PREFETCHING SYSTEM

In this work, a web caching/prefetching system is designed and implemented in Python language. Figure.3 depicts the major components of the system. It consists of three main components: the pre-processing dataset, clustering user data, and an integrated cache replacement algorithm. First, weblog records are cleaned and restructured into page visitation and website duration arrays. Different types of intervals such as users' page visit, users' website duration between two users are defined. Also, locally normalized versions of these intervals are also measured. Then the proximity of the users using various distance metrics like Manhattan, Minkowski, and squared Euclidean are analyzed using the k-means clustering. A classifier module is also used to improve the efficiency of the pre-fetch buffer. Finally, the proxy Greedy cache page replacement algorithm is used to perform the page substitution and calculate the page hit/miss performances.

E. CLUSTER EVALUATION

The principal tasks of cluster evaluation are: finding nonrandom patterns in the data, identifying the cluster count, and measuring the quality of the clusters. Typically, the effectiveness of the cluster is determined by four extrinsic measures: homogeneity(pureness of cluster), completeness(same class belongs to the same cluster), rag bag(class of objects cannot be merged), and tiny cluster preservation(slices of tiny class produces more inaccurate results than the slices of larger class). The Silhouette coefficient is one of the cluster evaluation factors in identifying the betterness of the clusters. In this work, the Silhouette measure [28] which is based on cluster compactness and separation is used to identify the optimized cluster count and best user proximity intervals between the clusters. This measure ranges between 1 and -1. The value closer to 1 indicates the compactness and good separability from other clusters.

V. EXPERIMENTS AND RESULTS

The results obtained by the web caching/prefetching system is furnished and analyzed in this section. As various distance metrics are used for clustering purposes, finding the optimized number of clusters and the accurate proximity measure between the users is very important in this work. A total of 200,000 web requests are selected from the dataset. The cache buffer is fixed at 256MB and pre-fetch buffer size is considered to be unlimited for this study. 100 random users(c_n) with unique IDs are generated and assigned to each of the chosen web requests. It is noted that these web requests are mutually exclusive between the users. Let's consider that every website is assumed to have 20 webpages(c_p) each, the number of websites(c_w) is equal to 50 and the number of timeframes(c_t) is 5. The size of the website duration and page visitation arrays created from the dataset is $100 \times 50 \times 20 \times 2$ and $100 \times 50 \times 20$ respectively. To measure an impartial estimate of the proposed distances, the logfile dataset is divided into train/validation/test data split of 60%/20%/20%, Therefore, 60 users are selected for training. The cluster optimization and parameter tuning are only done on training data. After clustering, the pages P_k are loaded to cache and prefetch buffers initially. Then the Greedy page replacement algorithm

Algorithm 1: Web Caching and Pre-Fetching Algorithm

begin
while *web page* P_k *is Requested* **do**
  if *the Requested* P_k *is in the* α_{proxy} **then**
    Retrieve the P_k from the α_{proxy}
    CacheHits = CacheHits + 1
    ByteHits = ByteHits + 1
    Re-calculate the weight of P_k using the cache formula:
    $K_n^{(P_k)} = L_{Inf} + \frac{freq(P_k)+K_n^{(P_k)} \times \left(\frac{\Delta CP_t^{(P_k)}}{\beta_t^{(P_k)} - \chi_t^{(P_k)}} \right)}{S_{P_k}}$
    Update α_{proxy}
  else if *web page* P_k *is in* $\beta_{prefetch}$ **then**
    PrefetchHits = PrefetchHits + 1
    PrefetchByteHits = ByteHits + 1
    Re-calculate the weight of P_k using the Prefetch formula:
    $K_n^{(P_k)} = L_{Inf} + \frac{Prob(P_k)+K_{n-1}^{(P_k)} \times \left(\frac{1}{\beta_t^{(P_k)} - \chi_t^{(P_k)}} \right)}{S_{P_k}}$
    Update $\beta_{prefetch}$
    Copy P_k from the α_{proxy} to $\beta_{prefetch}$ for accessing the same page in the future
  else if *web page* P_k *not in both* α_{proxy} *and* $\beta_{prefetch}$ **then**
    Retrieve the P_k from δ_{origin} to α_{proxy}
    while $S_{P_k} >$ *Avalible size of* $S_{\alpha_{proxy}}$ **do**
      Replace P_k with lowest key value $K_n^{(P_k)}$ using the Greedy Cache Replacement algorithm
      Re-calculate the key value of P_k using the cache formula: $K_n^{(P_k)} = L_{Inf} + \frac{freq(P_k)+K_n^{(P_k)} \times \left(\frac{\Delta CP_t^{(P_k)}}{\beta_t^{(P_k)} - \chi_t^{(P_k)}} \right)}{S_{P_k}}$
      Push P_k from α_{proxy} to $\beta_{prefetch}$
      Re-calculate the weight of P_k using the Prefetch formula:
      $K_n^{(P_k)} = L_{Inf} + \frac{Prob(P_k)+K_{n-1}^{(P_k)} \times \left(\frac{1}{\beta_t^{(P_k)} - \chi_t^{(P_k)}} \right)}{S_{P_k}}$
      Find the $Prob(P_k)$ for future access using the RFT classifier.
end

is used. Classifier model specified in the Algorithm.1 is trained with the validation dataset. This process is repeated 100 times. 5-fold cross validation is used for the clustering process.

A. DETERMINING THE OPTIMAL CLUSTER COUNT

Figs.4,5 shows the cluster evaluation silhouette measures of Users' Page visit interval($\Delta_{U_{wp}}$), Users' website duration interval($\Delta_{U_{wpt}}$), Locally normalized Users' Page visit interval($\overline{\Delta_{U_{wp}}}$), Locally Normalized Users' website duration interval($\overline{\Delta_{U_{wpt}}}$), Locally Normalized Users' Interest Interval(Δ_I) over Manhattan, squared Euclidean, 3-,4-,5-norm Minkowski distances for 2, 6, 10, and 14 clusters. This clustering process is done using training data and fitted on the validation and testing set. The box plots (Figs.4 and 5) represents the silhouette measures of validation data clusters. These graphs are used to identify the optimal cluster count that will be applied for the testing set. The good clusters count is based on the mean silhouette value. The more this value closer to 1, the better the clusters are. For cluster = 2, the mean silhouette values are all negative and ranges between −0.9 to −0.2. All the intervals perform poorly in this cluster. The mean silhouettes in cluster 6 are better than 2 and range between 0.25 to 0.16. But still, 10%-20% of the values are negative which is poor. For cluster = 10, the average range is from 0.57 to 0.25. In this group, the Manhattan and squared Euclidean shows good performances with a maximum of 0.9 and a minimum of 0.23. Minkowski 3-,4-,5- norm distances show an average of 0.3 and 5%-10% of values are negative. For the cluster = 16, the average value for Manhattan and squared Euclidean is in the range of 0.3 to 0.4. And the other distances show poor performances with many negative values. In summary, the good mean silhouettes are obtained for Manhattan and squared Euclidean distances

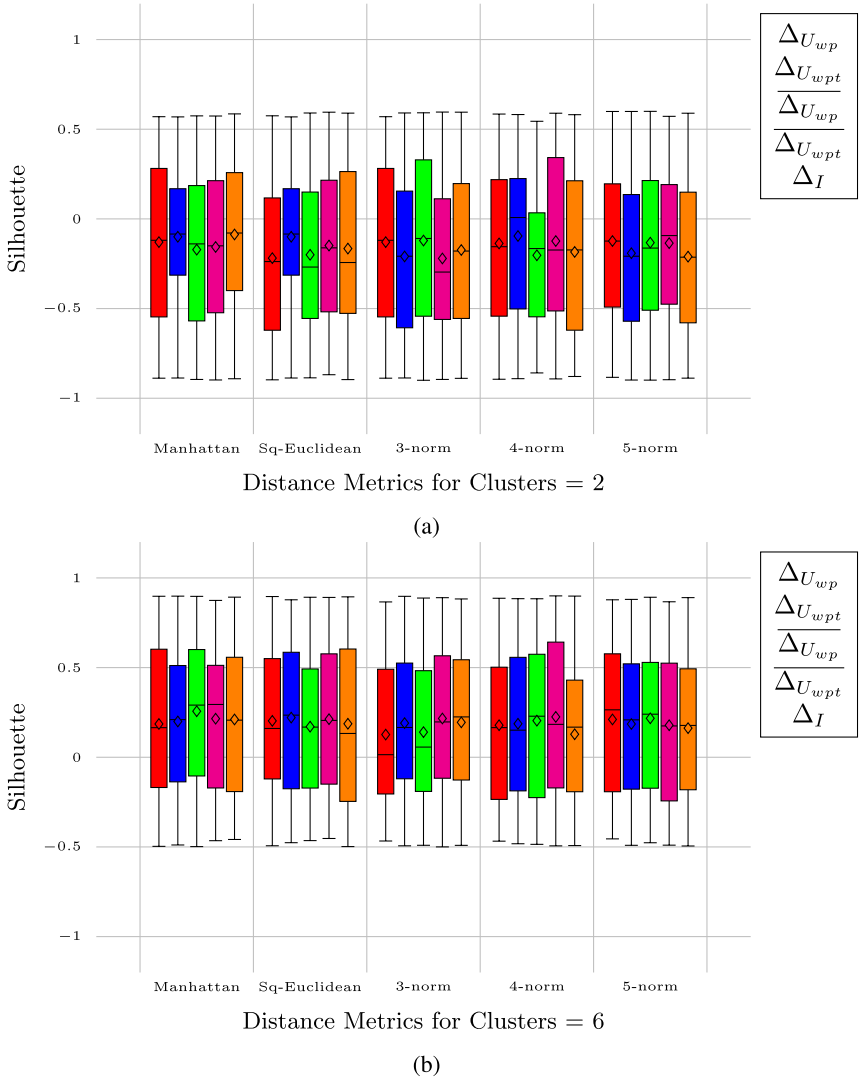


FIGURE 4. Summary of Silhouette measures with different intervals like $\Delta_{U_{wp}}$, $\Delta_{U_{wpt}}$, $\overline{\Delta_{U_{wp}}}$, $\overline{\Delta_{U_{wpt}}}$, Δ_I over various distance metrics with cluster count = 2 and 6.

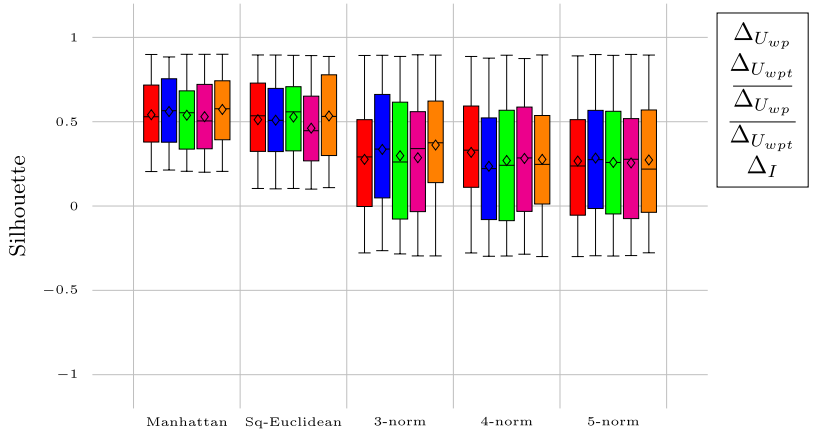
for the cluster count of 10. As there are no high variations in the results between the intervals, It is decided to use the two distances and five intervals with 10 clusters for the next step.

B. CLASSIFIER TRAINING

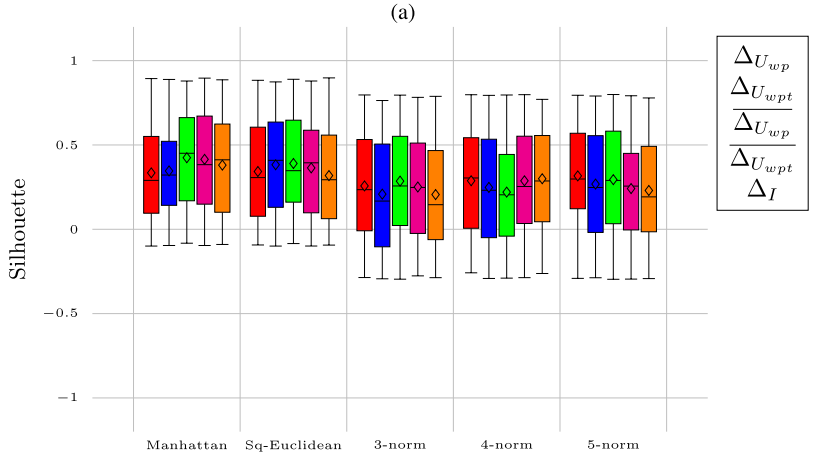
In the Web Caching and Pre-fetching Algorithm.1, it is noted that a classifier is used to improve the performance of the pre-fetch buffer by calculating the probability of the page P_k that will be used in the near future. Random forest(RFT) classifier is used for this purpose. This is a supervised technique that needs correct/wrong classes to work on. In this case, it's a hit or miss. For this purpose, the Algorithm.1 has to be run in two phases: first, to generate the training data for the classifier from the validation data and second, the trained classifier is applied on the test data. After the clustering step, for the generation of training data, the algorithm is applied on validation data without the classifier(consider all the page have equal probability($Prob(P_k)$)), for each request the hit and miss are calculated and stored. The webURLs requests are divided into network address and path. The input fields to the classifier are:

$$\begin{aligned} Input\ features\ -\ &(network\ address,\ path\ size\ of\ webpage) \\ &-\ Target\ class(hit/miss) \end{aligned}$$

The output of the classifier is the posterior probability for the target classes which can be used as $Prob(P_k)$ in the algorithm when applied on the test data in the next phase. Fig.6 illustrates the hit rate of the training data with the classifier. As the clustering step uses the training data for finding the optimal cluster count, usage of the same to train may lead to overfitting. So here, only the validation data is used for classifier training. 10-fold cross-validation is used here. The results clearly show that the Locally Normalized Users' Interest Interval(Δ_I) for both Manhattan and squared Euclidean distances performs better than others. These parameters are selected for the next test data study.



(a)



(b)

FIGURE 5. Summary of Silhouette measures with different intervals like $\Delta_{U_{wp}}$, $\Delta_{U_{wpt}}$, $\overline{\Delta_{U_{wp}}}$, $\overline{\Delta_{U_{wpt}}}$, Δ_I over various distance metrics with cluster count = 10 and 12.

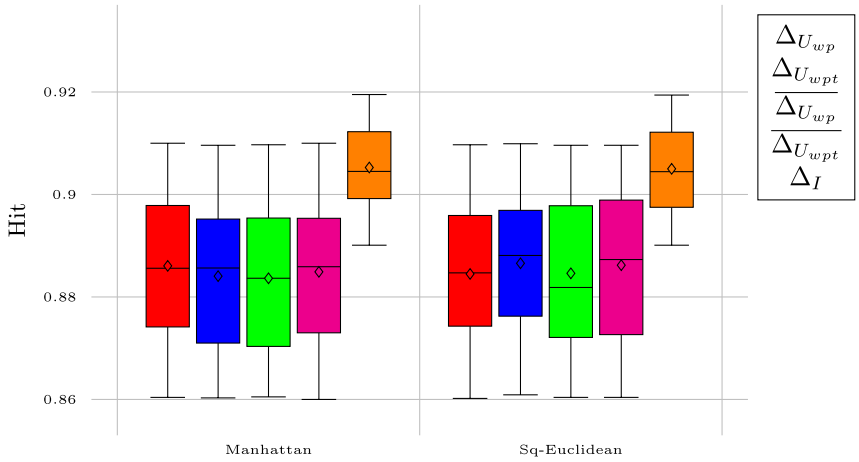


FIGURE 6. Training data hit rate using RFT classifier.

C. RESULTS ON TEST DATA

This is the final stage of the proposed system. The test set is clustered according to the training data and given as input to Algorithm.1. Fig.7 depicts the calculated hit/byte hit rates for replacement algorithms such as least Recently Used(LRU), and least Frequently Used(LFU), Greedy Dual

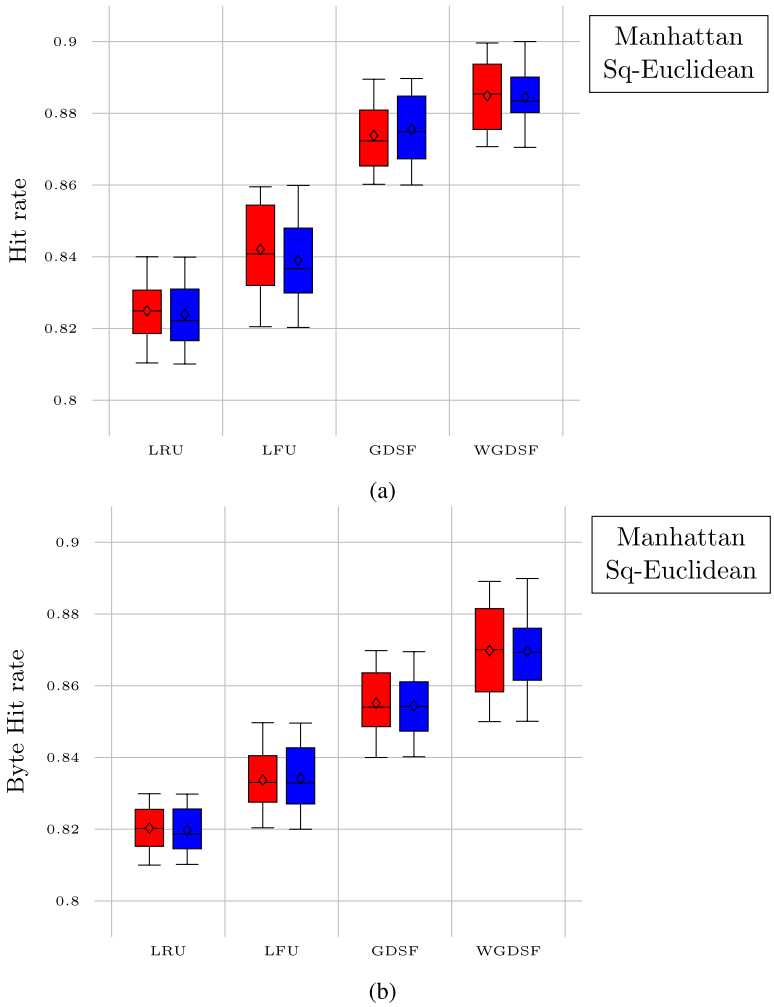


FIGURE 7. Test data accuracy.

Size Frequency(GDSF), and Weighted Greedy Dual Size Frequency(WGDSF) for Manhattan and squared Euclidean distances using the Locally Normalized Users' Interest Intervals(Δ_I). It is noted that the LFU and LRU produces slightly low byte/hit rate than GDSF and WGDSF for relatively large sets of data. Additional factors like the size of the webpage, key values weight updation and classifier play a major role in obtaining good results for a cache size of 128MB. The summary of Hit rates is as follows: LRU(maximum: 0.84, minimum: 0.81, mean: 0.82), LFU(0.86, 0.82, 0.84), GDSF(0.89, 0.86, 0.87), and WGDSF(0.90, 0.87, 0.89). The summary of byte hits is as follows: LRU(maximum: 0.83, minimum: 0.81, mean: 0.82), LFU(0.85, 0.82, 0.83), GDSF(0.87, 0.84, 0.85), and WGDSF(0.89, 0.85, 0.86). The difference between the hit rate performance of two distances is negligible. It indicates that these two distances can be interchangeably used along with greedy web proxy cache algorithms for web caching and pre-fetching to produce good results. In conclusion, the proposed Locally Normalized Users' Interest Intervals can efficiently model the user navigational behaviour to improve web caching and pre-fetching for larger datasets.

VI. CONCLUSION

In this research, a web cache/prefetching system is implemented and tested on a benchmark Boston University computer engineering department dataset. The common web access navigational behaviour of the users is explored thoroughly using five types of distances including Manhattan, squared Euclidean, 3-,4-,5- norms Minkowski metrics, two types of existing user intervals, and three types of customized locally normalized user intervals. The clustering phase infers that the Manhattan and squared Euclidean distances provide better collective information about the common user navigation patterns which can be efficiently used to improve the web caching. The RFT classifier phase in the web cache/pre-fetch algorithm further identifies the Locally Normalized Users' Interest Intervals users' data as the best accuracy feature. The WGDSF method is compared with lightweight methods like LRU and LFU, and produces the best hit rate of 0.89 and the best byte hit rate of 0.86 for the test data. It is proved that Locally Normalized Users' Interest Intervals can be efficiently implemented to model larger datasets. In the future, locally normalized user intervals can be extended to improve caching in information centric networking environment.

REFERENCES

[1] C. Li, M. Song, S. Du, X. Wang, M. Zhang, and Y. Luo, "Adaptive priority-based cache replacement and prediction-based cache prefetching in edge computing environment," *J. Netw. Comput. Appl.*, vol. 165, Sep. 2020, Art. no. 102715.

[2] D. Carra, G. Neglia, and P. Michiardi, "Elastic provisioning of cloud caches: A cost-aware TTL approach," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1283–1296, Jun. 2020.

[3] T. Trinh, D. Wu, and J. Z. Huang, "C3C: A new static content-based three-level web cache," *IEEE Access*, vol. 7, pp. 11796–11808, 2019.

[4] C. Fang, H. Yao, Z. Wang, P. Si, Y. Chen, X. Wang, and F. R. Yu, "Edge cache-based ISP-CP collaboration scheme for content delivery services," *IEEE Access*, vol. 7, pp. 5277–5284, 2019.

[5] X. Fan and J. H. Wang, "Proxy engine for custom handling of web content," U.S. Patent 9 906 549, Feb. 27, 2018.

[6] J. A. Toebes, "Presenter device as web proxy for collaborative sharing of web content having presenter context," U.S. Patent 9 781 175, Oct. 3, 2017.

[7] V. R. Chandaka, M. Smacinih, S. Guha, R. Mahalati, A. J. Sanghvi, and V. A. Mushkatin, "Proxy-based web application monitoring through script instrumentation," U.S. Patent 9 654 580, May 16, 2017.

[8] C. S. Joel, J. T. W. Benterou, L. H. Holloway, M. B. Prince, and I. G. Pye, "Loading of web resources," U.S. Patent 9 342 620, May 17, 2016.

[9] R. J. Cohen, A. L. Bolgert, R. M. Forlenza, M. Sang, and K. K. Yellepeddy, "Auto-detection of web-based application characteristics for reverse proxy enablement," U.S. Patent 14 487 414, Mar. 17, 2016.

[10] S. Reddy, S. Togaru, and C. Trillo, "Reducing latencies in web page rendering," U.S. Patent 9 158 845, Oct. 13, 2015.

[11] W. Chao, "Web cache intelligent replacement strategy combined with GDSF and SVM network re-accessed probability prediction," *J. Ambient Intell. Hum. Comput.*, vol. 11, no. 2, pp. 581–587, Oct. 2018.

[12] K. R. Baskaran and C. Kalaiarasan, "Improved performance by combining web pre-fetching using clustering with web caching based on SVM learning method," *Int. J. Comput. Commun. Control*, vol. 11, no. 2, pp. 67–178, 2016.

[13] R. F. Olanrewaju and A. W. Azman, "Enhancement web proxy cache performance using wrapper feature selection methods with NB and J48," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 260, 2017, Art. no. 012012.

[14] A. Abdalla, S. Sulaiman, and W. Ali, "Intelligent web objects prediction approach in web proxy cache using supervised machine learning and feature selection," *Int. J. Adv. Soft Comput. Appl*, vol. 7, no. 3, pp. 1–19, 2015.

[15] H. Ibrahim, W. Yasin, N. I. Udzir, A. Hamid, and N. A. Wati, "Intelligent cooperative web caching policies for media objects based on J48 decision tree and Naïve Bayes supervised machine learning algorithms in structured peer-to-peer systems," *J. Inf. Commun. Technol.*, vol. 15, no. 2, pp. 85–116, 2016.

[16] P. J. Benadit, F. S. Francis, and U. Muruganantham, "Improving the performance of a proxy cache using tree augmented Naive Bayes classifier," *Proc. Comput. Sci.*, vol. 46, pp. 184–193, Jan. 2015.

[17] J. B. Pernabas, S. F. Fidele, and K. K. Vaithinathan, "Enhancing greedy web proxy caching using weighted random indexing based data mining classifier," *Egyptian Informat. J.*, vol. 20, no. 2, pp. 117–130, Jul. 2019.

[18] M. Tinghuai, J. Qu, W. Shen, Y. Tian, A. Al-Dhelaan, and M. Al-Rodhaan, "Weighted greedy dual size frequency based caching replacement algorithm," *IEEE Access*, vol. 6, pp. 7214–7223, 2018.

[19] T. Ma, Y. Hao, W. Shen, Y. Tian, and M. Al-Rodhaan, "An improved web cache replacement algorithm based on weighting and cost," *IEEE Access*, vol. 6, pp. 27010–27017, 2018.

[20] S. G. Petridou, V. A. Koutsonikola, A. I. Vakali, and G. I. Papadimitriou, "Time-aware web users' clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 5, pp. 653–667, May 2008.

[21] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999.

[22] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY, USA: Springer, 2009.

[23] Y. Zhao, G. Karypis, and D.-Z. Du, *Criterion Functions for Document Clustering*. Minneapolis, MN, USA: Univ. of Minnesota, 2005.

[24] A. Bianco, G. Mardente, M. Mellia, M. Munafo, and L. Muscariello, "Web user session characterization via clustering techniques," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, vol. 2, Nov./Dec. 2005, p. 6.

[25] A. Singh, A. Yadav, and A. Rana, "K-means with three different distance metrics," *Int. J. Comput. Appl.*, vol. 67, no. 10, pp. 13–17, Apr. 2013.

[26] C. A. Cunha, A. Bestavros, and M. E. Crovella, "Characteristics of WWW client traces," Dept. Comput. Sci., Boston Univ., Boston, MA, USA, Tech. Rep. TR-95, 1995, vol. 10.

[27] Y.-F. Huang and J.-M. Hsu, "Mining web logs to improve hit ratios of prefetching and caching," *Knowl.-Based Syst.*, vol. 21, no. 1, pp. 62–69, Feb. 2008.

[28] J. Han, M. Kamber, and J. Pei, "10-cluster analysis: Basic concepts and methods," in *Data Mining*. San Mateo, CA, USA: Morgan Kaufmann, 2012, pp. 443–495.