Egyptian Informatics Journal 20 (2019) 117–130

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Enhancing Greedy Web Proxy caching using Weighted Random Indexing based Data Mining Classifier

Julian Benadit Pernabas [a,*], Sagayraj Francis Fidele [b], Krishna Kumar Vaithinathan [c]

[a] Department of Computer Science and Engineering,Faculty of Engineering, CHRIST(Deemed To be University), Kengeri Campus, Kanmanike,Bangalore,560074, Karnataka,India.
[b] Department of Computer Science and Engineering, Pondicherry Engineering College, ECR, Pillaichavady, Pondicherry 605014, India
[c] Department of Computer Engineering, Karaikal Polytechnic College, Varichikudy, Karaikal-609609, India

ARTICLE INFO

Article history:
Received 8 March 2017
Revised 7 December 2018
Accepted 7 January 2019
Available online 14 January 2019

Keywords:
GDS
GDSF
GD*
Random indexing
Clustering
Proxy
Data Mining

ABSTRACT

Web Proxy caching system is an intermediary between the Web users and servers that try to alleviate the loads on the origin servers by caching particular Web objects and behaves as the proxy for the server and services the requests that are made to the servers. In this paper, the performance of a Proxy system is measured by the number of hits at the Proxy. Higher number of hits at the Proxy server reflects the effectiveness of the Proxy system. The number of hits is determined by the replacement policies chosen by the Proxy systems. Traditional replacement policies that are based on time and size are reactive and do not consider the events that will possibly happen in the future. The performance of the web proxy caching system is improved by adapting Data Mining Classifier model based on Web User clustering and Weighted Random Indexing Methods. The outcome of the paper are proactive strategies that augment the traditional replacement policies such as GDS, GDSF, GD* which uses the Data Mining techniques.

1. Introduction

The World Wide Web and its usage are growing at a rapid rate and this has resulted in overloaded Web servers, network congestion, and consequently poor response time. Multitudes of approaches are continuously being made to overcome these challenges. 'Web caching' is one of the approaches that can enhance the performance of the Web [1]. A Web cache is a buffered repository of the Web objects that are most likely to be requested frequently and in the near future. The general architecture of the World Wide Web is shown below in Fig. 1. consists of the client users, the Proxy Server, and the Origin Server. Whenever the client requests the Web object, it can be retrieved from the Proxy server immediately, or it can be retrieved from the origin server. Therefore, whenever a user's request is satisfied from the Proxy server, it minimizes the response time as well as it reduces the overload of the Web origin server. Typically, the Web cache may be located [1] at the origin server cache, at the Proxy server cache, or the client cache.

The overall objective of the research work is to improve the performance of the Greedy Web Proxy caching algorithms by augmenting the Web user clustering and Data Mining Classifier model based on the random indexing methods and Weight assignment policy mechanism. Sections 2 address the work in Traditional Greedy Web Proxy Caching algorithms and Data Mining based Web caching Methods. Section 3 details the Overall working model for the Web Proxy caching system. Section 4 presents the Performance Evaluation Used in Clustering and Classification

Section 5 presents the Generic Model for Web Proxy Caching algorithm using Data Mining Classifier Model, Section 6 elaborates the Performance measures of web proxy cache replacement algorithms.

2. Related work

The methodologies for Web caching may be categorized into two groups. The first category of methods is *Traditional Greedy* in the sense they use computationally simple parameters for cache

* Corresponding author.
E-mail addresses: benaditjulian@gmail.com (J.B. Pernabas), fsfrancis@pec.edu (S.F. Fidele), vkichu77@gmail.com (K.K. Vaithinathan).
Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.eij.2019.01.001
1110-8665/

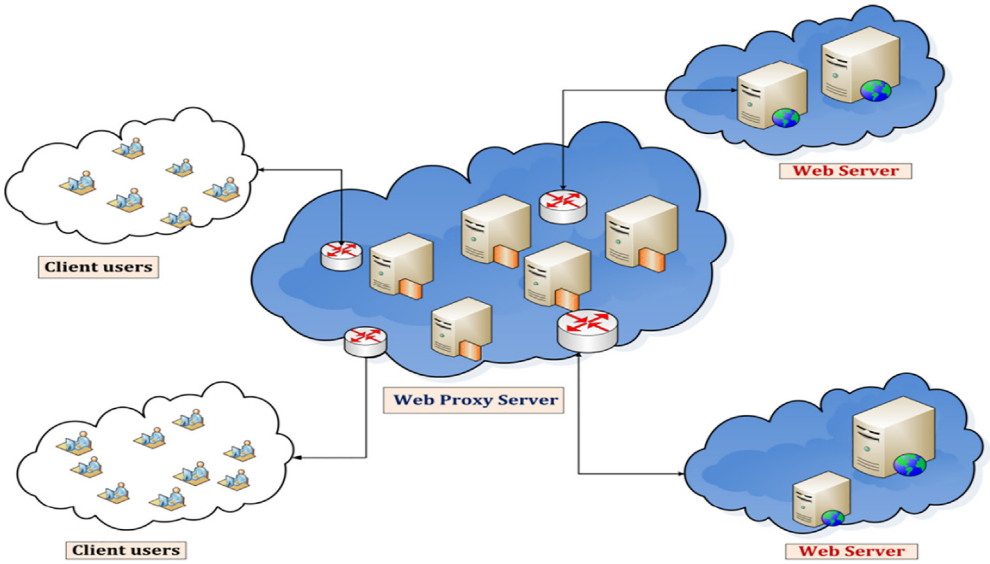

Fig. 1. World Wide Web architecture.

replacement. The second category of the methods combines Data Mining techniques with the traditional approaches for the enhancement of the Web caching system.

2.1. Summary of traditional Web Proxy caching algorithms

This section summaries the traditional Greedy Web Proxy caching algorithm [2,3] based on the key parameters and Table 1.Summaries the Key factors for the eviction and its limitations for the traditional replacement policies like Greedy-Dual Size (GDS), Greedy-Dual Size Frequency(GDSF), and Greedy Dual*(GD*) [3].

2.2. Data Mining based Web caching methods

Several policies have been proposed in the literature for augmenting the traditional Web caching methods with Data Mining techniques. In this Section, the Data Mining based Web caching Methods are broadly categorized as follows:

2.2.1. Regression based Web caching methods

Logistic Regression (LR) models [4] was applied in an adaptive Web cache, where the model predicts the Web objects that may be requested in the future and the LR policy replaces the Web object with the lowest re-visited value. Multinomial Logistic Regression (MLR) technique is also used as an early content classification scheme for Web cache objects [5] and is simple to implement, but results in low classification accuracy with reduced cache performance. A semi-intelligent web cache system with a lightweight machine learning technique was used for Proxy cache replacement [6]. It uses a novel caching scheme called east Worthy Evict–Least Recently Used (LWE-LRU) for cache replacement.

2.2.2. Classification based Web caching methods

KORA (Khalid Obaidat Replacement Algorithm) to enhance the Web cache performance [7] uses a neural network to identify transient and shadow cache lines. The temporary lines are the new web object arrivals to the cache, whereas the shadow lines are the arrivals of the Web object that were replaced recently from the cache in favor of some other lines. The KORA algorithm performs well compared to the conventional algorithm by having lower miss ratio. An adaptive web cache access predictor technique [8] uses Back-Propagation Neural Network (BPNN) to improve the performance of Web caching by predicting the most likely re-accessed objects. A non-linear model using object features [9] analyzes the Web cache optimization with Multilayer Perceptron (MLP) Network and predicts the value of the object based on the syntactic features of the HTML document. The Neural Network Proxy Cache Replacement (NNPCR) [10], uses the Back Propagation to adjust the weight factors in the network. Here, an object is selected for replacement based on the ratings returned by the Back Propagation Neural Network (BPNN).

Table 1
Summary of traditional Web Proxy caching algorithms.

S.no	Algorithms	Parameters	Evictions	Limitations
1.	GDS	Object size S_p. Object cost C_p. Inflation –Value L.	Least valuable objects with a Key value $K_p = L + C_p/S_p$.	It does not consider the frequency value of the Web object.
2.	GDSF	Object size S_p. Object cost C_p. Inflation -Value$L$. Number of non-aged references F_p.	Least valuable objects with a Key value. $K_p = L + C_p \times F_p/S_p$	It does not consider the inter-access time of the Web object.
3.	GD*	Object size S_p. Object cost C_p. Inflation-Value L. Temporal correlation measures β. Number of non-aged references F_p.	Least valuable objects with a Key value. $K_p = L + (C_p \times F_p/S_p)^{\frac{1}{\beta}}$.	Low hit ratio under constant cost model.

Fuzzy rules [11] were also used to identify the Web pages to be removed from the cache. The variables describing each web page are first *fuzzified*, and the output is *de-fuzzified* to replace the Web page. Artificial Neural Network algorithm (ANN) combined with PSO [12] was used for improving the neural network performance. The neuro-fuzzy system is partitioned client-side Web Cache which uses the Adaptive Neuro-Fuzzy Inference System (ANFIS) method for binary classification and for classifying the Web objects into cacheable or uncacheable objects. Here, the trained Neuro-fuzzy has been employed with the LRU algorithm in cache replacement decision. The Intelligent Naïve-Bayes approach [13] were also used to identify whether the Web object can be re-accessed in the future or not. In [14,15,16] various classification algorithm improves the performance of a proxy cache using Tree Augmented Naïve Bayes approach followed by very fast decision tree algorithm and Expectation Maximization with Naive Bayes Classifier for improving the Web proxy cache using sliding window mechanism and Data Mining classification performance. This method is integrated with greedy replacement algorithms like GDS, GDSF, and GD* which consider multiple factors like cost, size, frequency to form a novel Web caching.

2.2.3. Clustering based Web caching methods

A Cluster-based prefetching scheme on a Web cache is used to partition the clusters of *correlated* Web pages to identify the user access pattern [17]. In this method, when the user requests a Web object, the proxy server retrieves all the objects, which are in the same cluster with the requested object. In this approach, the proxy residues are represented by using the navigational graph algorithm. Moreover, this method has some limitations on the estimation of the number of clusters due to the complex nature resulting in low performance.

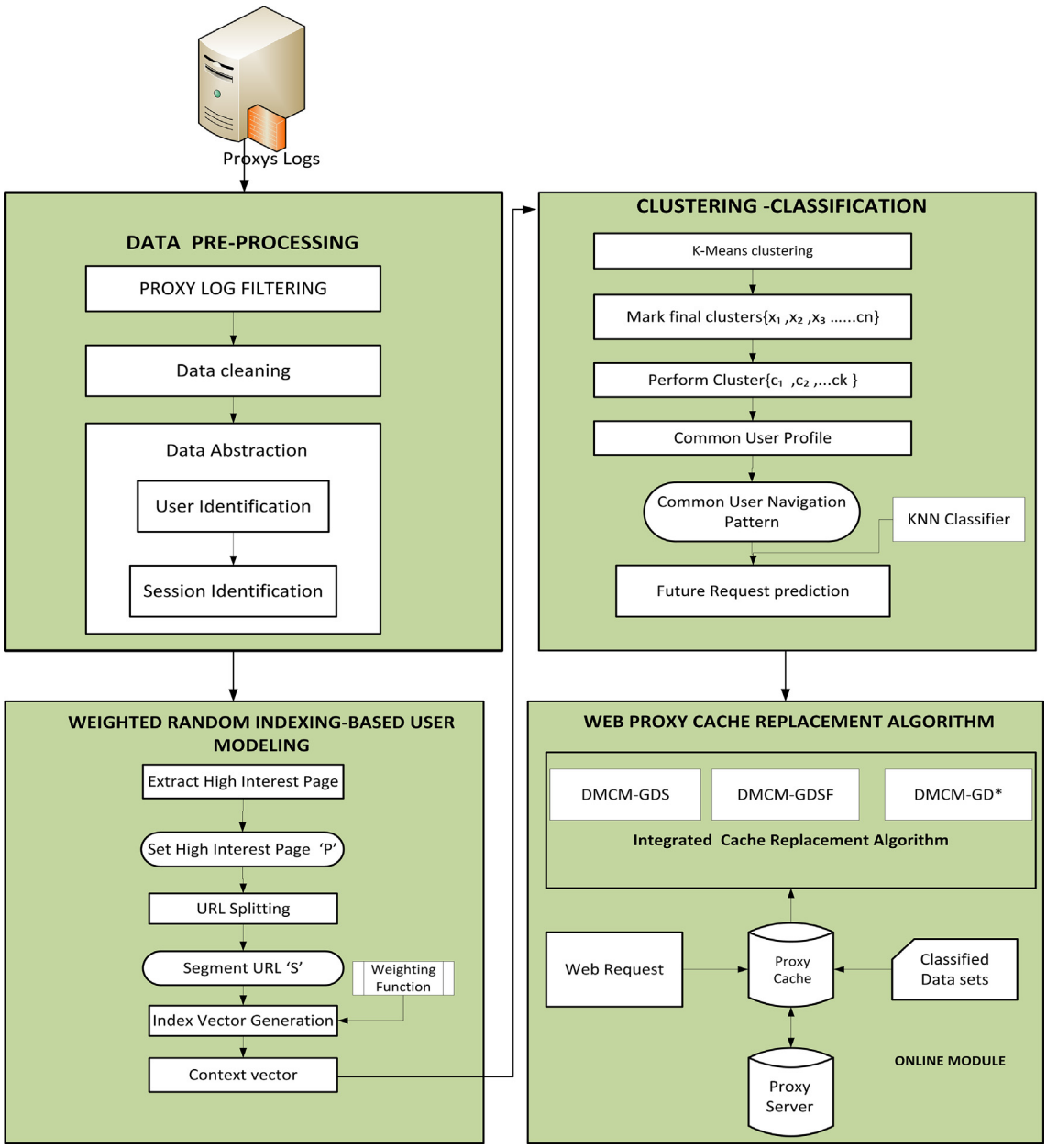

Fig. 2. Overall working model for the Web Proxy caching based on Data Mining Classifier Model.

2.2.4. Evolutionary based Web caching methods

An Evolutionary technique for Web caching applies the Genetic Algorithm (GA) for cache replacement decision [18]. Here, each Web object's "strength" is determined by the specific properties related to object's staleness, access frequency, and retrieval cost. The Finite State Machines (FSM) is coupled with an evolutionary algorithm shows a good prediction rate for web prediction [19]. GA-Based cache replacement policy has also been used for user side cache replacement [20]. In this process, the algorithm uses a fitness function, and the replacement policy considers the download time and number of references to a Web object along with its size.

3. Working principle of Web Proxy caching based on Data Mining Classifier Model

Clustering is one of the widely used data mining technique that is used in learning systems. The strategy presented in this chapter augments the traditional Web Proxy caching by clustering web user and amalgamating it with classifier model based on Weighted Random Indexing and Weight assignment policy. The overall working flow model consists of different phases as shown in Fig. 2. These working methods are classified as given below:

3.1. Web Proxy server datasets collection

In this section, the Web Log files for the Web Proxy cache simulation are obtained from the National Lab for Applied Network Research (NLANR) [21]. Each data set represents a proxy server located in a particular location.

3.2. Data Pre-Processing

In this method, the datasets are pre-processed and its converted into a structured format for reducing the time of simulation. The

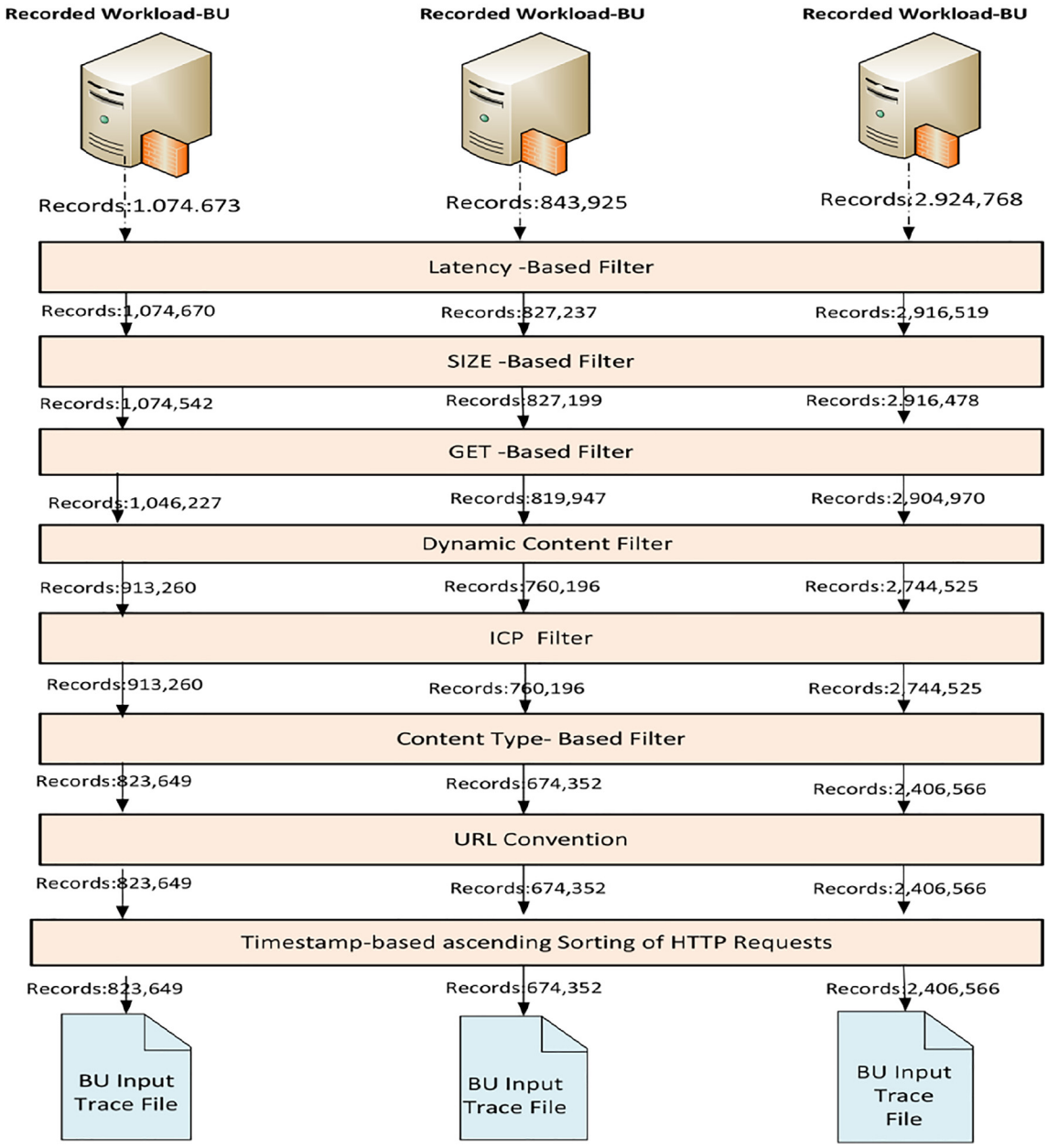

Fig. 3. Filtering phases for the Web Proxy datasets.

techniques undergo few pre-processing [22] steps to remove irrelevant requests, and to extract useful information. The steps involved in data pre-processing are Proxy Logs Filtration, Data cleaning, and Data Abstraction.

3.2.1. Proxy logs filtration

In this technique, the recorded proxy log files obtained from the NLANR have undergone the basic filtration process to reduce the size of the log data sets as well as the running time of the simulation. Thus, only the three proxies log datasets are considered for the filtration process [23] as shown in Fig. 3. This filtration module is based on the methods such as a Latency based method, SIZE based method, Dynamic based method, Content-based method. The Fig. 3. Illustrates the steps involved in various filtering methods [23] and its details are described below. After these filtering, the size of the proxy log files is reduced to have some unique requests, which have obtained after the filtering phases, i.e., 823649 for the UC datasets, 674352 NY datasets, and 2406556 SD datasets as shown in Table 2.

From the filtration results obtained, the URL convention and the required HTTP requests are sorted based on the Timestamp. Now the input trace file from UC, NY, SD is ready to undergo a certain data pre-processing steps for the simulation of the Data Mining Classifier.

3.2.2. Data cleaning

Data cleaning is the process of removing the irrelevant entries in the proxy log file [22]. Here only the relevant HTML file is considered and all other irrelevant log entries that were recorded by requesting graphics, sound, and other multimedia files, etc. are discarded.

3.3. Data Abstraction

Data Abstraction is the process of abstracting the log entries based on the User and Session Identification.

Table 2
Statistics for the trace file.

Trace File	Unique Requests	Unique Servers	Mean Object Size (bytes)	Median Object Size (bytes)
BOSTON	125,505	2580	13,280	2070
SILICON	935,630	27,674	5,822	2238
UC	823,649	47,210	41,651	2814
NY	674,352	59,139	17,440	209
SD	2,406,556	77,092	53,312	652

/* *Algorithm for User Identification* */
Input : Web Proxy logs
output : $U = \{u_1, u_2, u_3 ..... u_k\}$

1. ***begin***
2. *Initially* the datasets are sorted based on the timestamp and by the IP address.
3. ***begin***
4. *for* each IP address identification, each user agent is pertained as a different user.
5. *for* each user identified in step *2*, apply path information collected and determine whether this behaviour is more likely the result of two or more users.
6. ***end;***
7. Finally each user, are identified from step *2 to step 5* with available cookie and registration information.
8. ***end;***

Fig. 4. Algorithm for User Identification.

/* *Algorithm for identifying Session Identification* */
Input : Web Proxy Logs (t_i time stamp of the 'i^{th}' record belonging to a user in the proxy logs)
Output : Set of user session $S = \{s_1, s_2, ...s_n\}$

1. ***begin***
2. *for* each user u_k identified, assign a session ID.
3. ***begin***
4. *Set* the maximum time limit $\Delta t = 30 \min s$.
5. *for* each user u_k, repeat step 5 to step 7 recursively.
6. Calculate the time difference between the web page requests i.e. $t_{i+1} - t_i$
7. *if* the difference in time is greater than the maximum limit i.e. $t_{i+1} - t_i \le \Delta t$,create a new session ID.
8. Finally sort the entries based on the session ID.
9. ***end;***
10. ***end;***

Fig. 5. Algorithm for Session Identification.

3.3.1. User identification

The goal of user identification [24] is to identify each user in the Proxy datasets. This method can be carried out easily if the user provides his or her registration information. There are several ways to identify the individual users in the Proxy datasets, which are collected from proxy servers. The algorithm for identifying user identification is given in Fig. 4.

3.3.2. Session identification

The goal of session identification is to divide each user into different segments based on the user's pattern, which is called a session. Session identification approaches identify the user's session by a maximum time limit. A new session is created, when the difference in time of a Web page is more significant than the threshold limit of access time, or if the time spent in the same session exceeds the maximum limit. Based on the empirical findings, the maximum time limit has been set to be 30 min [24] for our simulation. The algorithm for session identification is shown in Fig. 5.

After the proxy datasets are pre-processed; it is further analyzed to obtain the common user features for Web user clustering.

3.4. User modeling based on Weighted Random Indexing

The user modeling based on the Weighted Random Indexing consists of the following methods such as High-Interest Page Set, Segmentation of URL, and Random Indexing based on weight, Weight Function, and Navigation set of users.

3.4.1. High-Interest page set

Once the dataset has been preprocessed, it is segmented to find out the high-interest Web object, based on these two measures, i.e., Frequency and Duration [25]. Let P be the set of a Web object $P = \{p_1, p_2, p_3 \ldots p_n\}$ accessed by the users in the Proxy server logs. Here the session is transformed into 'n' a dimensional vector for the weight is assigned as $S = \{w(p_1, s), w(p_2, s) \ldots w(p_n, s)\}$ where, $w(p_i, s)$ is a weight assigned to i^{th} Web page p visited in a Session S. The parameters involved in obtaining the High-Interest Web object [25] is shown in Table 3. The frequency of the Web object p is calculated by the number of times the Web object p is accessed.

Table 3
Parameters for High-Interest Web Object.

Symbols	Parameters for High-Interest Web Object
p	Web object.
α	Frequency of the Web object p.
β	Duration of the Web object p.
$Size$	Size of the Web object p.
λ	High-Interest of the Web object p.

The formula for the *Frequency* [25] of the Web object p is given in the Eq. (1)

$$\alpha(p) = \frac{Number\ of\ times(p)\ accessed}{\sum_{WebPage \in\ Visited\ Web\ Pages}(Number\ of\ times(p)\ accessed} \tag{1}$$

Similarly, the *Duration* [25] of the Web object p is given by the Eq. (2) as shown below:

$$\beta(p) = \frac{TotalDuration\ of\ (p)\ /Size\ (p)}{\sum_{Max\ WebPage \in\ Visited\ Web\ Pages} TotalDuration\ (p)/Size(p)} \tag{2}$$

From, these two measures, we can obtain the High-Interest Web object set and this high-interest object set value is normalized to 0 or 1 based on the Eq. (3)given below:

$$\chi(p) = \frac{2 \times \alpha(p) \times \beta(p)}{\alpha(p) + \beta(p)} \tag{3}$$

Once the high-interest Web object p is obtained from the URL,$P = \{URL_1, URL_2, \ldots URL_m\}$ a navigation pattern profile is generated from a set of the individual user, which is given by $U = \{U_1, U_2, \ldots U_n\}$ and these given URLs are segmented based on the user interest Object set p.

3.4.2. Segmentation of URL

The high-interest Web page p obtained from the URL is segmented based on the hierarchical structure of the Websites. The URL is composed of different levels, which are segmented by '/'based on the sequence obtained in the Proxy log.

3.4.3. Weighted Random Indexing

Random Indexing performs well with a small volume of data, but when the volume of data increases, the performance degrades. So to overcome this limitation, the random indexing method is modified with the weight function [26] to improve its performance. In Random Indexing Weights are generated using the statistical information based on the frequency of each term and its contexts. In this case, each index vector in the context is multiplied by the context vector generated. Therefore, the context vector of each term t is calculated as shown in Eq. (4)

$$C_t = \sum_{(r,t') \in (t,*,*)} R(r,t').weight(t,r,t') \tag{4}$$

where,

$R(r,t')$ is the index vector of the context(r,t')
Weight(t,r,t') is the weight function for the term t' and its context.

```
/* Algorithm K-Means (k, Z)*/
1. begin
2.   Initially the data points k are generated as the centroid.
3.     begin
4.       repeat
5.         for each data point y ∈ Z do
6.           Calculate the distance from y to each centroid.
7.           assign y to the closest centroid
8.         end for;
9.       end repeat; /* Re-compute the centroid using the current cluster until the
           condition is met. */
10.    end;
11. end;
```

Fig. 6. Algorithm for K-Means clustering.

In this caching process, the list of web documents are obtained and in each web document, a context window size of 2 + 2 is used for it's constituent words. Using the formula, the weight for each word is generated. This weight is used to generate the index vector. Once generated, the weight function is applied to it to determine the context vector and this is applied iteratively for each word in the document. For example, *Christuniversity.in/studentlogin.html/CS433*, we can segment the URL using '/' to obtain the different sections of the webpage. Using these various segmented sections, the index vectors can be used to determine personalized high interest pages for each user, where a navigation path can be explained in a $n \times d$ dimensional matrix.

3.5. Web user clustering

The main objective Web user clustering [26,27] is creating the common patterns and grouping of user access behavior. Thus, the clustering system would group the Web documents and organize them according to their user interests. Once the random indexing process is generated, an efficient clustering algorithm called the K-Means algorithm is used. It is one of the most popular partitions based clustering algorithms, which is widely used.

K-Means is one of the well-known clustering algorithms from unsupervised learning.The K-Means algorithm partitions the given data into k clusters. Each cluster has a cluster center, called the cluster centroid. The centroid usually used to represent the mean of all the data points in the cluster, which gives the name of the algorithm, i.e., since there are K clusters, thus it is called K- means. The algorithm for K-Means clustering is given in Fig. 6.

3.6. Data Mining classification

K-Nearest Neighbor Classifier [28] is one of the supervised machine learning algorithms used in a variety of application. The K-Nearest Neighbor classifier is working under the principle of distance measures. The KNN algorithm trains not only the data set but also the classification for each training example. This indicates that in the KNN algorithm the training samples are used to build classification models. In KNN classifier the learning process occurs when a test sample needs to be classified. The algorithm for the KNN Classifier is given in Fig. 7

4. Performance evaluation used in clustering and classification

Precision and Recall [29] are the performances suited in many machine learning applications. These two parameters can be measured based on the result obtained from the confusion matrix, which is shown in Table 4. This confusion matrix has been modified according to the correctly or un-correctly classified Web pages which are classified by the classifier. This performance metrics which is shown in Table 5 is mainly used to find out the optimal Data Mining Classifier Model for enhancement of the Web Proxy caching strategies. The following measures are classified as follows:

Table 4
Confusion Matrix.

Actual (classes)	Predicted (classifiers)	
	Actual Positive	Actual Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual negative	False Positive (FP)	True Negative (TN)

Table 5
Performance Metrics used in Data Mining Classifier.

Metric	Description	Formula
Precision (p)	It is defined as the ratio of the number of relevant Web pages that were retrieved and the total number of retrieved Web pages	$Precision(p) = \frac{TP}{TP+FP}$
Recall (r)	It is defined as the ratio between the number of relevant Web pages retrieved and the total number of relevant Web pages.	$Recall(r) = \frac{TP}{TP+FN}$
Correct Classification Rate (CCR)	*Correct Classification Ratio*(CCR) is a good measure for evaluating Classifier .	$CCR = \frac{TP+TN}{TP+TN+FP+FN}$

Table 6
Statistics of Data preprocessing datasets.

S. No	Attributes	Generated Values
1.	Total entries accessed by the users.	1,248,675.
2.	Number of filtered access entries.	116,200
3.	Number of different access users.	12,931.
4	Maximum no of users (Considered for simulation).	100.
5.	User –ID	1 to 100.
6.	Session Duration Time limit (S_i).	30 min.
7.	Duration of the Web page.	10 min.
8.	Minimum Frequency access of the Web page (F_i)	10 times.
9.	Total number of accessed Web pages.	800.
10.	Number of the Web page accessed with Frequency $\geqslant$10 times.	600.
11.	Total Number of Identified sessions.	24,000.
12.	Number of Identified sessions with Session duration (30 min).	12,000.
13.	Number of Identified sessions with Page duration (10 min).	12,500.

4.1. Experimental results and discussion

The experimental results in this section are classified as follows:

4.1.1. Data preprocessing results

In the pre-processing procedure, the original Web proxy datasets are cleaned, formatted, and grouped into user's sessions. This experimental simulation is carried out by the Web Utilization Miner tool (WUM) [30].Table 6 presents some statistics of the experimental data sets. From Table 6, it is shown that 116,200

/*Algorithm K-NN (D, d, k) */
1. ***begin***
2. *Compute* the distance between d and every example in D
   Choose the k examples in D that are nearest to d, denote the set by $(P \subseteq D)$.
3. Assign d the class that is the most frequent class in P
4. ***end:***

Fig. 7. KNN classifier algorithm.

clean entries are extracted. In this, 800 Web pages were visited, and 600 were accessed more than ten times. Also, the total session identified as 24,000, from which the identified session reduced 12,000 with session duration threshold to limit as 30 min. Similarly, the page duration of the session is identified as 12,500 with page duration threshold limit set 10 min.

4.1.2. Clustering result

The parameters used in the Weighted Random Indexing experiments have some values assigned, i.e., +1, −1 in index vector as ε and the context window size are denoted as $l.d$ as a dimension of the index vector. So here,ε is set to$\varepsilon = 10$ as proposed [26] and $l = 1$ as the URLs is short k value for the K-Means clustering. The maximum value of the cluster number k_{max} and be chosen as $\sqrt{n}$ (n is the data size).From the above Table 7, the values are assigned to the parameters, and the experiments were conducted using the simulation tool Text Matrix Generator tool (TMG) [31]. Therefore, after data preprocessing and the Random Indexing for the preprocessed Proxy datasets, a High-Interest page set P is obtained containing 97 requested URLs Web pages. These URLs are split by '/' to get the segment set 'S', which comprises 152 different segments. Similarly, the dimension of the index vector in RI is selected to 300. Finally, 100×300 the matrix is constructed $X = \{X_1, X_2, \ldots X_{100}\}$ for the single user pattern matrix based on the Weighted-RI method, and it takes it as input to the K-means clustering algorithm.

The K-Means clustering algorithm generates the common user's requests and groups these 100 users in the different Clusters Number (C-N) (Varying 1 to 9) based on the Weighted Random Indexing methods. After obtaining the common user's requests, the individual clusters are evaluated based on the performance metrics as shown in Table 8.

From this Table 8, the no of Pre-URL required for each user is identified, and a common navigation pattern profile will be created for each cluster. The parameters Precision and Recall evaluates the performances of clusters. According to the clustered results obtained, the precision is defined as the ratio of hits to the number of URLs that are cached. Similarly, recall is the ratio of hits to the number of URLs that are requested. Fig. 8 signifies the clustering results based on the performance metrics.

Table 7
Parameters for Weighted Random Indexing methods.

Parameters	Values
l	1
d	(3 0 0)
ε	10
k	2 to 10

Table 8
Results of Performance Metrics used in Cluster.

S. No	No. of Clusters	Pre-URLs	Avg. Precision %	Avg. Recall %
1	1	8	83.9	58.4
2	2	10	93.3	45.9
3	3	3	78.8	28.1
4	4	14	62.5	51.8
5	5	3	91.7	60.3
6	6	3	63.6	24.9
7	7	5	82.5	43.1
8	8	9	100	45.3
9	9	9	69.0	63.0

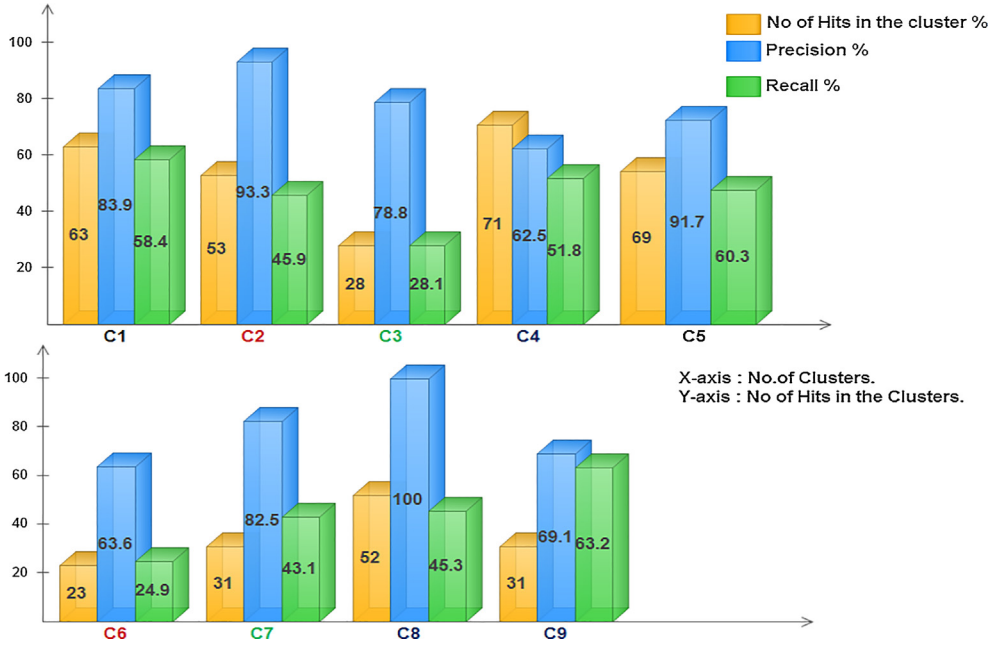

Fig. 8. Results of Precision Recall, and Hits in clusters.

4.1.3. Classifier results

The Data Mining classification (KNN), which aims to evaluate the experimental results of classification [25]. The evaluation of the classification is about whether the Web page belongs to the class of common user navigation pattern profile. This navigation pattern profile is generated based on the session duration, high-interest page, the frequency of access, duration of the Web page. The different classifier trains these values whether the requested Web page belongs to a common user navigation pattern profile. So, if the requested Web page belongs to the common user profile pattern it assigned as class one otherwise zero. Therefore, if the class value of the Web page is 1, it indicates that Web page may request in future and this type of Web page is considered cacheable requests. In this section, different classifiers [25] like Support Vector Machines (SVM), Decision Tree (J48), Naïve Bayes (NB) classifier and K-Nearest classifier (KNN), for classification purpose. So among the different classifier, the KNN classifiers has a better classification accuracy which is shown in Fig. 9

5. Generic Model for Web Proxy caching algorithm using Data Mining Classifier Model

In this Section, a generic model for Web Proxy caching strategies integrated with Data mining Classifier Model was introduced. The algorithm for Generic Web Proxy caching algorithm integrated

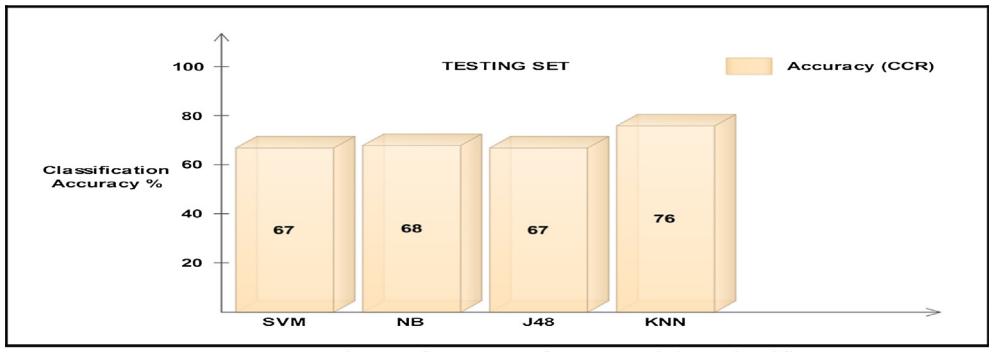

Fig. 9. Comparisons of Accuracy for Data Mining Classifier.

```
Procedure Data Mining Classifier Model ((DMCM))
Proxy cache entry t , t _fresh; int Hits = 0, Byte.Hits = 0;
int Cache Max_Size N
1. begin
2.    DMCM.Build ( );
3.    begin
4.      loop forever
5.      begin
6.        do
7.          Get request the Web object p from the Proxy Cache t .
8.          if (Proxycache( t ).Contains_fresh_copy (Web page p )
9.            begin
10.             Hits=Hits++.            /*Cache Hit*/
11.             Byte. Hits = Byte. Hits + t. Bytes-Retrieved- to-client.
12.             Cache. Update ( t , (DMCM))
13.           end;
14.         else
15.         begin
16.           Cache. Delete( t ).     /*Cache Miss*/
17.           Retrieve _Fresh Copy of Web object p  from origin Server S .
18.           Cache. Push ( t , DMCM )
19.           While (Proxy cache ( t ).Size >Max_size (N)
20.           Cache.Pop( t );  */Remove the Web object with lowest key*/
21.           Switch ( )
22.           Case 1   : "DMCM-GDS".
23.           Case 2   : "DMCM-GDSF".
24.           Case 3   : "DMCM-GD*".
25.       end;
27        While (Condition);
28.   end;
29.   DMCM. Update model();
30. end;
```

Fig. 10. Generic model for Web Proxy caching replacement algorithms using the DMCM algorithm.

with Data Mining Classifier Model is shown in Fig. 10. In (line 1), an initial Data Mining Classifier Model (KNN) is built on history Weblogs. For each Web page p requested from the Proxy cache t that contains the Web page p and then the Web page p is returned to the Web Client. Concerning this performance, measures (line 7–11), in this case, it is considered as *Cache Hit*.Also, the Number of bytes transfers back to the client is counted for the weighted hit rate measure. Once the data are transferred back to the client, the Proxy cache is updated (line 12) by the Data Mining Classifier Model (KNN) based on the weight assignment policy, (Class value of the Web page, i.e., whether the Web page p that can be revisited in future or not).On the contrary, if the requested Web page p is not available in the Proxy cache t or it is stale, a *Cache Miss occurs,* i.e., the Web page q is deleted from the cache (line 16), in this case, the Proxy server forwards the request to the origin server S (line 17), and a fresh copy of the Web page p is retrieved from the origin server S and pushed into the Proxy cache t (line 18). The push method consists of assigning the class value of the Web page p by the Data Mining Classifier Model (KNN) based on the Weight assignment Policy.Also, if the cache space t exceeds the maximum cache size N (line 19), the Web page q from the cache is popped out from the cache (line 20), based on the Class assigned and lowest key value based on the weight assignment policy by the Data Mining Classifier Model (KNN). Such an approach is known as weight assignment cache replacement policy (line 21–24), i.e., each time when the cache gets overflows. Finally, the Data Mining Classifier Model periodically updates the key value of the remaining Web page is stored in the Proxy cache t. This process continues iteratively when the cache performance decreases. Also, notice that the update of the Data Mining Classifier Model (line 28), is dissociated from the online caching of the Web page, and it can be performed in parallel.

Table 9
Parameters for Weight Assignment Policy.

Parameters	Description
L	Inflation factor to avoid cache pollution in the proxy cache t.
$f(p)$	Previous frequency access of Web object p in the proxy cache t.
$F(p)$	Current frequency access of Web object p in the proxy cache t.
$K_{n-1}(p)$	Previous key value of the Web object p in the proxy cache t.
$\Delta T_t(p)$	Difference in time between the current requests and previous requests for the Web object p in the proxy cache t.
$C_t(p)$	Current reference time of the Web object p in the proxy cache t.
$L_t(p)$	Last reference time of the Web object p in the proxy cache t.
$S(p)$	Size of the Web object p.
$K_n(p)$	Current key value of the Web object p.

5.1. Weight assignment policy for cache replacement

The weight assignment policy [32] of the Web object p in the proxy cache t is expressed in Eq. (5), and the parameters are shown

```
/* Algorithm for GDS replacement based on Weight assignment policy and DMCM
1. begin
2.   for each Web page p requested by user do
3.     begin
4.       if Web page p resides in the Proxy cache t then        /* Cache Hit*/.
5.        begin
6.           Update information of by DMCM.
7.           Class of p = KNN Classifier.
8.           if Class of Web page p = 1 then
9.             Push the Web page p into the top of cache and update the key
10.                K_n(p) = L + (F(p)×C(p) + K_{n-1}(p)×(ΔT_t(p)/(C_t(p) − L_t(p)))) / S(p)      (9)
11.       end;
12.     else
13.     begin
14.       if Web page p is not available in the Proxy cache t then /* Cache Miss */
15.         Bring the Web page p into Proxy cache t from the origin Server S
16.         While there is not enough free space in the Proxy cache t .
17.           Remove min{k(q) | q} from the Proxy cache t && Class of q = 0
18.         end While;
19.       Class of p = KNN Classifier.
20.       Class of Web page p = 1.
21.       Push the Web page p in to the top of cache and update the key
22.       K_n(p) = L + (F(p)×C(p) + K_{n-1}(p)×(ΔT_t(p)/(C_t(p) − L_t(p)))) / S(p)      (21)
23.     end;
24.   end;
25. end;
```

Fig. 11. GDS replacement algorithm based on weight assignment policy and DMCM.

in Table 9. From the above strategy, the key value used in the caching system is applied to the Greedy family replacement algorithm (GDS, GDSF, GD*) and the key factor of the replacement algorithm is modified according to Eq. (5) as shown below.

$$K_n(p) = L + \frac{F(p) + K_{n-1}(p) \times \left(\frac{\Delta T_t(p)}{C_t(p) - L_t(p)}\right)}{S(p)} \quad (5)$$

Therefore, whenever the cache replacement occurs, the replacement algorithm replaces the Web object based on the key value used by the weight assignment policy.

From the above strategy, the key value used in the caching system is applied to the Greedy family replacement algorithm (GDS, GDSF, GD*) [2] and the key factor of the replacement algorithm is modified according to the equation as shown above. Therefore, whenever the cache replacement occurs, the replacement algorithm replaces the Web page based on the key value used by the weight assignment policy.

5.2. Integration of GDS replacement with the Data Mining Classifier Model

The GDS policy [2] is adapted with Data Mining Classifier Model when there is a need for cache replacement. The algorithm illustrates GDS cache replacement with Data Mining Classifier Model. In this method, GDS associates a value key value $K(p)$ with each Web page p in the cache. When a Web page p is requested in the Proxy cache t. and it is already available in the Proxy cache t then *Cache Hit occurs,* and the Web page p is pushed onto to the top of the cache. Also, the key value is updated based on the DMCM (KNN) and Weight assignment policy of the caching system, i.e. $K_n(p) = L + \frac{F(p) \times C(p) + K_{n-1}(p) \times \left(\frac{\Delta T_t(p)}{C_t(p) - L_t(p)}\right)}{S(p)}$. Similarly, if the Web page p is not available in the Proxy cache t then *Cache Miss Occurs* and the Web page p retrieved from the origin Server S and if there is no space in the Proxy cache t the Web page q has to be replaced based on the DMCM (KNN) and lowest key value assigned by the weight assignment policy of the caching system, i.e. Web page q with minimum key value ($min_{q \in cache}\{k(q)|q\}$) in the cache is chosen among the other Web pages resident in the Proxy cache t. Subsequently, the values are reduced by $K_{\min}$, and the key value of the Web page p is updated as $K_n(p) = L + \frac{F(p) \times C(p) + K_{n-1}(p) \times \left(\frac{\Delta T_t(p)}{C_t(p) - L_t(p)}\right)}{S(p)}$ and it is pushed to the top of the cache. Also, the Data Mining Classifier Model updates the remaining Web pages. The algorithm for the GDS replacement based on weight assignment policy and DMCM is given in Fig. 11.

5.3. Integration of GDSF replacement with the Data Mining Classifier Model

The GDSF policy [2] is adapted with Data Mining Classifier Model when there is a need for cache replacement. This algorithm illustrates GDSF cache replacement with Data Mining Classifier Model. In this method, GDSF considers variability in cost and size of a Web page p by choosing the victim based on the ratio between the cost and size of documents. GDSF associates a value key value $k(p)$ with each Web page p in the cache. When a Web page p is requested in the Proxy cache t, and it is already available in the

```
/* Algorithm GDSF replacement based on Weight assignment policy and DMCM model */
1. begin
2.   for each Web page p requested by user do
3.     begin
4.       if Web page p resides in the Proxy cache t then          /*Cache Hit */
5.       begin
6.          Update information of p by DMCM.
7.          Class of p = KNN Classifier.
8.          if Class of Web page p = 1 then
9.             Push the Web page p in to the top of cache and update the key
10.            K_n(p) = L + ((F(p) + f(p)) × C(p) + K_{n-1}(p) × (ΔT_t(p) / (C_t(p) − L_t(p)))) / S(p)   (9)
11.      end;
12.      else
13.        begin
14.        if Web page p is not available in the Proxy cache t then /* Cache Miss */
15.           Bring the Web page p into Proxy cache t from the origin Server S
16.           While there is not enough free space in the Proxy cache t .
17.              Remove min{k(q) | q} from the Proxy cache t && Class of q= 0
18.           end While;
19.           Class of p = KNN Classifier.
20.           if Class of Web page p = 1.
21.           Push the Web page p in to the top of cache and update the key
22.           K_n(p) = L + ((F(p) + f(p)) × C(p) + K_{n-1}(p) × (ΔT_t(p) / (C_t(p) − L_t(p)))) / S(p)   (21)
23.        end;
24.    end;
25. end;
```

Fig. 12. GDSF replacement algorithm based on weight assignment policy and DMCM.

Proxy cache t then *Cache Hit occurs* and the Web page p is pushed onto to the top of the cache, also the key value is updated based on the DMCM(KNN) and weight assignment policy of the caching system i.e.$K(p)$ is set to $K_n(p) = L + \frac{(F(p)+f(p))\times C(p)+K_{n-1}(p)\times\left(\frac{\Delta T_t(p)}{C_t(p)-L_t(p)}\right)}{S(p)}$. Similarly, if the Web page$p$ is not available in the Proxy cache t then *Cache Miss Occurs* and the Web page p retrieved from the origin server S and if there is no space in the Proxy cache t the Web page q has to be replaced based on the DMCM, and the lowest key value assigned by the Weight assignment policy of the caching system, i.e. Web page p with minimum key value and class value as 0 is chosen among all the other Web pages resident in the Proxy cache t. Subsequently, these values are reduced by $k_{\min}$, and the key value of the Web page p is updated as $K_n(p) = L + \frac{(F(p)+f(p))\times C(p)+K_{n-1}(p)\times\left(\frac{\Delta T_t(p)}{C_t(p)-L_t(p)}\right)}{S(p)}$ and it is pushed to the top of the cache. Also, the Data Mining Classifier Model updates the remaining Web pages. The algorithm for the GDSF replacement based on weight assignment policy and DMCM is given in Fig. 12.

5.4. Integration of GD* replacement with the Data Mining Classifier Model

The GD* policy [2,3] is adapted with Data Mining Classifier model when there is a need for cache replacement. The algorithm illustrates GD* cache replacement with Data Mining Classifier Model. In this method, GD* captures both popularity and temporal correlation in a Web page. GD*associates a value key value $k(p)$ with each Web page p in the cache. When a Web page p is requested in the Proxy cache t and it is already in the Proxy cache t then *Cache Hit occurs* and the Web page p is pushed onto to the top of the cache, also the key value is updated based on the DMCM (KNN) and Weight assignment policy of the caching System $k(p)$ is set to i.e.

$$K_n(p) = L + \left[\frac{(F(p)+f(p))\times C(p)+K_{n-1}(p)\times\left(\frac{\Delta T_t(p)}{C_t(p)-L_t(p)}\right)}{S(p)}\right]^{\frac{1}{\beta}}$$

Similarly, if the Web page p is not available in the Proxy cache t then *Cache Miss occurs* and the Web page p retrieved from the origin server S and if there is no space in the Proxy cache t the Web page q has to be replaced based on the weighted assignment policy and DMCM, i.e.Web page q with minimum key value $\min_{q\in cache}\{k(q)|q\}$ with class value 0 in the cache is chosen among all the other Web page resident in the Proxy cache t. Subsequently, the values are reduced by$k_{\min}$ and the key value of the Web page p is updated as $K_n(p) = L + \left[\frac{(F(p)+f(p))\times C(p)+K_{n-1}(p)\times\left(\frac{\Delta T_t(p)}{C_t(p)-L_t(p)}\right)}{S(p)}\right]^{\frac{1}{\beta}}$ and it is pushed to the top of the cache. Also, the remaining Web pages are updated by the Classifier Model. The algorithm for the GD* replacement based on weight assignment policy and DMCM is given in Fig. 13

```
/* Algorithm GD replacement based on Weight assignment policy and DMCM*/
1. begin
2.   for each Web page p requested by user do
3.     begin
4.       if Web page p resides in the Proxy cache t then        /* Cache Hit */
5.       begin
6.           Update information of p by DMCM.
7.           Class of p = KNN Classifier.
8.           if Class of Web page p = 1 then
9.           Push the Web page p in to the top of cache and update the key
10.                K_n(p) = L + [ ((F(p)+f(p))×C(p)+K_{n-1}(p)×(ΔT_t(p)/(C_t(p)−L_t(p)))) / S(p) ]^{1/β}   (9)
11.          end;
12.      else
13.      begin
14.          if Web page p is not available in the Proxy cache t then /* Cache Miss */
15.          Bring the Web page p into Proxy cache t from the origin Server S
16.          While there is not enough free space in the Proxy cache t .
17.          Remove min{k(q) | q} from the Proxy cache t && Class of q = 0.
18.      end While;
19.          Class of p = KNN Classifier.
20.          if Class of Web page p = 1.
21.          Push the Web page p in to the top of cache and update the key
22.               K_n(p) = L + [ ((F(p)+f(p))×C(p)+K_{n-1}(p)×(ΔT_t(p)/(C_t(p)−L_t(p)))) / S(p) ]^{1/β}   (21)
23.      end;
24.  end;
25. end;
```

Fig. 13. GD* replacement algorithm based on weight assignment policy and DMCM.

6. Experimental setup for the Web Proxy cache simulation

For the simulation of the Web Proxy cache algorithm, the window based cache is simulator is used for the integration of Data Mining Classifier Model (KNN). Results obtained from the Classifier are taken as input to the Web Proxy cache simulator [33]. The experimental setup is carried out based on the following parameters like Trace file name, cache size, replacement scheme, the content type used. The Trace file name includes the following attributes Timestamp, URL-ID, object size, etc. The cache size used in these experiments may vary in size from 5% to 45% (Maximal volume of the cached content) and the replacement scheme used are, GDS, GDSF, GD*.

6.1. Experimental results for the Web Proxy cache simulation

In this section, the results obtained for the Data Mining Classifier Model-based Web proxy cache replacement algorithms are compared with the traditional cache replacement algorithms. From the experimental results, it is shown that the performs measures of hit and byte hit ratio for the DMCM (KNN) based. Web proxy caching algorithms outperform the traditional caching algorithms in all aspects. Here the experimental results are graphically illustrated, and the results are shown below. The experimental result compares Greedy family algorithms under constant and packet model based on the Data Mining Classifier Model which is shown in Figs. 14 and 15. Respectively

7. Conclusion

The various working modules of the overall work flow as implemented in the paper are data pre-processing, Weighted Random Indexing, clustering the web users, followed by a Data Mining Classifier Model. The results obtained show significant improvement over generic caching and are described further discussed. The KNN classifier outperforms the other machine learning classifier by improving the classification accuracy to 76% which is comparatively higher than the other machine learning algorithms. The

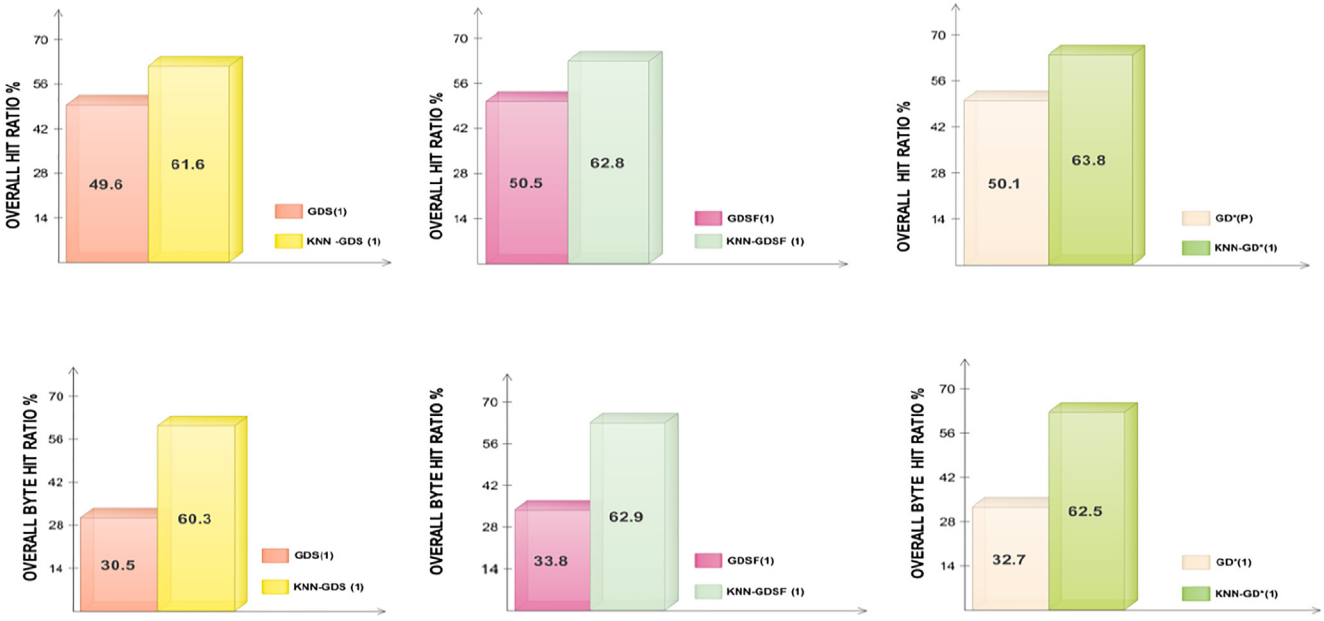

Fig. 14. Comparison of the Overall Hit Ratio of GDS(p), GDSF(p), GD*(p) vs DMCM based replacement.

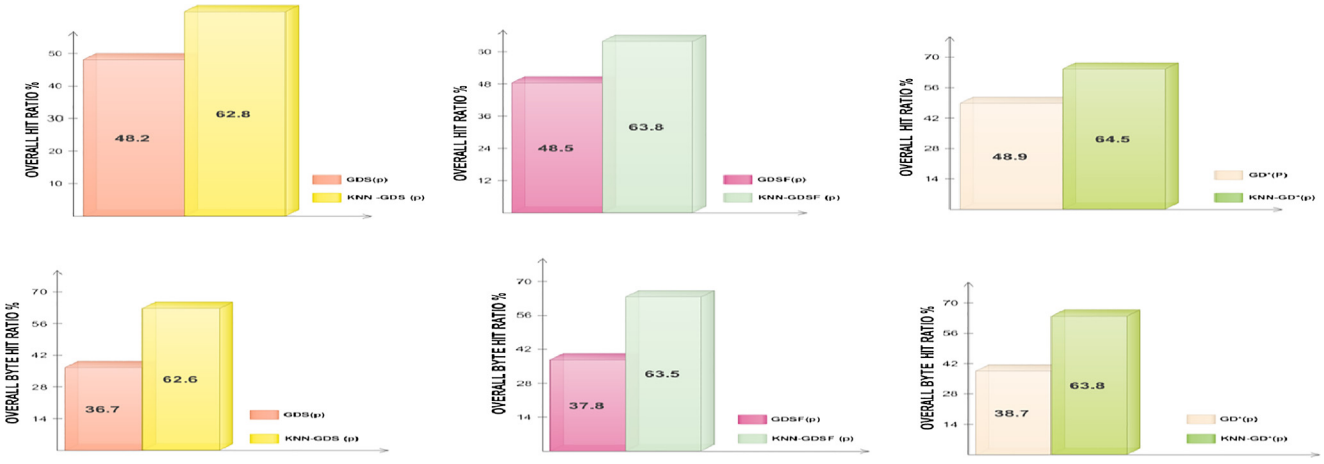

Fig. 15. Comparison of the Overall Byte Hit Ratio of GDS(p), GDSF(p), GD*(p) vs DMCM based replacement.

average precision and recall value for all the clusters generated by the K- means algorithm is around 80.1% and 46% respectively.

The integration of Web user clustering and classifier methods with traditional Web Proxy caching improves the performance of overall hit and byte hit ratio. The hit ratio of GDS, GDSF, and GD* under constant cost model based on DMCM has been improved to 11.4%, 12.8%, 13.7%. The byte hit ratio has been improved to 30.2%, 29.1%, 30.2%. The hit ratio of GDS, GDSF, and GD* under packet cost model based on DMCM has been improved to 26.7%, 23.3%, and 24.4% respectively. The byte hit ratio has been improved to 34.1%, 34.2%, 33.1%. In future we would like to implement advanced clustering and classification algorithms for improving the performance of the model. Also we would like to incorporate our work for CDN (Content Distribution Networks).

References

[1] Baentsch M, Baun L, Molter G, Rothkugel S, Sturn P. World Wide Web caching: the application-level view of the internet. IEEE Commun Mag 1997;35 (6):170–8. doi: https://doi.org/10.1109/35.587725.

[2] Balamash A, Krunz M. An overview of Web caching replacement algorithms. IEEE Commun Surv Tutorials 2004;6(2):44–56. doi: https://doi.org/10.1109/COMST.2004.5342239.

[3] Jin S, Bestavros A. Greedy-dual∗ Web Caching Algorithm. Int J Comput Commun 2001;24(2):174–83. doi: https://doi.org/10.1016/S0140-3664(00)00312-1.

[4] Foong AP, Hu Y-H, Heisey DM. Logistic Regression in an adaptive Web Cache. IEEE Internet Comput 1999;3(5):27–36. doi: https://doi.org/10.1109/4236.793455.

[5] Sajeev GP, Sebastin MP. A novel content classification scheme for Web Caches. Evol Syst 2011;2(2):101–18. doi: https://doi.org/10.1007/s12530-010-9026-6.

[6] Sajeev GP, Sebastin MP. Building semi-intelligent Web cache systems with light weight machine learning technique. Comput Electr Eng 2013;39 (4):1174–91. doi: https://doi.org/10.1016/j.compeleceng.2013.02.005.

[7] Khalid H, Obaidat M. KORA: a new cache replacement scheme. Comput Electr Eng 2000;26(3):187–206. doi: https://doi.org/10.1016/S0045-7906(99)00041-5.

[8] Wen Tian, Ben Choi, Vir Phoba. An Adaptive Web cache access predictor using network. Developments in Applied Artificial Intelligence, Lecture Notes in Artificial Intelligence, vol. 23. Springer; 2002. p. 450–9. doi: https://doi.org/10.1007/3-540-48035-8_44. no. 58.

[9] Koskela T, Heikkonen J, Kaski K. Web Cache optimization with non linear model using networks object features. Comput Networks 2003;43(4):805–17. doi: https://doi.org/10.1016/S1389-1286(03)00334-7.

[10] Cobb J, ElAarag H. Web Proxy cache replacement scheme based on back-propagation neural network. J Syst Software 2008;81(5):450–9. doi: https://doi.org/10.1016/j.jss.2007.10.024.

[11] Sabeghi M, Yaghmaee MH. Using Fuzzy logic to Improve cache replacement decisions. Int J Comput Sci Network Security 2006;6(3):182–8.

[12] Ali W, Shamsuddin SM. Neuro-Fuzzy system in partitioned client-side Web cache. Expert Syst Appl 2011;38(12):14715–25. doi: https://doi.org/10.1016/j.eswa.2011.05.009.

[13] Ali W, Shamsuddin SM, Ismail AS. Intelligent Naïve Bayes approaches for Web Proxy caching. Knowl Based Syst 2012;31:162–75. doi: https://doi.org/10.1016/j.knosys.2012.02.015.

[14] Julian Benadit P, Sagayaraj Francis F, Muruganantham U. Improving the performance of a Proxy Cache using tree augmented Naive Bayes classifier. Procedia Comput Sci 2015;46:184–93. doi: https://doi.org/10.1016/j.procs.2015.02.010.

[15] Julian Benadit P, Sagayaraj Francis F. Improving the performance of a proxy cache using very fast decision tree C classifier. Procedia Comput Sci 2015;48:304–12. doi: https://doi.org/10.1016/j.procs.2015.04.186.

[16] Julian Benadit P, Francis Sagayaraj F, Muruganantham U, Improving the Performance of a Proxy Cache Using Expectation Maximization with Naive Bayes Classifier. In Jain L, Behera H, Mandal J, Mohapatra D. (eds) Computational Intelligence in Data Mining - Volume 2. Smart Innovation, Systems and Technologies, vol 32. pp. 355–368 Springer, New Delhi. https://doi.org/10.1007/978-81-322-2208-8_33.

[17] Pallis G, Vakali A, Pokorny J. A Clustering Based–prefetching scheme on a Web cache environment. Comput Electr Eng 2008;34(4):309–23. doi: https://doi.org/10.1016/j.compeleceng.2007.04.002.

[18] Vakali A. Evolutionary techniques for Web caching. Distrib Parallel Databases 2002;11(1):93–116. doi: https://doi.org/10.1023/A:1013385708178.

[19] Bonino D, Corno F, Squillero G. A Real Time Evolutionary algorithm for Web prediction. In Proceedings of the IEEE/WIC International Conference on Web Intelligence; 2003. p. 139–145. https://doi.org/10.1109/WI.2003.1241185.

[20] Chen Y, Li Z, Wang Z. A GA-Based cache replacement policy. In: Proc. of the 3rd International Conference on Machine Learning and Cybernetics; 2004. p. 263–265. https://doi.org/10.1109/ICMLC.2004.1380674.

[21] NLANR, National Lab of Applied Network Research (NLANR), Sanitized Access Logs. [Online] http://www.ircache.net/2010.

[22] Cooley R, Mobasher B, Srivastava J. Data preparation for Mining World Wide Web Browsing Patterns. Knowl Inf Syst 1999;1(1):5–32. doi: https://doi.org/10.1007/BF03325089.

[23] Kastaniotis G, Maragos E, Douligeris C, Despotis DK. Using Data envelopment analysis to evaluate the efficiency of web caching object replacement strategies. J Network Comput Appl 2012;35(2):803–17. doi: https://doi.org/10.1016/j.datak.2006.06.001.

[24] Sule Gunduz Oguducu. Web Page Recommendation Models: Theory and Algorithms. Morgan and Clay Pool; 2011. https://doi.org/10.2200/S00305ED1V01Y201010DTM010.

[25] Liu H, Keselej V. Combined mining of Web server logs and Web contents for classifying user navigation patterns and Predicting users' future requests. Data Knowl Eng 2007;61:304–30. doi: https://doi.org/10.1016/j.datak.2006.06.001.

[26] Wan M, Jonsson A, Wang C, Li L, Yang Y. Web user clustering and Web prefetching using Random Indexing with Weight functions. Knowl Inf Syst 2012;33(1):89–115. doi: https://doi.org/10.1007/s10115- 011-0453-x.

[27] Jin Hua Xu and Hong Liu, Web user clustering analysis based on K-Means algorithm. In: International Conference on Information, Networking and Automation (ICINA), Kunming, vol. 2, pp. noV2-6-V2-9, 2010 https://doi.org/10.1109/ICINA.2010.5636772.

[28] Wu X, Kumar V, Quinlan JRJ, et al. Top 10 algorithms in Data mining. Knowl Inf Syst 2008;14(1):1–37. doi: https://doi.org/10.1007/s10115-007-0114-2.

[29] Hall M, Frank E, Holmes G, P fahringer B, Reutemann P, Witten I, Weka Data Mining Software; 2009. [Online version Weka 3.7.10]. http://www.cs.waikato.ac.nz/ml/weka.

[30] WUM: "A Web Utilization Miner, 2003" [Online] http://wum.wiwi.huberlin.de.

[31] Zeimpekis D, Gallopoulos E. TMG: A MATLAB Toolbox for Generating Term-Document Matrices from Text Collections. In: Kogan J, Nicholas C, Teboulle M, editors. Grouping Multidimensional Data. Berlin, Heidelberg: Springer; 2006. p. 187–210. doi: https://doi.org/10.1007/3-540-28349-8_7.

[32] Bonchi F, Giannotti F, Gozzi C, Manco G, Nanni M, Pedreschi D, Renso C, Ruggeri S. Web log Data Ware housing and Mining for Intelligent Web Caching. Data Knowl Eng 2001;39(2):165–89. doi: https://doi.org/10.1016/S0169-023X(01)00038-6.

[33] Gonzalez-Cante FJ, Casilari E, Trivino-cabrera A. A Windows Based Web Cache Simulator Tool. In Proc. of the 1st International conference on Simulation tools and techniques for communications, networks and systems & workshops; 2008. p. 1–5, 2008. https://doi.org/10.4108/icst.simutools2008.2933.