# Enhancement of Web Proxy Caching Using Random Forest Machine Learning Technique

**Julian Benadit.P[1], Sagayaraj Francis.F [2], Nadhiya.M[3]**

**[1] Research scholar,**
**Department of Computer Science & Engineering,**
**Pondicherry Engineering College, Puducherry – 605 014, India**

**[2] Professor, Department of Computer Science & Engineering,**
**Pondicherry Engineering College, Puducherry – 605 014, India**

**[3] PG Scholar, Department of Computer Science & Engineering,**
**Dr. S.J.S Paul Memorial College of Engineering and Technology,**
**Puducherry – 605 502, India**

**Abstract**

The Random Forest Tree is an ensemble learning method for Web data classification. In this study, we attempt to improve the performance of the traditional Web proxy cache replacement policies such as LRU and GDSF by integrating machine learning technique for enhancing the performance of the Web proxy cache. Web proxy caches are used to improve performance of the web. Web proxy cache reduces both network traffic and response time. In the first part of this paper, a supervised learning method as Random Forest Tree classifier (RFT) to learn from proxy log data and predict the classes of objects to be revisited or not. In the second part, a Random Forest Tree classifier (RFT) is incorporated with traditional Web proxy caching policies to form novel caching approaches known as RFT-LRU and RFT-GDSF. These proposed RFT-LRU and RFT-GDSF significantly improve the performances of LRU and GDSF respectively.

***Keywords:*** *Web caching, Proxy server, Cache replacement, Classification, Random Forest Tree classifier.*

## 1. Introduction

For the past few years, many researches are going on in Web proxy caching and integration of supervised techniques in Web cache replacement. This paper also comes under this category. Web proxy caching plays a significant part in improving Web performance by conversing web objects that are likely to be visited again in the proxy server close to the user. This internet proxy caching aid in decreasing user perceived latency, i.e. delay from the time missive of request is issued till response is received, reducing network information measure[4, 15]. Cache space is restricted; the space should be used competently. A cache replacement principle is required to handle the cache content [11,4]. If the cache is full when an object desires to be stored, the replacement strategy will work out which objects to be evicted to permit space for the new object.

Table 1: Cache replacement policies

| Policy | Brief description |
|---|---|
| LRU | The least recently used objects are taken first. |
| LFU | The least frequently utilized objects are taken first. |
| SIZE | Big objects are removed first. |
| GDS | It assigns a key value to each object in the cache. The object with the low key value is evicted. |
| GDSF | It expands GDS algorithm by integrating the frequency component into the key word. |

The most common internet caching ways (Table 1) aren't effective enough and flout alternative factors that aren't often visited. This decreases the effective cache size and affects the performance of the online proxy caching negatively. Therefore, a supervised mechanism is needed to manage internet cache content with efficiency.

In the preceding papers exploiting supervised learning methods to cope with the matter [1,6,7,9,10,12,15]. Most of these surveys use an Adaptive Neuro-Fuzzy Inference System (ANFIS) in World Wide Web caching. Though ANFIS training might consume wide amounts of time and need further process overheads.

In this paper, we attempted to increase the performance of the web cache replacement strategies by integrating supervised learning method of Random Forest Tree classifier (RFT). In conclusion, we achieved a large-scale evaluation with other supervised learning algorithm on different log files and the proposed methodology has enhanced the performance of the web proxy cache.

### 1.1 Random Forest Tree Classifier

Random Forest is very unique among popular machine learning methods. Random Forest was presented by Lepetit et.al. [9]. In a Random Forest, the features are randomly selected in each split decision. The correlation between trees are reduced randomly by selecting the features which improve predictive power and provides results for higher efficiency.

**Random Forest Algorithm:**
1) For b= 1 to B:

(a) Draw a bootstrap sample Z* of size N from the training data.
(b) Develop a Random Forest tree $T_b$ to the bootstrapped data, by recursively iterating the subsequent steps for every terminal node of the tree, until the minimum node $n_{min}$ size is reached.

(i) Select $m$ variables at from the $p$ variables.
(ii) Pick the most effective variable/split-point among the $m$.
(iii) Divided the node into two child nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a replacement purpose $x$:

Classification: Let $C_b(x)$ be the class prediction of the $b$ th random-forest tree. Then $C_{rf}^B(x)$= majority vote $\{C_b(x)\}_1^B$.

The Random Forest algorithm for web data classification is as follows:

Drawn $n$ tree bootstrap samples of unique data.

1. For every of the bootstrap samples, raise an unpruned classification tree, with the subsequent modification: at every node, rather than choosing the most effective split among all predictors, randomly sample m try of the predictors and choose the most effective split from among those variables. (Bagging can be thought of as the different case of Random Forests obtained when $m_{try} = p$, the quantity of predictors.)

2. Predict new data by aggregating the predictions of the $n_{tree}$ trees (i.e., majority votes for classification).

An estimate of the error rate is obtained, based on the training data.

The Random Forest is suitable for high dimensional data modeling because it can handle missing values and can handle continuous, categorical and binary data. The Random Forest main features that gained focus are: accurate prediction and better generalizations are achieved due to utilization of ensemble strategies and random sampling.

## 2. Proposed Novel Web Proxy Caching Approaches

The proposed system will present a framework (Fig. 1) For novel Web proxy caching approaches based on machine learning techniques [2,5,19].

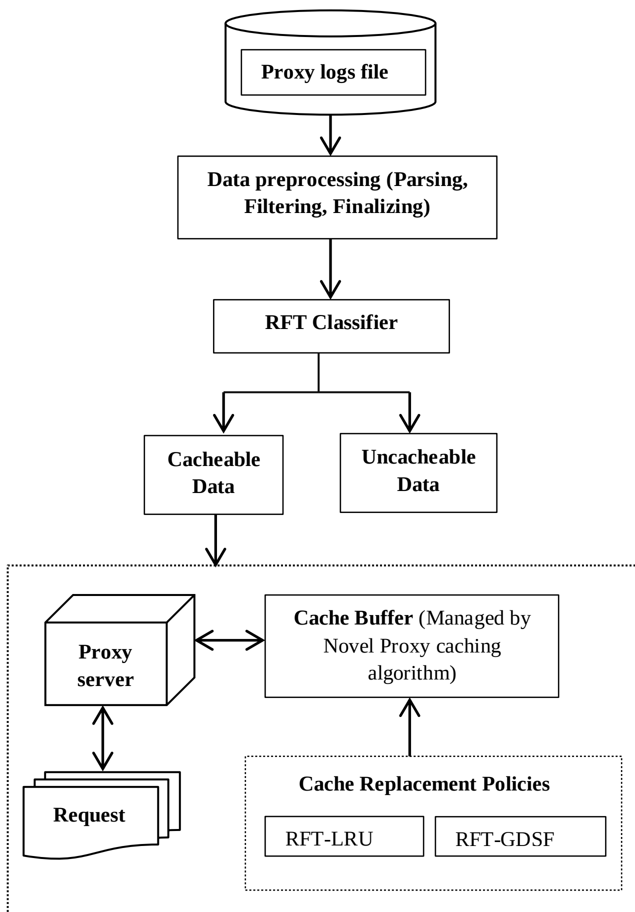


Fig. 1 Novel integrated approach

In the first part, once the dataset is prepared, the machine learning techniques are trained to depend on the concluded dataset to order the web objects into objects that may be revisited or not. In the second part, we present novel Web proxy caching approaches which depend on integrating supervised techniques with traditional Web caching algorithms.

## 2.1 RFT-GDSF

The main advantage of the GDSF [16] principle is that it executes well in terms of the hit ratio. However, the byte hit ratio of GDSF principle is too reduced. Thus, the RFT classifier is integrated with GDSF for advancing the performance in terms of the byte hit ratio of GDSF. The suggested novel proxy caching approach is called RFT-GDSF.

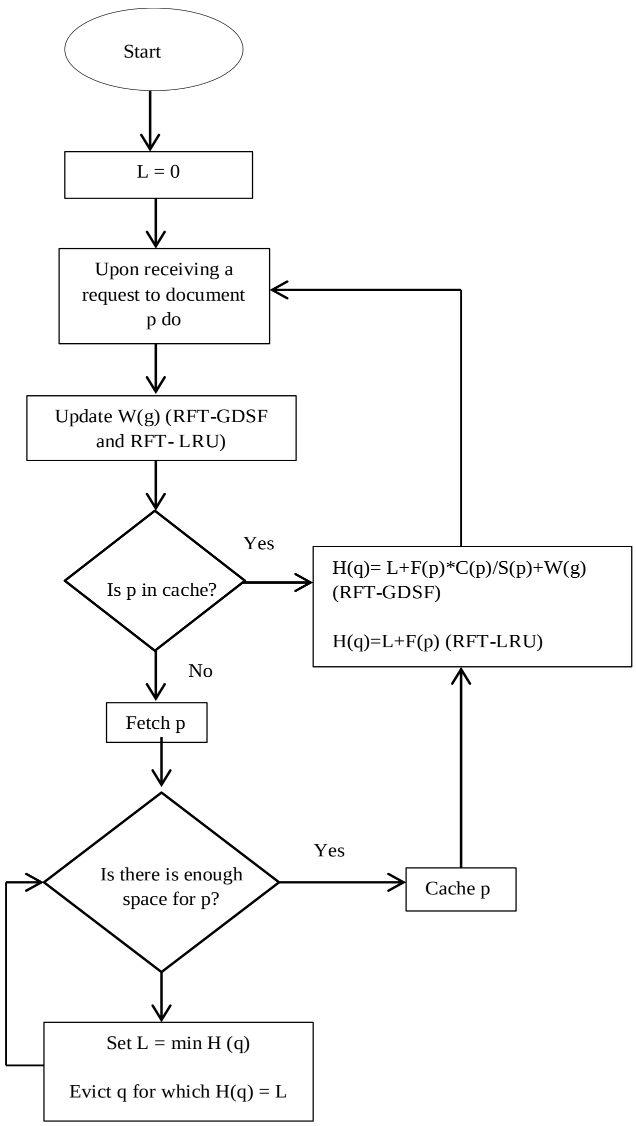


Fig. 2 RFT-LRU and RFT-GDSF policies

In RFT-GDSF, a trained RFT classifier is used to predict the classes of web objects either objects may be re-visited later or not. After this, the classification, assessment is integrated into cache replacement policy (GDSF) to give a key value for each object in the cache buffer; the lowest values are removed first. The proposed RFT-GDSF illustrated Fig. 2.

## 2.2 RFT-LRU

LRU policy[18] is the most common web proxy caching scheme among all the Web proxy caching algorithms [1,9]. But, LRU policy suffers from cache pollution, which means that unpopular data's will remain in the cache for a long period. For reducing cache pollution in LRU, a RFT classifier is joint with LRU to form a novel approach (Fig. 2) Called RFT-LRU.

# 3. Experimental Result

## 3.1 Proxy Log File Collection

We obtained data for the proxy log files of the Web object requested in some proxy servers found (Table 2) nearby the United States of the IR cache network for fifteen days [21].

Table 2: Different proxy log file

| **Proxy Data set** | **Proxy server name** | **Location** | **Duration Of Collection** |
|---|---|---|---|
| UC | uc.us.ircache.net | Urbana-Champaign, | 1/8-4/9/2011 |
| BO2 | bo2.us.ircache.nt | Boulder-Colorado, | 1/8-4/9/2011 |
| SV | sv.us.ircache.net | Silicon, Valley, | 1/8-4/9/2011 |
| SD | sd.us.ircache.net | San Diego, | 1/8-4/8/2011 |
| NY | ny.us.ircache.net | New York | 1/8-4/9/2011 |

An access proxy log entry generally consists of the consequent fields: timestamp, lapsed time, log tag, message protocol code, size, user identification, request approach, URL, hierarchy documents and hostname, and content type.

## 3.2 Data pre-processing

In the data pre-processing [14], irrelevant and not valid request, is removed from the logs proxy files. The pre-processing, including parsing, filtering and finalizing, has a strong influence on the performance, therefore, a correct preparation is required in order to achieve results reflecting the behavior of the algorithms.

After the pre-processing, the final format of our data consist of URL ID, timestamp, lapsed time, size and set of Mesh data (type) as shown in Table 3.

Table 3: Sample of pre-processed data set

| ***URL-id*** | ***Timestamp*** | ***Lapsed Time*** | ***Size*** | ***Type*** |
|---|---|---|---|---|
| **1** | **1082348905.73** | **53** | **43097** | **Application** |
| 2 | 1082348907.41 | 703 | 14179 | Application |
| 3 | 1082348908.47 | 284 | 1276 | image/jpeg |
| **4** | **1082349578.75** | **263** | **25812** | **image/jpeg** |
| **1** | **1082349661.61** | **71** | **43097** | **application** |
| 5 | 1082349675.35 | 203 | 8592 | text/html |
| 6 | 1082349688.90 | 231 | 24196 | text/html |
| **4** | **1082349753.72** | **875** | **25812** | **text/html** |
| **4** | **1082350464.01** | **173** | **25812** | **text/html** |
| **1** | **1082351887.76** | **115** | **43097** | **application** |
| **4** | **1082352609.09** | **35** | **25812** | **text/html** |
| **1** | **1082352861.56** | **311** | **43097** | **application** |

## 3.3 Training phase

The training datasets are prepared, the desired characteristics of Web objects are extracted from pre-processed proxy log files. These features comprise of URL id, timestamp, lapsed time, size and category of Web object (type).

Table 4: Sample of training data set

| **Inputs** | | | | | | |
|---|---|---|---|---|---|---|
| ***Recency*** | ***Frequency*** | ***SWL frequency*** | ***Retrieval Time*** | ***Size*** | ***Type*** | ***Output*** |
| **900** | **1** | **1** | **53** | **43097** | **5** | **1** |
| 900 | 1 | 1 | 703 | 14179 | 5 | 0 |
| 900 | 1 | 1 | 284 | 1276 | 2 | 0 |
| **900** | **1** | **1** | **263** | **25812** | **2** | **1** |
| **900** | **2** | **2** | **71** | **43097** | **5** | **0** |
| 900 | 1 | 1 | 203 | 8592 | 1 | 0 |
| 900 | 1 | 1 | 231 | 24196 | 1 | 0 |
| **900** | **2** | **2** | **875** | **25812** | **1** | **1** |
| **900** | **3** | **3** | **173** | **25812** | **1** | **0** |
| **1226.15** | **3** | **1** | **115** | **43097** | **5** | **1** |
| **1145.08** | **4** | **1** | **35** | **25812** | **5** | **0** |
| **900** | **4** | **2** | **311** | **43097** | **5** | **0** |

Consequently, these features are transformed to the input and output dataset or training forms in the format $< a_1, a_2, a_3, a_4, a_5, a_6,$ b>. $a_1$ is recency of mesh data based on sliding window, $a_2$ is frequency of mesh data, $a_3$ is frequency of mesh data based on sliding window, $a_4$ is retrieval time of mesh data $a_5$ is size of mesh data, $a_6$ is category of mesh data. $a_1 \ldots a_6$ Represent the inputs and b represents the output of the requested mesh data. $a_1$ And $a_3$ are extracted based on a sliding window. The sliding window of a request is that the period, afar and later once the demand were created. In additional, the sliding window ought to be around the signify time that the data usually stays during a cache (SWL is 15 min).

Similarly the data are classified into five types: HTML with worth 1, image with worth 2, audio with worth 3, video with worth 4, application with worth 5 and others with worth 0. The worth of b will be assigned to 1 if the object might be re-visited again within the progressive sliding window. Otherwise the output should be assigned to 0. One time the dataset is prepared (see Table 4), the machine learning techniques is taught depending on the concluded dataset to categorize the World Wide Web objects into objects that will be re-visited or not.

Each proxy dataset is then separated randomly into training data (75%) and testing data (25%). Consequently, the dataset is normalized according into the series [0, 1]. When the dataset is arranged and normalized, the machine learning methods are applied using WEKA3.7.10 [20] see Fig. 3 and 4.

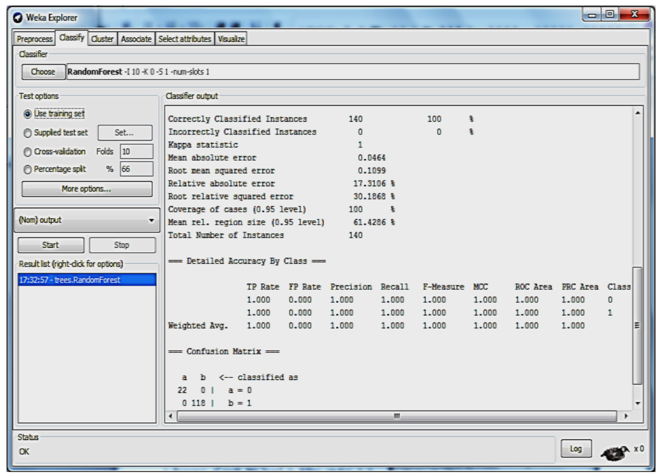


Fig. 3 Training dataset classification

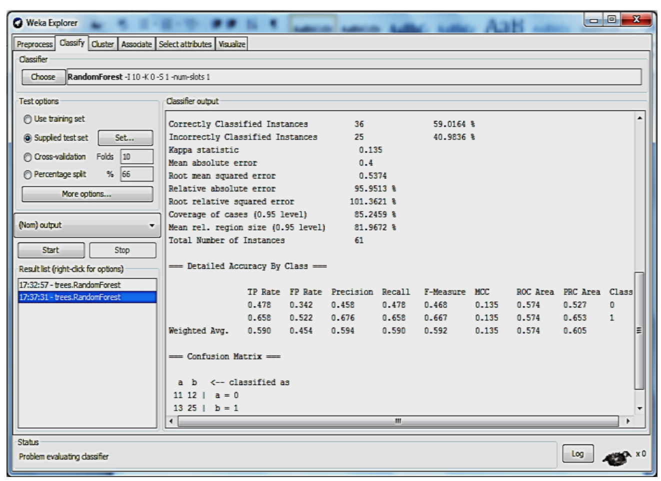


Fig. 4 Testing dataset classification

### 3.4 Web proxy cache Simulation

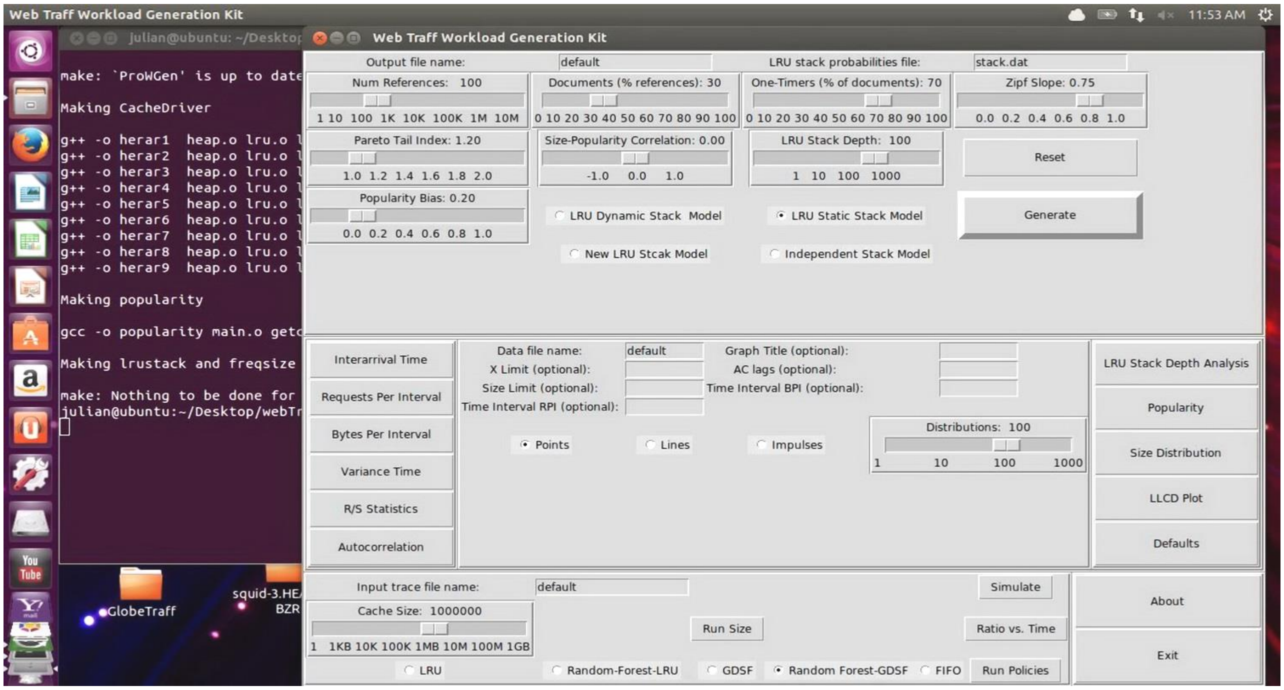


Fig. 5 WebTraff simulator

The simulator WebTraff [13] can be modified to rendezvous our suggested proxy caching approaches. WebTraff simulator is used to evaluate distinct replacement Policies such as LRU, LFU, GDS, GDSF, FIFO and RAND policies Fig. 5. The trained classifiers are integrated with WebTraff to simulate the suggested novel World Wide Web proxy caching approaches. The WebTraff simulator receives the arranged log proxy document as input and develops file encompassing performance measures as outputs.

## 4. Performance Evaluation

### 4.1 Classifier Evaluation

A correct classification ratio (CCR) is a measure for estimating classifier. However, CCR alone is deficient for evaluating the performance of a classifier, particularly if the data are unbalanced. In an unbalanced data item, where The data set covers significantly more popular class than smaller class instances, one can always select the popular class and obtain good CCR [4] see Table 7.

We address that the object will belong to the positives class if the object is re-visited again either the forward-looking SWL.

Table 5: The most common measures

| Measure name | Formula |
|---|---|
| Correct classification ratio | $CCR = \frac{\#correctly\ classified\ examples}{\#total\ examples}(\%)$ |
| True positive ratio | $TPR = \frac{TP}{TP+TN}(\%)$ |
| True negative ratio | $TNR = \frac{TN}{TN+FP}(\%)$ |
| G mean | $GM = \sqrt{TPR * TNR}(\%)$ |

Table 6: Confusion matrix

| | Assessed positive | Assessed negative |
|---|---|---|
| Actual positive | True Positive (TP) | False Negative (FN) |
| Actual negative | False Positive (FP) | True Negative (TN) |

Table 7: CCR for different proxy datasets

| Datasets | CCR of training data set | | CCR of testing data set | |
|---|---|---|---|---|
| | *RFT* | *ANFIS* | *RFT* | *ANFIS* |
| BO2 | 0.959 | 0.845 | 0.950 | 0.781 |
| NY | 0.920 | 0.689 | 0.919 | 0.724 |
| UC | 0.976 | 0.872 | 0.950 | 0.770 |
| SV | 0.935 | 0.707 | 0.931 | 0.725 |
| SD | 1.000 | 0.642 | 0.956 | 0.751 |
| Average | 0.958 | 0.751 | 0.941 | 0.750 |

Otherwise, the Web object will belong to the contradictory class. From proxy files, we can observe that most World Wide Web objects are remained just one time using the users. Hence, the contradictory class describes the most class, while the positive class contains the smaller class, which is the utmost important class in Web caching. Therefore, the true positive ratio (TPR) and the true negative ratio (TNR) can furthermore be utilized to assess the performance of the machine learning methods using some common measures as shown in Table 5 and 6.

Table 8: TPR and TNR for training data sets

| Datasets | TPR for training set | | TNR for training set | |
|---|---|---|---|---|
| | ***RFT*** | ***ANFIS*** | ***RFT*** | ***ANFIS*** |
| BO2 | 0.839 | 0.708 | 0.868 | 0.982 |
| NY | 1.000 | 0.591 | 0.828 | 0.786 |
| UC | 0.874 | 0.681 | 0.868 | 0.883 |
| SV | 0.827 | 0.552 | 0.796 | 0.861 |
| SD | 1.000 | 0.484 | 0.870 | 0.799 |
| Average | 0.908 | 0.603 | 0.840 | 0.862 |

Table 9: TPR and TNR for testing data sets

| Datasets | TPR for testing set | | TNR for testing set | |
|---|---|---|---|---|
| | ***RFT*** | ***ANFIS*** | ***RFT*** | ***ANFIS*** |
| BO2 | 0.821 | 0.681 | 0.767 | 0.881 |
| NY | 0.882 | 0.591 | 0.823 | 0.857 |
| UC | 0.869 | 0.693 | 0.869 | 0.847 |
| SV | 1.989 | 0.552 | 0.769 | 0.898 |
| SD | 0.889 | 0.508 | 0.756 | 0.994 |
| Average | 0.894 | 0.605 | 0.797 | 0.856 |

Table 10: G mean for different proxy datasets

| Datasets | G mean for training set | | G mean for testing set | |
|---|---|---|---|---|
| | ***RFT*** | ***ANFIS*** | ***RFT*** | ***ANFIS*** |
| BO2 | 0.901 | 0.571 | 0.898 | 0.778 |
| NY | 0.912 | 0.791 | 0.902 | 0.889 |
| UC | 0.920 | 0.875 | 0.817 | 0.787 |
| SV | 0.875 | 0.852 | 0.893 | 0.863 |
| SD | 0.908 | 0.808 | 0.982 | 0.858 |
| Average | 0.903 | 0.779 | 0.888 | 0.835 |

Table 11: The training time (in Sec) for different data sets

| Datasets | Training time ( in seconds) | |
|---|---|---|
| | ***RFT*** | ***ANFIS*** |
| BO2 | 0.12 | 20.39 |
| NY | 2.00 | 22.66 |
| UC | 0.56 | 18.54 |
| SV | 0.21 | 16.18 |
| SD | 0.65 | 16.92 |

Gmean (GM) is used to estimate the overall performance of the machine learning methods, as shown in table 5.

Table 8 and Table 9 displays a relationship among the performance measures of RFT and ANFIS for five dissimilar proxy datasets in the training and testing stage. As can be discerned from Table 8 and 9, all of RFT and ANFIS yield good performance. Table 8 and 10 apparently displays that the RFT accomplishes the best TPR and Gmean for all data sets. On the Contrary, ANFIS achieve the worst TPR and Gmean for all data sets. This is because ANFIS tends to classify most of the pattern as the most class.

This contributes to getting the highest TNR of ANFIS. A higher weight is set to a positive class, while fewer weights are fixed to a negative class. Thus, the RFT has better TPR when related to further approaches; this specifies that RFT can forecast the positive or lesser class which comprises the objects that might be re-visited within the close to future.

In addition, the computational time for training RFT, ANFIS can be measured on the same computer for dissimilar datasets, as seen in Table 11. As expected, RFT is faster than ANFIS for all data sets. Thus, we can conclude that the applications of RFT in web proxy caching are more valuable and effective when related to other algorithms.

## 4.2 Evaluation of Integrated Web proxy caching

**Performance Measure:** In web caching, hit ratio (HR) and byte hit ratio (BHR) are two commonly utilized metrics for assessing the performance of web proxy caching strategies [1,9,15]. HR is well-defined as the ratio of the number of demands served from the proxy cache and the complete number of demands. BHR denotes to the number of bytes assisted from the cache, riven up by the complete number of byte assisted. It is important to memo that HR and BHR work in slightly opposite ways.

It is very difficult to accomplish the best performance for both metrics [1]. This is due to the fact that the strategies that increase HR typically give preference to little objects, but these strategies are inclined to decline BHR by giving less concern to bigger objects. On the contrary, the strategies that do not give preference to small objects tend to increase BHR at the expense of HR [1].

In terms of HR, the outcomes of Fig.6 clearly show that RFT-LRU and RFT-GDSF advance the performance in terms of HR for GDSF and LRU respectively for all proxy datasets. On the opposing, the HR of LRU-RFT is similar or slightly not as good as than the HR of GDSF.

IJCSI International Journal of Computer Science Issues, Vol. 11, Issue 3, No 1, May 2014
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org 

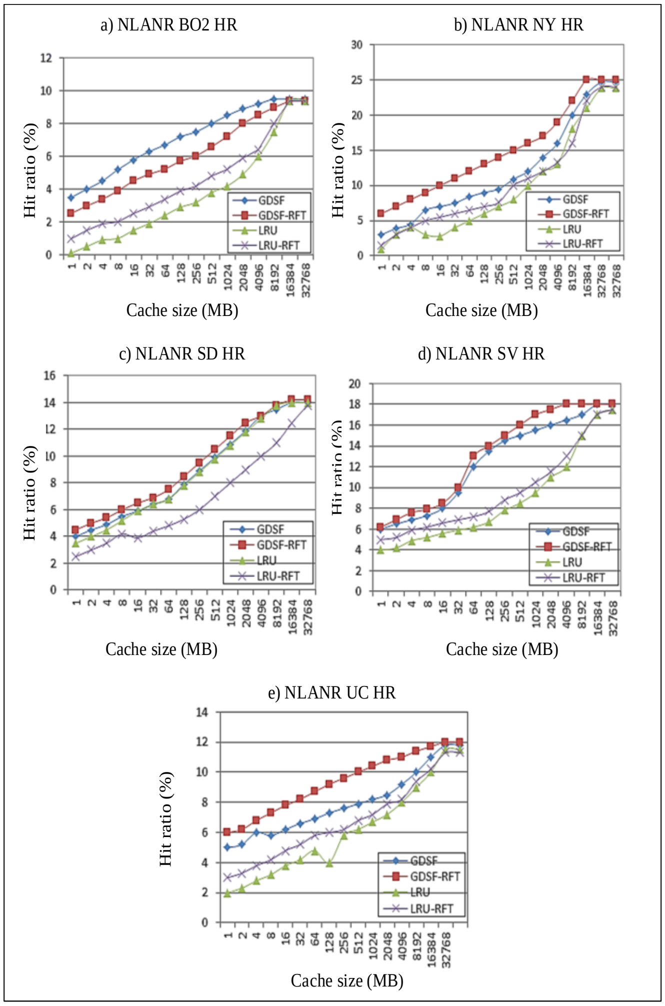


Fig. 6 Hit ratio for different data sets

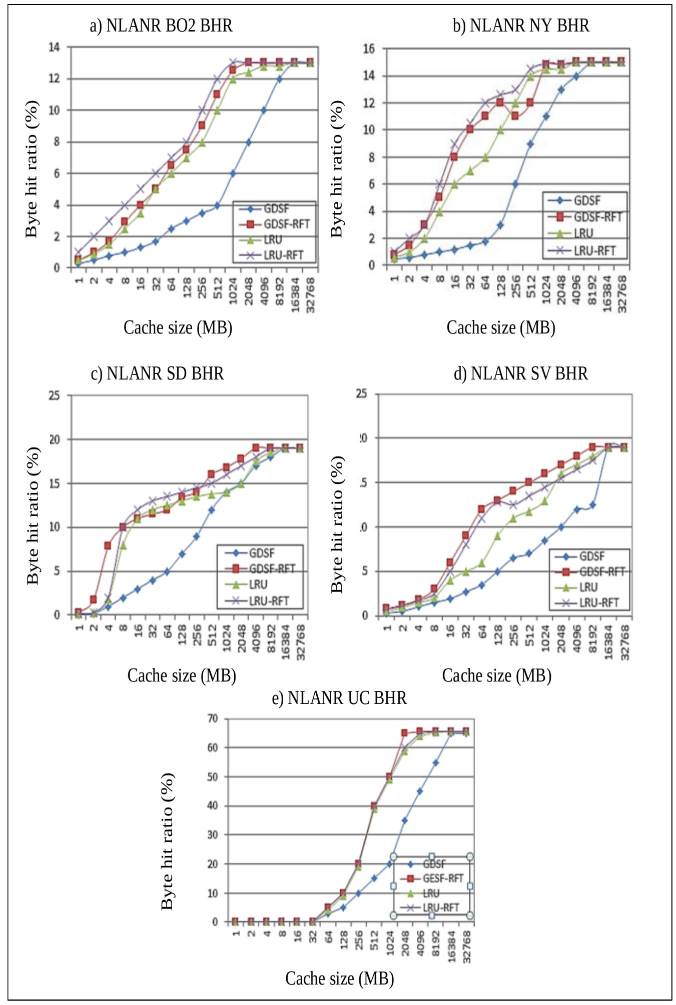


Fig. 7 Byte hit ratio for different datasets

In terms of BHR, Fig. 7 illustrates that BHR of LRU-RFT is better than BHR of GDSF-RFT for the five proxy datasets. This is anticipated, since the LRU policy eliminates the old objects despite of their sizes.

It is very difficult to accomplish the best performance for both metrics [1]. This is due to the fact that the strategies that increase HR typically give preference to little objects, but these strategies are inclined to decline BHR by giving less concern to bigger objects. On the contrary, the strategies that do not give preference to small objects tend to increase BHR at the expense of HR [1].

The norms of HR and BHR for five proxy datasets in all specific cache size are computed as Eq. (6). Wherever, ER is the percent of enhancement attained by the proposed technique (PT) over the conventional technique (CT).

$$ER = \frac{(PT - CT)}{CT} \times 100\ (\%) \tag{6}$$

The enhancement ratios (ER) of the performances in terms of HR and BHR which are attained using the suggested approaches are determined and concise in Table 12.

The outcomes in Table 12 specify that RFT-GDSF increases GDSF performance in terms of HR up to 20.90% and in terms of BHR by up to 115.56% and RFT-LRU over LRU is up to 31.87% in terms of HR and up to 32.34% in terms of BHR.

Table 12: Enhancement ratio

| | RFT-GDSF Over GDSF | | RFT-LRU Over LRU | |
|---|---|---|---|---|
| ***Cache size (MB)*** | ***HR*** | ***BHR*** | ***HR*** | ***BHR*** |
| 1 | 20.90 | 33.01 | 26.64 | 24.17 |
| 2 | 18.19 | 97.46 | 31.87 | 27.68 |
| 4 | 14.27 | 34.69 | 15.04 | 32.34 |
| 8 | 12.33 | 47.69 | 26.70 | 27.95 |
| 16 | 10.94 | 95.46 | 30.77 | 18.53 |
| 32 | 9.61 | 86.66 | 8.86 | 15.06 |
| 64 | 9.27 | 58.77 | 61.08 | 16.38 |
| 128 | 6.66 | 115.56 | 4.77 | 8.41 |
| 256 | 4.73 | 82.80 | 3.47 | 4.87 |
| 512 | 2.64 | 68.49 | 5.18 | 3.44 |
| 1024 | 1.93 | 52.12 | 6.14 | 1.92 |
| 2048 | 0.55 | 47.75 | 4.14 | 0.57 |
| 4096 | 0.23 | 26.53 | 1.64 | 0.56 |
| 8192 | 0.13 | 8.29 | 1.31 | 0.19 |
| 16,384 | 0.04 | 1.43 | 0.50 | 0.14 |
| 32,768 | 0 | 0.26 | 0.12 | 0.09 |

## 5. Conclusion

This study has suggested two novel web proxy caching approaches, namely RFT-LRU, and RFT-GDSF for improving the performance of the conventional World Wide Web proxy caching algorithms. Primarily, RFT discovers from World Wide Web proxy log file to forecast the categories of objects to be revisited or not. Experimental results have revealed that RFT achieves much better true positive rates, and performance much faster than ANFIS in all proxy datasets. More importantly, the trained classifiers are combined effectually with conventional Web proxy caching to provide more productive proxy caching policies.

## References

[1] S. Romano, H. ElAarag, “A neural network proxy cache replacement strategy and its implementation in the squid proxy server,” Neural Computing and Applications, vol. 20, 2011,pp. 59-78.

[2] G. Sajeev, M. Sebastian, “A novel content classification Scheme for web caches,” Evolving Systems vol. 2, 2011, pp. 101–118.

[3] C.J. Huang, Y.W. Wang, T.H. Hung, C. -F. Lin, C.-Y. Li, H.M. Chen, P.C. Chen, J.J. Liao, “Applications of machine learning techniques to a sensor- network-based prosthesis training system,” Applied Soft Computing, Vol. 11, 2011,pp. 3229-3237.

[4] H.T. Chen,”Pre-fetching and Re-fetching in web caching system”, Algorithms and Simulation, Trent University, Peterborough, Ontario, Canada, 2008.

[5] C..Kumar, J.B. Norris, A new approach for a proxy-level web caching mechanism,” Decision support System, vol. 46, 2008, pp. 52-60.

[6] J. Cobb & H. ElAarag, Web proxy cache replacement scheme based on back-propagation neural network. Journal of Systems and Software,vol. 81, 2008 pp.1539-1558.

[7] Farhan, “Intelligent web caching architecture,”, Faculty of Computer Science and Information System, UTM University, Johor, Malaysia, 2007.

[8] B. Liu, “Web data mining: exploiting hyperlinks, contents, and usage data,” Springer, 2007.

[9] Lepetit, V, P. Fa,“ Key point Recognition using randomized trees,” IEEE trans. Pattern Anal. Mach. Intell Vol.28, No.9, 2006, pp. 1465-1479.

[10] W.Kin-Yeung, “Web cache replacement policies a pragmatic approach”, IEEE Network, Vol. 20, 2006, pp.28-34.

[11] S. Podlipnig, L. Boszormenyi, “A survey of web cache replacement strategies,” ACM Computing Surveys, vol. 35, 2003, pp. 374-398.

[12] T. Koskela, J. Heikkonen, & K. Kaski, Web cache optimization with nonlinear model using object features. Computer Networks 2003, 43, 805-817.

[13] N. Markatchev, C. Williamson, “WebTraff: a GUI for Web proxy cache workload modeling and analysis,” Proceedings of the10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, IEEE Computer Society, 2002, pp.356.

[14] J. Han, M. Kamber, “Data Mining: Concepts and Techniques,” Morgan Kaufmann, 2001.

[15] I.Bose, H.K.Cheng,” Performance models of a firm's proxy cache server,” Decision Support System and Electronic Commerce, Vol.29, 2000, pp.45-57.

[16] L. Cherkasova, “Improving WWWProxies Performance with Greedy-Dual-Size-Frequency Caching Policy”, Technical Report HPL-98-69R1, Hewlett-Packard Laboratories, November 1998.

[17] P. Lorenzetti and L. Rizzo, “Replacement Policies for a Proxy Cache”, Technical Report, University Pisa, December1996.

[18] E. O’Neil, P. O’Neil and G. Weikum, “The LRU-K Page Replacement Algorithm for Database Disk Buffering”, *Proceedings of SIGMOD ‘93*, Washington, DC, May 1993.

[19] LAN H. Witten, Eibe Frank, Mark A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kauffmann, 2011.

[20] WEKAtool:Available at http://www.cs.waikato.ac.nz/ml/weka/.

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

**Julian Benadit. P** received the B. Tech degree in computer science engineering from Pondicherry University, Puducherry, India and the M.E. Degree in computer science engineering from Anna University, Chennai, India. He is currently pursuing the Ph.D degree in computer science engineering at Pondicherry Engineering College, Pondicherry University, Puducherry, India.

Since 2006, he has been an Assistant Professor with the computer science Engineering Department, in the consortium engineering college affiliated to Pondicherry University.

His research interest includes web caching, web prefetching, machine learning, content distribution network. He was the Associate member of Professional society Institution of Electronics and Telecommunication Engineers (IETE), Computer Society of India (CSI) and Institution of Engineers (IE), Indian Society for Technical Education (ISTE).

**Sagayaraj Francis.F** received the B.Sc, computer science in Madras university and M.Sc, computer science at St.Joseph college Trichy (Autonomous) and M.Tech degree in computer science Engineering from the Pondicherry University, Puducherry, India and he obtained his Ph.D degree in Computer Science Engineering from Pondicherry University, Puducherry, in2008. Currently, he is working as a professor in the Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry, India. His research interest includes Database Management System, Knowledge and Intelligent system, Data analysis, Data Modeling.

**Nadhiya.M** received Master of Computer Application (MCA) degree from Anna University, Tamilnadu, India. She is currently pursuing the M.Tech degree in computer science engineering at Dr. S.J.S Paul Memorial College of Engineering and Technology, Pondicherry University, Puducherry, India. She currently works in the domain Web caching, Machine learning.