IJESRT: 8(5), May, 2019 ISSN: 2277-9655

International Journal of Engineering Sciences & Research Technology

(A Peer Reviewed Online Journal)
Impact Factor: 5.164

IJESRT

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

WEB PROXY CACHE REPLACEMENT POLICIES USING NAÏVE BAYES (NB) MACHINE LEARNING TECHNIQUE FOR ENHANCED PERFORMANCE OF WEB PROXY

V. Raghunatha Reddy
Assistant Professor, Department of Computer Science & Technology, Sri Krishnadevaraya University, Anantapuramu, INDIA

DOI: 10.5281/zenodo.7027357

ABSTRACT

Web cache is a mechanism for the temporary storage (caching) of web documents, such as HTML pages and images, to reduce bandwidth usage, server load, and perceived lag. A web cache stores the copies of documents passing through it and any subsequent requests may be satisfied from the cache if certain conditions are met. In this paper, Naïve Bayes (NB) a machine learning technique has been used to increase the performance of traditional Web proxy caching policies such as SIZE, and Hybrid. Naïve Bayes (NB) is used and integrated with traditional Web proxy caching techniques to form better caching approaches known as NB–SIZE and NB–Hybrid. The proposed approaches are evaluated by trace-driven simulation and compared with traditional Web proxy caching techniques. Experimental results have revealed that the proposed NB–SIZE and NB–Hybrid significantly increased Pure Hit-Ratio, Byte Hit-Ratio and reduced the latency when compared with SIZE and Hybrid.

Keywords: Web caching, Proxy Cache, Cache replacement, Classification, Naïve Bayes, Machine Learning.

1. INTRODUCTION

Web proxy caching plays a key role in improving Web performance by keeping Web objects that are likely to be visited again in the proxy server close to the user. This Web proxy caching helps in reducing user perceived latency, i.e. delay from the time a request is issued until response is received, reducing network bandwidth utilization, and alleviating loads on the original servers. Since the space apportioned to a cache is limited, the space must be utilized effectively. Therefore, an intelligent mechanism is required to manage Web cache content efficiently. The cache replacement is the core or heart of Web caching. Thus, the design of efficient cache replacement algorithms is extremely important and crucial for caching mechanism achievement [1]. The most common Web caching methods are not efficient enough and may suffer from a cache pollution problem, since they consider just one factor and ignore other factors that may have an impact on the efficiency of Web proxy caching [2]. Many Web proxy caching policies have attempted to combine some factors which can influence the performance of Web proxy caching for making decisions about caching.

However, this is not an easy task, because examining one factor in a particular environment may be more important in one environment which may not be same in other environments [3]. The challenge lies in predicting which Web objects should be cached and which Web objects should be replaced to make the best use of available cache space, improve hit rates, reduce network traffic, and alleviate loads on the original server [4]. Web proxy log files record the activities of the users in a Web proxy server. These proxy log files contain complete and prior knowledge of future accesses. The availability of Web proxy log files that can be used as training data is the main motivation for utilizing machine learning techniques in adopting Web caching approaches. Recent studies have proposed that using machine learning techniques is proved to cope with the above problem [5]. Naïve Bayes (NB) is popular supervised learning algorithms that perform classifications more accurately and faster than other algorithms [6]. Naïve Bayes (NB) algorithms have a wide range of applications such as text classification, Web page classification and bioinformatics applications. Hence, Naïve Bayes (NB) can be utilized to produce promising solutions for Web proxy caching. Naïve Bayes (NB) classifier has been applied successfully in many domains.

This paper combines the most significant factors using Naïve Bayes (NB) classifier for predicting Web objects that can be re-visited later. In this paper, we present new approaches that depend on the capability of Naïve Bayes (NB) classifier to learn from Web proxy logs files and predict the classes of objects to be re- visited or not. The trained Naïve Bayes (NB) classifier can be effectively incorporated with traditional Web proxy caching algorithms to present novel Web proxy caching approaches with good performance in terms of hit ratio and byte hit ratio and reduced latency. The remaining parts of this paper are organized as follows. Background and related works are presented in Section 2. The framework for improved Web proxy caching approaches based on machine learning techniques is illustrated in Section 3. Implementation and experimental results are presented in Section 4. Section 5 discusses performance evaluation and discussion. Finally, Section 6 concludes the paper.

II. BACKGROUND AND RELATED WORK

A. How Web Caches Work

Web caching is the temporary storage of web objects (such as HTML documents) for later retrieval. The three significant advantages of web caching are [8]:

a. Reduced bandwidth consumption.
b. Reduced server load.
c. Reduced latency.

All types of caches have a set of rules that theyuse to determine when to serve an object from the cache, if it is available. Some of these rules are set in the protocols such as HTTP 1.1 and some are set by the administrator of the cache.

The following are the most common rules that are followed for a particular request:

1. If the object's headers directive tell the cache not to keep the object, it won't Cache the Object. Also, if no validator is specified, most caches will mark the object as uncacheable.
2. If the object is authenticated or secure, it will not be cached.
3. A cached object is considered to be fresh (that is, able to be sent to a client without checking with the origin server) if:
• The object has an expiry time or other age-controlling directive set, and is still within the fresh period.
• If a browser cache has already seen the object, and hasbeen set to check once in a session.
• If a proxy cache has seen the object very recent, and itwas modified relatively long ago.
Fresh documents are served directly from the web cache, without checking with the origin server.
4. If an object is stale, the origin server will be asked to validate the object, or tell the cache whether the copy that it has is still good enough to serve or not.

Together, freshness and validation are the two most important ways that a cache works with content. A fresh object will be available instantly from the cache, while a validated object will avoid sending the entire object over again if it has not changed. Web content canbe cached at a number of different locations along the path between a client and an origin server. The 3 types of Web Caches are

a. Browser Cache,
b. Proxy Cache, and
c. Surrogate/Server Cache.

The different types of caches are shown in Figure 1.

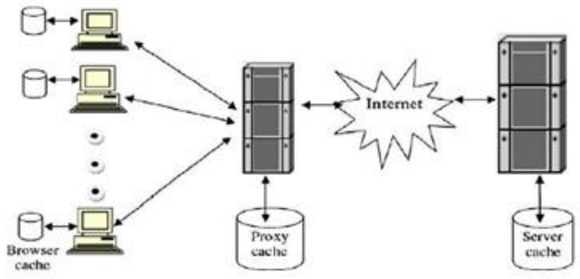


Fig. 1: Different types of caches

B. Traditional Web proxy caching algorithms

Cache replacement algorithms or replacement policies are optimizing instructions or algorithms that a computer program or a hardware-maintained structure can follow, in order to manage a cache of information stored on the computer. When the cache is full, the replacement algorithm must choose which items to discard to make room for the new ones. The most widely used replacement algorithms include Least Recently Used (LRU), Least Frequently Used (LFU), Most Recently Used (MRU), SIZE [9], Greedy-Dual- Size(GDS), Hybrid [11], Lowest Relative Value(LRV)[10] etc.

Least Recently Used (LRU) discards the least recently used web items first. This algorithm requires keeping track of what was used and when, which is expensive if one wants to make sure the algorithm always discards the least recently used item. General implementations of this technique require keeping "age bits" for cache-lines and track the "Least Recently Used" cache-line based on age-bits. In such an implementation, every time a cache-line is used, the age of all other cache-lines also changes. Least Frequently Used (LFU) counts how often an item is needed. Those that are used least are often discarded first. Most Recently Used (MRU) discards, in contrast to LRU, the most recently used web objects first. SIZE policy [9] is one of the common Web caching policies that replace the largest object(s) from a cache when space is needed for a new object. Thus, a cache can be polluted with small objects which will not be accessed again. Williams et al. [9] presented taxonomy of cache retrieval policies, by means of trace-driven simulations they measure the maximum feasible hit ratio and byte hit ratio. They suggested that it would be much better to replace documents based on the size as this maximizes the hit ratio in each of their workloads. Cao and Irani

[12] introduced Greedy-Dual-Size (GDS) cache replacement algorithm. This algorithm integrates locality along with cost and size factors. Greedy-Dual- Size tags a cost with every object and expels the object that has the lowest cost or size. Wooster and Abrams[11] proposed Hybrid cache replacement algorithm make use of a combination of multiple requirements such as maintaining in the cache documents from servers that take significant time to connect to, those that need to be fetched from the slowest links, those that have been accessed very frequently, and those that are small. Wooster and Abrams checked the performance of Hybrid algorithm alongside LRU, LFU and SIZE. Hybrid algorithm performed well when compared with traditional LRU, LFU and SIZE replacement algorithms. The Lowest Relative Value (LRV) cache replacement algorithm proposed by Rizzo and Vicisano [10] expels the object that has the lowest utility value. In LRV, the utility of a document is calculated adaptively on the basis of data readily available to a proxy server. Rizzo and Vicisano show that LRV performs better than LRU and can substantially better the performance of a cache that is of modest size.

C. Improved Web proxy caching algorithms

A.P. Foong, H. Yu-Hen, D.M. Heisey [15] proposed a logistic regression model (LR) to predict the future request. Then, the objects with the lowest re- access probability value were replaced first regardless of cost and size of the predicted object. T. Koskela, J. Heikkonen, K. Kaski [14] used Multilayer perceptron network (MLP) classifier in Web caching to predict the class of Web objects depending on syntactic features from HTML structure of the document and the HTTP responses of the server as inputs of MLP. The class value was integrated with LRU, known to be LRU-C, to optimize the Web cache. However, frequency factor was ignored the frequency factor in Web cache replacement decision. An integrated solution of back- propagation neural network (BPNN) as caching decision policy and LRU technique as replacement policy for script data object has been proposed by Farhan [13]. Recently W. Ali, S.M. Shamsuddin, A.S. Ismail [5] proposed three algorithms

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

namely SVM– LRU, SVM–GDSF and C4.5–GDS which make use of the capability of Support vector machine (SVM) and decision tree (C4.5) to learn from Web proxy logs files and predict the classes of objects to be re-visited or not. The trained SVM and C4.5 classifiers were incorporated with traditional Web proxy caching algorithms to present new Web proxy caching approaches. They proved that their proposed SVM– LRU, SVM–GDSF and C4.5–GDS significantly improved the performances of LRU, GDSF and GDS respectively in terms of hit ratio and byte hit ratio.

D. Machine learning Techniques

Machine learning, a branch of artificial intelligence, concerns with the construction and study of systems that can learn from data. For example, a machine learning system could be trained on email messages to learn to distinguish between the spam and the non-spam messages. After learning, it can then be used to classify new email messages into spam and non- spam and send them to respective folders. The core of machine learning deals with representation and generalization. The two areas Machine learning and data mining overlap in many ways: data mining uses many machine learning techniques, but often with a slightly different goal in mind. On the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy.

1) Naïve Bayes(NB) Classifier

Bayesian classifiers can predict class membership probabilities, such as the probability that a given tuple belongs to a particular class. Bayesian classification is based on Bayes' theorem. Simple Bayesian classifier known as the naive Bayesian classifier proved to be comparable in performance with decision tree selected neural network classifiers and some other classification algorithms. These classifiers have also exhibited high accuracy and speed when applied to large databases.

Bayes' theorem is named after Thomas Bayes. Let **X** be a data tuple. In Bayesian terms, **X** is considered "evidence." It is described by measurementsmade on a set of *n* attributes. Let H be some hypothesis,such as that the data tuple **X** belongs to a specified class

C. For classification problems, we want to determine **P (H|X)**, the probability that the hypothesis **H** holds given the "evidence" or observed data tuple **X**. In other words, the probability that tuple **X** belongs to class **C**, given that we know the attribute description of **X**. **P (H|X)** is the posterior probability, or a posteriori probability, of **H** conditioned on **X**. Bayes' theorem is useful in that it provides a way of calculating the posterior probability, **P (H|X)**, from **P (H)**, **P (X|H)**, and **P(X)**.

Baye's theorem is given as

$$\mathbf{P(H|X) = P(X|H) / P(X)}$$

Usually, Bayesian classifiers have the minimum error rate in comparison to all other classifiers. These classifiers are also useful in that they provide a theoretical justification for other classifiers that do not explicitly use Bayes' theorem.

III. THE PROPOSED WEB PROXY CACHING ALGORITHMS

The proposed two algorithms namely NB–SIZE and NB–Hybrid make use of the capability of Naïve Bayes (NB) classifier to learn from Web proxy logs files and predict the classes of objects to be re-visited ornot. The trained Naïve Bayes (NB) classifier was incorporated with traditional Web proxy caching algorithms to present new Web proxy caching approaches. Training and testing was done on Web proxy logs files (datasets) offline which can be used to predict the classes of objects to be re-visited or not in future. In order to prepare the training dataset, the desired features of Web objects are extracted from traces and proxy access log files. The important features of Web objects that indicate the user interest are extracted for preparing the training dataset. These features consist of URL ID, timestamp, elapsed time, size and type of Web object. The common features wereselected and extracted as suggested by W. Ali et al [5]. Subsequently, these features are converted to theinput/output dataset or training patterns in the format <x1, x2, x3, x4, x5, y>. x1, …, x5 represent the inputs and y represents target output of the requested object. Table 1 shows the inputs and their meanings for each training pattern.

Table I: the inputs and their meanings

Input	**Meaning**
x1	Elapsed time of Web object
x2	Frequency of Web object
x3	Recency of Web object
x4	Size of Web object
x5	URL ID of Web object

A. NB–SIZE

Abrams et al. [9] suggested that it would be much better to replace documents based on the size as this maximizes the hit ratio in each of their workloads. As SIZE policy is replace the largest object(s) from a cache when space is needed for a new object. Thus, a cache can be polluted with small objects which will not be accessed again. Therefore, the Naïve Bayes (NB) classifier is integrated with SIZE for improving the performance in terms of the hit ratio of SIZE. The proposed proxy caching policy is called NB–SIZE. In NB–SIZE, a trained Naïve Bayes (NB) classifier is usedto predict the classes of Web objects either objects may be re-visited later or not. After this, the classification decision is integrated into cache replacement policy (SIZE) to give a value (unchanged/decreased) for each object in the cache. Consequently, the objects with the maximum size are removed first, there by postponing the removal of an object based on common factors of web object.

B. NB–HYBRID

Wooster and Abrams[11] proposed Hybrid cache replacement algorithm make use of a combination of multiple requirements such as maintaining in the cache documents from servers that take significant time to connect to, those that need to be fetched from the slowest links, those that have been accessed very frequently, and those that are small. Wooster and Abrams checked the performance of Hybrid algorithm alongside LRU, LFU and SIZE. Hybrid algorithm performed well when compared with traditional LRU, LFU and SIZE replacement algorithms. Though, Hybridalgorithm takes into consideration multiple factors of a web object, the capability of Naïve Bayes (NB) will boost the performance of the Hybrid algorithm. Therefore, the Naïve Bayes (NB) classifier is integrated with Hybrid algorithm for improving the performancein terms of the hit ratio of Hybrid algorithm. The proposed proxy caching policy is called NB–Hybrid. In NB–Hybrid, a trained Naïve Bayes (NB) classifier is used to predict the classes of Web objects either objects may be re-visited later or not. After this, the classification decision is integrated into cache replacement policy (Hybrid) to give a value (unchanged/decreased) for each object in the cache. Consequently, the objects with the maximum size and less frequently visited were removed first, there by postponing/proponing the removal of an object based on common factors of web object.

IV. IMPLEMENTATION AND EXPERIMENTAL RESULTS

A. Raw data collection

The proxy logs files and traces of the Web objects were collected from [7]. We have collected three different datasets namely ClarkNet-HTTP, NASA-HTTP, and Saskatchewan HTTP servers contains of all HTTP requests for three different serversrespectively. ClarkNet-HTTP traces were collectedfrom 04 September, 1995 to 10 September, 1995, NASA-HTTP traces were collected from 01 July, 1995 to 31 July, 1995 and Saskatchewan-HTTP traces were collected from 01 June, 1995 to 31 December, 1995. Timestamps have 1 second resolution.

B. Data Pre-processing

Data pre-processing is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Analyzing the data that has not been carefully screened for such problems may produce misleading results. Thus, the representation and quality of data is the first step before running an analysis. Data pre-processing includes cleaning, normalization, transformation, feature extraction and selection, etc. The product of data pre-processing is the final training set.

C. Training Phase
Proxy log traces were preprocessed and training data sets were prepared based on the format requirement of the simulators. Each proxy dataset is trained and tested with defaults. Naïve Bayes (NB) was implemented using RapidMiner version 5.3.013. The default values of parameters and settings are used as determined in RapidMiner. After training and verification, the trained classifiers were saved in the files which were utilized in improving the performance of the traditional Web proxy caching policies.

D. Web proxy cache simulation
The simulator software for non-uniform size web document caches was provided by University of Wisconsin [16]. The simulator can simulate LRU, SIZE, LRV, Hybrid and variations of Greedy Dual algorithms. The trained classifiers are integrated with simulator software to simulate the proposed Web proxy caching policies. The simulator takes input a text file describing each HTTP requests, calculates the hit ratio and byte hit ratio under an infinite-sized cache, and thencalculates the hit ratio and byte hit ratio for each algorithm, under cache sizes being various percentages of the total data set size.

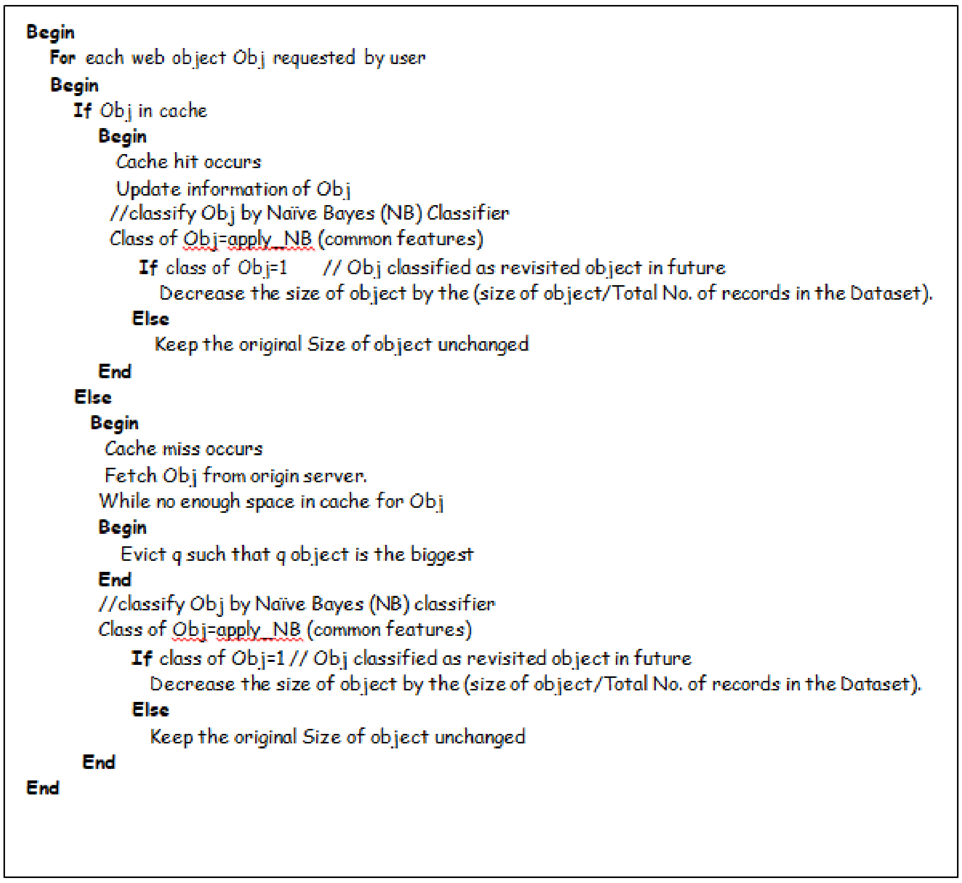
V. PERFORMANCE EVALUATION

A. Classifier evaluation
Three different performance measurements were used for evaluating the model/classifier. Table 4 shows each measure name and formula to calculate the same. Table 3 shows the values of performance measures of testing datasets. A correct classificationrate (CCR) is a measure for evaluating a model or classifier. The true positive rate (TPR) or sensitivity, the true negative rate (TNR) or specificity also used to evaluate the performance of machine learning techniques. Table 2 shows the Confusion matrix.

```
Begin
   For each web object Obj requested by user
   Begin
      If Obj in cache
         Begin
           Cache hit occurs
           Update information of Obj
          //classify Obj by Naïve Bayes (NB) Classifier
          Class of Obj=apply_NB (common features)
              If class of Obj=1      // Obj classified as revisited object in future
                Decrease the size of object by the (size of object/Total No. of records in the Dataset).
                Decrease the latency of object by the probability of frequency of object.
             Else
                Keep the original Size of object unchanged
         End
      Else
        Begin
          Cache miss occurs
          Fetch Obj from origin server.
         While no enough space in cache for Obj
         Begin
            Evict q such that q object is the biggest
         End
         //classify Obj by Naïve Bayes (NB) classifier
         Class of Obj=apply_NB (common features)
             If class of Obj=1 // Obj classified as revisited object in future
               Decrease the size of object by the (size of object/Total No. of records in the Dataset).
               Decrease the latency of object by the probability of frequency of object.
             Else
               Keep the original Size of object unchanged
       End
End
```

Fig. 2: The proposed NB-HYBRID algorithm

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7



```
Begin
  For each web object Obj requested by user
  Begin
    If Obj in cache
      Begin
        Cache hit occurs
        Update information of Obj
       //classify Obj by Naïve Bayes (NB) Classifier
       Class of Obj=apply_NB (common features)
          If class of Obj=1      // Obj classified as revisited object in future
            Decrease the size of object by the (size of object/Total No. of records in the Dataset).
          Else
            Keep the original Size of object unchanged
      End
    Else
      Begin
        Cache miss occurs
        Fetch Obj from origin server.
       While no enough space in cache for Obj
       Begin
         Evict q such that q object is the biggest
       End
       //classify Obj by Naïve Bayes (NB) classifier
       Class of Obj=apply_NB (common features)
          If class of Obj=1 // Obj classified as revisited object in future
            Decrease the size of object by the (size of object/Total No. of records in the Dataset).
          Else
            Keep the original Size of object unchanged
     End
End
```

Fig. 3: The proposed NB-SIZE algorithm

Table II: Confusion matrix

	Predicted positive	**Predicted negative**
Actual positive	True positive (TP)	False negative (FN)
Actual negative	False positive (FP)	True negative (TN)

Table Iii: the performance measures of testing data (in %).

		Clarknet	**NASA**	**Saskatchewan**	**Average**
Naïve Bayes	**CRR**	98.88	95.80	93.36	96.01
	TPR	98.71	99.38	96.26	98.12
	TNR	80.40	81.81	81.67	81.29

Table IV: the measures used for evaluatingperformance of naïve bayes (nb) algorithm

Measure name	**Formu la**
Correct classification rate	**(%)**
True positive rate	**(%)**
True negative rate	**(%)**

B. Evaluation of proposed Web proxy cachingapproaches

1) Performance measures

In Web proxy caching, hit ratio (HR) and byte hit ratio (BHR) are two widely used metrics for evaluating the performance of Web proxy caching policies [14]. HR is defined as the ratio of the numberof requests served from the proxy cache and the total number of requests. BHR refers to the number of bytes served from the cache, divided by the total number of bytes served. Usually HR and BHR work in opposite ways. The following table shows the Pure hit-rate, Byte hit-rate and Reduced latency for existing and proposed algorithms. Table 5 shows the Pure hit-rate, Byte hit- rate and Reduced latency for existing and proposed algorithms. Graph of the interpreted data can be seen in figures 4, 5, 6.

VI. CONCLUSION

Experimental results have revealed that the proposed two algorithms NB–SIZE and NB–Hybrid significantly increased Pure Hit-Ratio, Byte Hit-Ratio and reduced the latency when compared with SIZE and Hybrid. The same is evident from the below shown graphs for NASA dataset.

Table V: pure hit-rate, byte hit-rate and reduced latency for existing and proposed algorithms for nasa server traces.

Algorithm	**Cache Size (%)**	**Pure Hit-Rate**	**Byte Hit-Rate**	**Reduced Latency**
SIZE	0.05%	0.03583	0.002296	0.036163
	0.50%	0.118443	0.024544	0.118963
	5.00%	0.383321	0.183968	0.383685
	10.00 %	0.516043	0.305841	0.516329
	20.00 %	0.677655	0.498402	0.677845
NB-SIZE	0.05%	0.078145	0.011233	0.077592
	0.50%	0.263749	0.095857	0.262866
	5.00%	0.706094	0.535683	0.70623
	10.00 %	0.877198	0.785813	0.87749

	20.00%	0.996109	0.992499	0.996118
Hybrid	0.05%	0.017389	0.003614	0.01733
	0.50%	0.110464	0.038888	0.110086
	5.00%	0.353239	0.20589	0.352029
	10.00%	0.473421	0.305478	0.472528
	20.00%	0.612032	0.45123	0.611623
NB-Hybrid	0.05%	0.057608	0.01548	0.057747
	0.50%	0.252399	0.136547	0.253008
	5.00%	0.635816	0.479554	0.637191
	10.00%	0.809611	0.706353	0.810329
	20.00%	0.986633	0.976403	0.986683

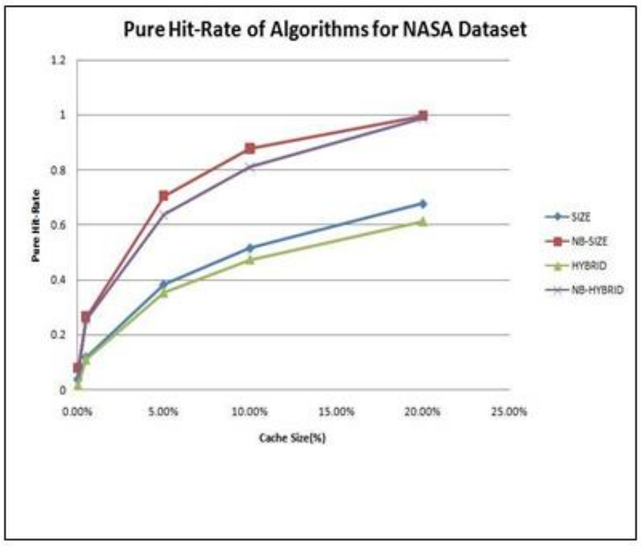


Fig. 4: Pure Hit-Rate graph for existing and proposed Algorithms

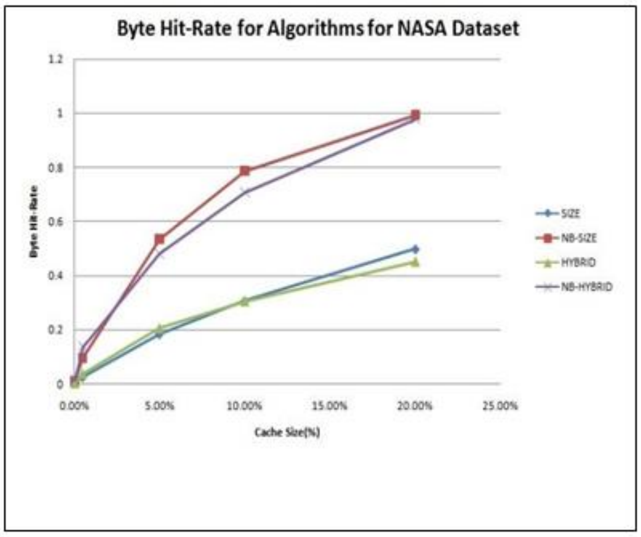


Fig. 5: Byte Hit-Rate graph for existing and proposed Algorithms

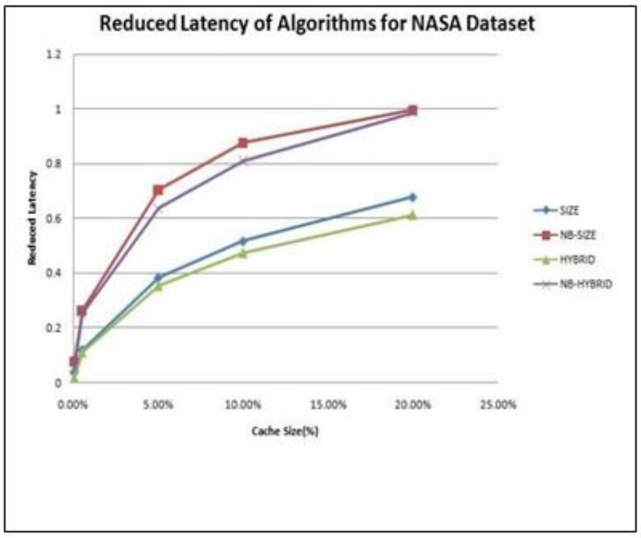


Fig.6: Reduced Latency graph for existing and proposed Algorithms

REFERENCES

[1] H.T. Chen, Pre-Fetching and Re-Fetching in Web Caching Systems: Algorithms and Simulation, Trent University, Peterborough, Ontario, Canada, Peterborough, Ontario, Canada, 2008.
[2] S. Romano, H. ElAarag, A neural network proxy cache replacement strategy and its implementation in the Squid proxy server, Neural Computing and Applications 20 (2011) 59–78.
[3] W. Kin-Yeung, Web cache replacement policies: a pragmatic approach, IEEE Network 20 (2006) 28–34.
[4] C. Kumar, J.B. Norris, A new approach for a proxy- level web caching mechanism, Decision Support Systems 46 (2008) 52–60.
[5] W. Ali, S.M. Shamsuddin, A.S. Ismail, Intelligent Web proxy caching approaches based on machine learning techniques, Elsevier (2012) 0167-9236.
[6] Caruana, R.; Niculescu-Mizil, A. (2006). "An empirical comparison of supervised learning algorithms". Proceedings of the 23rd international conference on Machine learning.
[7] The University of California and Lawrence Berkeley National Laboratory, Traces: Available at http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html

[8] Sulaiman, S.; Shamsuddin, S.M.; Forkan, F.; Abraham, A. "Intelligent Web Caching Using Neurocomputing and Particle Swarm Optimization Algorithm" IEEE Modeling & Simulation, 2008. AICMS 08. Second Asia International Conference, Page(s): 642 - 647, Print ISBN: 978-0-7695-3136-6, 13- 15 May 2008.
[9] M. Abrams, C.R. Standridge, G. Abdulla, E.A. Fox,
[10] S. Williams, Removal Policies in Network Caches for World-Wide Web Documents, ACM, 1996, pp. 293– 305.
[11] Rizzo, Vicisano, Replacement Policies for a Proxy Cache, IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 8, NO. 2, APRIL 2000
[12] Roland P. Wooster and Marc Abrams: Proxy Caching That Estimates Page Load Delays from: Computer Networks and Isdn Systems - CN, Vol. 29, No. 8-13, pp. 977-986, 1997.
[13] Pei Cao and Sandy Irani, Cost-aware www proxy caching algorithms. In Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, California, December 1997.
[14] J. Cobb, H. ElAarag, Web proxy cache replacement scheme based on back-propagation neural network, Journal of Systems and Software 81 (2008) 1539–1558.
[15] T. Koskela, J. Heikkonen, K. Kaski, Web cache optimization with nonlinear model using object features, Computer Networks 43 (2003) 805–817.
[16] A.P. Foong, H. Yu-Hen, D.M. Heisey, Logistic regression in an adaptive Web cache, IEEE Internet Computing 3 (1999) 27–36.
[17] Copyright 1997. University of Wisconsin – Madison. All Rights Reserved.