Intelligent Web Caching Using Adaptive Regression Trees, Splines, Random Forests and Tree Net

Sarina Sulaiman, Siti Mariyam Shamsuddin
Soft Computing Research Group
Faculty of Computer Science and Information Systems
Universiti Teknologi Malaysia, Johor, Malaysia
sarina@utm.my, mariyam@utm.my

Ajith Abraham
Machine Intelligence Research Labs (MIR Labs)
Washington 98092, USA
http://www.mirlabs.org
ajith.abraham@ieee.org

Abstract— **Web caching is a technology for improving network traffic on the internet. It is a temporary storage of Web objects (such as HTML documents) for later retrieval. There are three significant advantages to Web caching; reduced bandwidth consumption, reduced server load, and reduced latency. These rewards have made the Web less expensive with better performance. The aim of this research is to introduce advanced machine learning approaches for Web caching to decide either to cache or not to the cache server, which could be modelled as a classification problem. The challenges include identifying attributes ranking and significant improvements in the classification accuracy. Four methods are employed in this research; Classification and Regression Trees (CART), Multivariate Adaptive Regression Splines (MARS), Random Forest (RF) and TreeNet (TN) are used for classification on Web caching. The experimental results reveal that CART performed extremely well in classifying Web objects from the existing log data and an excellent attribute to consider for an accomplishment of Web cache performance enhancement.**

Keywords-component; Data mining; Web caching; classification

I. Introduction

Caching operation can be executed at the client application, and is generally it is embedded in most Web browsers. There are a number of products that extend or replace the embedded caches with systems that contain larger storage, more features, or better performance. In any cases, these systems only cache net objects from many servers for a single user.

Caching can also be operated between the client and the server as a part of proxy cache, which is often located close to network gateways to decrease the bandwidth connections. These systems can serve many users (clients) with cached objects from many servers. In fact, the usefulness of web caching (reportedly up to 80% for some installations) is in caching objects requested by one client for later retrieval by another client. Even for better performance, many proxy caches are part of cache hierarchies; a cache can appeal neighbouring caches for a requested document to lessen the need for direct fetching.

Furthermore, caches can be situated directly in front of a particular server in order to reduce the number of requests that the server must handle. Most proxy caches can be used in this fashion with different names; reverse cache, inverse cache, or sometimes httpd accelerator, to replicate the fact that it caches objects for many clients but normally from one server [1]. This paper investigates the performance of Classification and Regression Trees (CART), Multivariate Adaptive Regression Splines (MARS), Random Forest (RF) and TreeNet (TN) for classification in Web caching.

The rest of the paper is organised as follows: Section 2 describes the related works, followed by introduction about the machine learning approaches in Section 3. Section 4 on experimental setup and Section 5 illustrates the performance evaluation of the proposed approaches. Section 6 discusses the result from the experiment and finally, Section 7 concludes the article.

II. Related Works

Many researchers have looked for ways to improve current caching techniques. Padmanabhan and Mogul [2] proposed predictive model to be used as server hint. The proposed model is equipped with server that is able to create Markov model by predicting probability of object A will be tag along, with next n requests and object B (n is a parameter of the algorithm). The server will use the model to produce a forecast for subsequent references and throw the forecast to the client. The client will use forecast result to pre-fetch an object on the server if only that object is not in the client cache. The simulation done was able to reduce the latency until 45%. However, their technique has a limitation as it also makes network traffic larger by two times [2]. That is why a lot of people try to find latency and network reduction at the same time.

Bestavros and Cunha [3] have presented a model for the speculative dissemination of World Wide Web data. His work illustrates that reference patterns from a Web server can be used as a key information source for pre-fetching. They also investigate that latency reduction increases until 50%, though it still increases the bandwidth utilisation.

On the other hand, Pallis et al. [4] has proposed a pre-fetching based on the clustering method. Web pre-fetching is an attractive solution to reduce the network resources consumed by Web services as well as the access latencies perceived by Web users. Unlike Web caching, which exploits the temporal locality, Web pre-fetching utilises the spatial locality of Web objects. Specifically, Web pre-fetching fetches objects that are likely to be accessed in the near future and stores them in advance. In this context, a sophisticated combination of these two techniques may cause significant improvements on the performance of the Web infrastructure.

Kroeger et al. [5] observed a local proxy caching which is able to decrease latency until 26%, while pre-fetching could decrease latency at 57%. The combination of both of them will give better latency reduction until 60%. Furthermore, he had also found that algorithm on pre-fetching has contribution on reducing latency. From his work, it also explained that pre-fetching can provide double improvement on caching. However, it is only for decreasing the latency.

Xu et al. in [6] proposed solutions by creating proxy management. The Caching dynamic content is obtained by generating data and personalise data that contributes up to 30-40% of the total traffic. These types of data are normally identified as "uncachable". To further improve Web performance, reverse caching has also been suggested to make more dynamic content cachable and manageable [4]. Xu et al. [6] also proposed collaboration among proxies is based on the premise that it would be faster and cheaper to fetch an object from another close proxy rather than the origin server. See Fig. 1 for the cooperative cache organization. However, the challenge still stuck on how to efficiently maintain consistency between the cached content and the data source that frequently change. Another important issue is the analysis of query semantics to evaluate a complex query over the cached content.

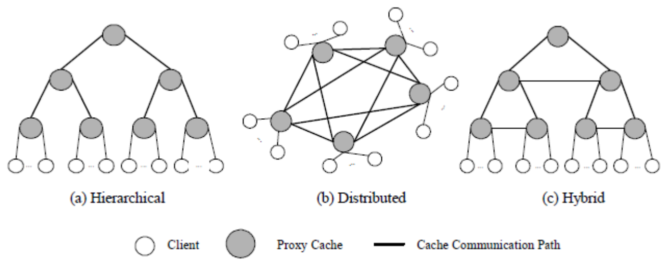


Figure 1. Examples of different cooperative cache organizations. [6]

Caching streaming objects: It is predicted that streaming media such as music or video clips will symbolize a significant portion of Web traffic over the Internet. Due to the distinct features of streaming objects like big size, long duration, intensive use of bandwidth, and interactivity, conventional proxy caching techniques are not able to solve this problem. To solve these problems, many partial caching algorithms have been proposed in recent years [7,8]. The proposed algorithms expressed that even if small size of video is stock up on the proxy, the consumption of network will be reduced significantly.

Teng et al., in [9] proposed a combination between Web caching and Web pre-fetching. These two techniques can go together since the Web caching technique use the temporal locality while Web pre-fetching technique utilizes the spatial locality of Web objects. The proposed technique is obtained by evaluating the pre-fetching rules.

Lots of future works have been drawn by previous researchers especially on clustering pre-fetching, caching on proxy level or even designing cache organization. Considering that there have been several caching policies proposed in the past, the challenge is to extend them by using data mining techniques. It presented a clustering-based pre-fetching scheme where a graph-based clustering algorithm identifies clusters of ''correlated'' Web pages based on the users' access patterns and to create adaptive websites [10].

Nevertheless, this research has proposed a scheme that can be realise to integrate data mining techniques into a cache server for Web object classification thus improving its performance. Through a simulation environment, using a real data set, CART, MARS, RF and TN can be an effective way in improving the performance of the Web caching environment.

In this research, real data set were used for classification of Web object data based on two different Web log data; Boston University (BU) and E-learning@UTM (EL). According to the related work, the issues of performance to classify the Web objects and implementation on cache server were highlighted.

III. Machine Learning Approaches

3.1. CART

Classification and Regression Trees (CART) is a robust decision-tree tool for data mining, pre-processing and predictive modelling, suggested by Breiman et al. [11]. CART can be used by complex data for patterns and relationships and uncovering a hidden structure [12]. Moreover, it is a nonparametric technique that can select from among a large number of variables, and their interactions that are most important in determining the outcome variable to be explained.

Decision Tree (DT) induction is one of the classification algorithms in data mining. The classification algorithm is inductively learned to construct a model from the pre-classified data set. Inductive learning means making general assumptions from the specific examples in order to use those assumptions to classify unseen data. The inductively learned model of classification algorithm is known as classifier. Classifier may be viewed as mapping from a set of attributes to a particular class. Data items are defined by the values of their attributes and X is the vector of their values $\{x_1, x_2 \ldots x_n\}$, where the value is either numeric or nominal. Attribute space is defined as the set containing all possible attribute vectors and is denoted by Z. Thus X is an element of Z ($X \in Z$). The set of all classes is denoted by $C = \{c_1, c_2, \ldots, c_n\}$. A classifier assigns a class $c \in C$ to every attribute of the vector $X \in Z$. The classifier can be considered as a mapping f, where f: $X \rightarrow C$. This classifier is used to classify the unseen data with a class label. A decision tree classifies the given data item using the values of its attributes. The decision tree is initially constructed from a set of pre-classified data. Each data item is defined by values of the attributes.

The main issue is to select the attributes which best divides the data items into their classes. According to the values of these attributes the data items are partitioned. This process is recursively applied to each partitioned subset of the data items.

The process terminates when all the data items in the current subset belongs to the same class.

A decision tree consists of nodes, leaves and edges. A node of a decision tree specifies an attribute by which the data is to be partitioned. Each node has a number of edges, which are labelled according to a possible value of edges and a possible value of the attribute in the parent node. An edge connects either two nodes or a node and a leaf. Leaves are labelled with a decision value for categorization of the data. Induction of the decision tree uses the training data, which is described in terms of the attributes. The main problem here is deciding the attribute, which will best partition the data into various classes.

3.2. MARS

Multivariate Adaptive Regression Splines (MARS) model is a spline regression model that uses a specific class of basis functions as predictors in place of the original data [13,14]. The MARS basis function transform makes it possible to selectively blank out certain regions of a variable by making them zero, allowing MARS to focus on specific sub-regions of the data. MARS excels at finding optimal variable transformations and interactions, as well as the complex data structure that often hides in high-dimensional data.

Given the number of predictors in most data mining applications, it is infeasible to approximate a function y=f(x) in a generalization of splines by summarizing y in each distinct region of x. Even if we could assume that each predictor x had only two distinct regions, a database with just 35 predictors would contain 235 or more than 34 billion regions. This is known as the curse of dimensionality. For some variables, two regions may not be enough to track the specifics of the function. If the relationship of y to some x's is different in three or four regions, for example, with only 35 variables the number of regions requiring examination would be even larger than 34 billion. Given that neither the number of regions nor the knot locations can be specified a priori, a procedure is needed that accomplishes the following:

- judicious selection of which regions to look at and their boundaries, and
- judicious determination of how many intervals are needed for each variable.

A successful method of region selection will need to be adaptive to the characteristics of the data. Such a solution will probably reject quite a few variables (accomplishing variable selection) and will take into account only a few variables at a time (also reducing the number of regions).

A key concept underlying the spline is the knot, which marks the end of one region of data and the beginning of another. Thus, the knot is where the behaviour of the function changes. Between knots, the model could be global (e.g., linear regression). In a classical spline, the knots are predetermined and evenly spaced, whereas in MARS, the knots are determined by a search procedure. Only as many knots as needed are included in a MARS model. If a straight line is a good fit, there will be no interior knots. In MARS, however, there is always at least one "pseudo" knot that corresponds to the smallest observed value of the predictor.

In MARS, Basis Functions (BFs) are the machinery used for generalizing the search for knots. BFs are a set of functions used to represent the information contained in one or more variables. Much like principal components, BFs essentially re-express the relationship of the predictor variables with the target variable. The hockey stick BF, the core building block of the MARS model is often applied to a single variable multiple times. The hockey stick function maps variable X to new variable X*:

$$\max(0, X - c), \text{ or } \max(0, c - X)$$

In the first form, X* is set to 0 for all values of X up to some threshold value c and X* is equal to X for all values of X greater than c (actually X* is equal to the amount by which X exceeds threshold c). The second form generates a mirror image of the first. It starts with a constant in the model and then begins the search for a variable-knot combination that improves the model the most (or, alternatively, worsens the model the least). The improvement is measured in part by the change in Mean Squared Error (MSE). Adding a basis function always reduces the MSE. MARS searches for a pair of hockey stick basis functions, the primary and mirror image, even though only one might be linearly independent of the other terms. This search is then repeated, with MARS searching for the best variable to add given the basis functions already in the model. The brute search process theoretically continues until every possible basis function has been added to the model.

In practice, the user specifies an upper limit for the number of knots to be generated in the forward stage. The limit should be large enough to ensure that the true model can be captured. A good rule of thumb for determining the minimum number is three to four times the number of basis functions in the optimal model. This limit may have to be set by trial and error.

3.3. RF

The Random Forests (RF) algorithm was proposed by Leo Breiman in 1999[15]. The algorithm can be used for both regression and classification, as well as for variable selection, interaction detection, clustering etc. This technology represents a substantial advance in data mining technology and it is based on novel ways of combining information from a number of decision trees [11] [15].

A Decision Tree Forest (DTF) is an ensemble (collection) of decision trees whose predictions are combined to make the overall prediction for the forest. A decision tree forest grows a number of independent trees in parallel, and they do not interact until after all of them have been built. Decision tree forest models often have a degree of accuracy that cannot be obtained using a large, single-tree model. An outline of the algorithm used to construct a decision tree forest consisting of N observations is given below:

(1) Take a random sample of N observations from the data set with replacement. Some observations will be selected more than once, and others will not be selected. On average, about 2/3 of the rows will be selected by the sampling. The

remaining 1/3 of the rows are called the out of bag rows. A new random selection of rows is performed for each tree constructed.

(2) As the tree is built, allow only a subset of the total set of predictor variables to be considered as possible splitters for each node. Select the set of predictors to be considered as a random subset of the total set of available predictors. For example, if there are ten predictors, choose a random five as candidate splitters. Perform a new random selection for each split. Some predictors (possibly the best one) will not be considered for each split, but a predictor excluded from one split may be used for another split in the same tree.

(1) and (2) are repeated a large number of times to construct a forest of trees.

Decision tree forests have two stochastic elements: (1) the selection of data rows used as input for each tree, and (2) the set of predictor variables considered as candidates for each node split. For reasons that are not well understood, these randomizations along with combining the predictions from the trees significantly improve the overall predictive accuracy.

3.4. TN

TreeNet (TN) is a robust multi-tree technology for data mining, predictive modelling and data processing. This technology is an exclusive implementation of Jerome Friedman's MART methodology [16]. It offers exceptional accuracy, blazing speed, and a high degree of fault tolerance for dirty and incomplete data. It can handle both classification and regression problems and has been proven to be remarkably effective in traditional numeric data mining and text mining [16].

TN is an enhancement of the CART model using stochastic gradient boosting [16]. Boosting means that endeavours to "boost" the accuracy of any given learning algorithm by fitting a series of models each having a low error rate and then combining into an ensemble that may perform better [18, 17]. The key features of TN models consist of [18]: automatic variable subset selection; ability to handle data without pre-processing; resistance to outliers; automatic handling of missing values; robustness to dirty and partially inaccurate data; high speed; and resistance to over-training. A TN model can be thought of as a series expansion approximating the true functional relationship [19] (1):

$$F(X) = F_0 + \beta_1 T_1(X) + \beta_2 T_2(X) + \ldots + \beta_M T_M(X) \qquad (1)$$

where T_i is a small tree. Each tree refines and improves on its predecessors. TN models are thus typically composed of hundreds of small trees, each of which contributes slight refinement to the overall model.

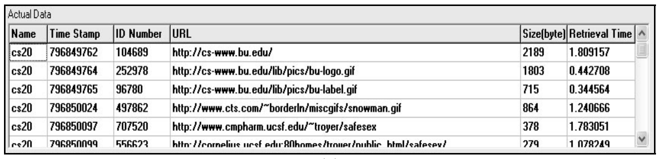
IV. EXPERIMENTAL SETUP

In this experiment, two different log records were utilised. The first data from BU Web Trace (client-side) [20] collected by Oceans Research Group at BU functioned as the experiment data set. BU traces records collected 9,633 files, instead of a population of 762 different users, and recording 109,759 requests for data transfer. The browser log data (from November 1994 to May 1995) were obtained from Mosaic clients at the BU [21][22][23].

The second data was from EL (Web server) from Universiti Teknologi Malaysia (UTM). The server log data that was obtained on 13 and 14 January 2008 with 65,015 records were from one of EL Apache servers at Centre of Information and Communication Technology (CICT) [24].

A. Pre-processing and Normalise Data

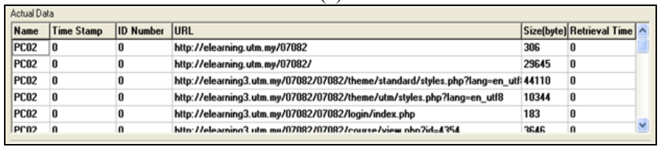
The pre-processing is the key component to classify an object to cache. Fig. 2 shows the actual data prior to data pre-processing, and Fig. 3 depicts the pre-process data for BU, and EL logs data. Each line of a condensed log in BU Web traces corresponds to a single URL requested by the user; it contains the machine name, the time stamp when the request was made (seconds and microseconds), the URL, the size of the document (in bytes) and the object retrieval time in seconds. Detail explanation can be referred to [20][21][24].



Actual Data

Name	Time Stamp	ID Number	URL	Size(byte)	Retrieval Time
cs20	796849762	104689	http://cs-www.bu.edu/	2189	1.809157
cs20	796849764	252978	http://cs-www.bu.edu/lib/pics/bu-logo.gif	1803	0.442708
cs20	796849765	96780	http://cs-www.bu.edu/lib/pics/bu-label.gif	715	0.344564
cs20	796850024	497862	http://www.cts.com/~borderln/miscgifs/snowman.gif	864	1.240666
cs20	796850097	707520	http://www.cmpharm.ucsf.edu/~troyer/safesex	378	1.783051
cs20	796850099	556623	http://cornelius.ucsf.edu:80homes/troyer/public_html/safesex/	279	1.078249

(a)

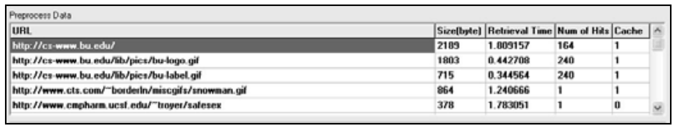


Actual Data

Name	Time Stamp	ID Number	URL	Size(byte)	Retrieval Time
PC02	0	0	http://elearning.utm.my/07082	306	0
PC02	0	0	http://elearning.utm.my/07082/	29645	0
PC02	0	0	http://elearning3.utm.my/07082/07082/theme/standard/styles.php?lang=en_utf	44110	0
PC02	0	0	http://elearning3.utm.my/07082/07082/theme/utm/styles.php?lang=en_utf8	10344	0
PC02	0	0	http://elearning3.utm.my/07082/07082/login/index.php	183	0
PC02	0	0	http://elearning3.utm.my/07082/07082/course/view.php?id=4354	3646	0

(b)

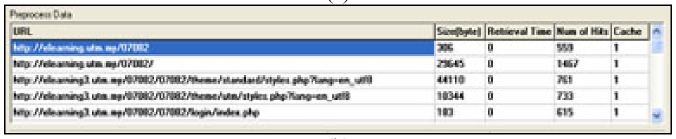
Figure 2. Examples data from BU and EL log data: (a) BU (b) EL.



Preprocess Data

URL	Size(byte)	Retrieval Time	Num of Hits	Cache
http://cs-www.bu.edu/	2189	1.809157	164	1
http://cs-www.bu.edu/lib/pics/bu-logo.gif	1803	0.442708	240	1
http://cs-www.bu.edu/lib/pics/bu-label.gif	715	0.344564	240	1
http://www.cts.com/~borderln/miscgifs/snowman.gif	864	1.240666	1	1
http://www.cmpharm.ucsf.edu/~troyer/safesex	378	1.783051	1	0

(a)



Preprocess Data

URL	Size(byte)	Retrieval Time	Num of Hits	Cache
http://elearning.utm.my/07082	306	0	559	1
http://elearning.utm.my/07082/	29645	0	1467	1
http://elearning3.utm.my/07082/07082/theme/standard/styles.php?lang=en_utf8	44110	0	761	1
http://elearning3.utm.my/07082/07082/theme/utm/styles.php?lang=en_utf8	10344	0	733	1
http://elearning3.utm.my/07082/07082/login/index.php	183	0	615	1

(b)

Figure 3. Pre-process BU and EL log data: (a) BU (b) EL.

Meanwhile, each line in EL file represents an incoming HTTP request, and Apache records information about it using a format known as the Common Log Format (CLF). Reading from left to right, this format contains the following information about the request; the source IP address, the client's identity the remote user name (if using HTTP authentication), the date, time, and time zone of the request, the actual content of the request, the server's response code to the request, and the size of the data block returned to the client, in bytes.

Three common variables or attributes have been identified in Web performance analysis [25][26]. The attributes used in this study are:

Web Object Size: *the size is expressed in bytes and kilobytes.*
Numbers of Hits: *the number of hits per data. Each completed request for a Web file will increase the number of hit for requested file.*
Retrieval Time: *the counter that observes the time takes to receive a data in seconds.*

Each variable or attribute must be multiplied with defined Priority Value (PV) [27] to get the total of the attributes for target output generation of the network. Equation (2) explains an example of PV calculation:

$$\text{Expected target} = (\text{Size} * 0.266667) + (\text{Num\_of\_hits} * 0.200000) + (\text{Retrieval\_time} * 0.066667) \qquad (2)$$

The total value determines the expected target for current data. The total value is compared to a threshold number, and this threshold values are dynamic. A new threshold calculation is proposed based on the latency ratio on singular hit rate data [28].

The threshold is calculated and updated for every epoch of the training (3). If the *expected_target* is smaller than the threshold, then the expected target would be 0, or else it becomes 1 if the *expected_target* is equal or greater than to the threshold [21][24] shown below:

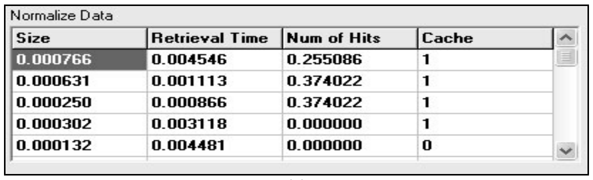
$$\textit{Expected Network Output} = \begin{cases} 0 & \text{if expected\_target} < \textit{threshold}, \\ 1 & \text{if expected\_target} \geq \textit{threshold}. \end{cases} \qquad (3)$$

The network incorporates simplicity in generating output for the Web caching to cache or not to cache. For each output generated from the non-training mode, the outputs can be illustrated by employing sigmoid function that bounded between 0 and 1. For each output values that represents between the interval of [0.5,1], the data will be cached in the caching storage, and for each output that represents values less than 0.5 the data will be fetched directly from the originating database resource in case the data is not found in the cache storage [21][24].

Normalisation process (see Fig. 4) is done by determining the maximum and minimum value for each attribute [21][24]. The end values are between 0 and 1 to improve training characteristics. Min-max normalisation is given as in (4):

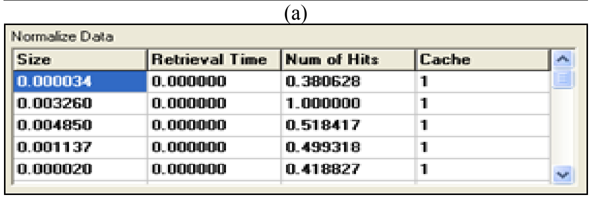
$$X^* = \frac{X - \min(X)}{\max(X) - \min(X)} \qquad (4)$$

Let X refers to an original attribute value and X^* refers to the normalised attribute value. From this formula, we can summarise the results as shown in Table 1. The summarisation shows the normalisation process had reduced the number of data up to 15.69% and 42.31% for BU and EL data set respectively at a proxy cache.


Normalize Data

Size	Retrieval Time	Num of Hits	Cache
0.000766	0.004546	0.255086	1
0.000631	0.001113	0.374022	1
0.000250	0.000866	0.374022	1
0.000302	0.003118	0.000000	1
0.000132	0.004481	0.000000	0

(a)


Normalize Data

Size	Retrieval Time	Num of Hits	Cache
0.000034	0.000000	0.380628	1
0.003260	0.000000	1.000000	1
0.004850	0.000000	0.518417	1
0.001137	0.000000	0.499318	1
0.000020	0.000000	0.418827	1

(b)

Figure 4. Normalise BU and EL log data: (a) BU (b) EL.

TABLE I. SUMMARISATION OF BU AND EL LOG DATA AFTER THE NORMALISATION

Summary	BU	EL
Number of actual data	109 759	65 015
Number of pre-process data	17 224	23 105
Maximum size (byte)	16 711 260	16 384 015
Longest retrieval time (seconds)	1 749	0
Highest hits	3 328	1 141

B. Training and testing

The actual BU log data consists of 109,759 records and EL log data involves of 65,015 records. Moreover, 70% of it was used for training and the remaining for testing purposes. These experiments were carried out on a Core Duo CPU, 2GHz Machine and the codes were executed using Salford System tools.

The details of model statistics for CART, MARS, RF and TN by using the BU and EL data are shown below:

CART:
Number of predictors = 3, 2
Important = 3, 2
Nodes = 38, 5
Min node cases = 2, 312
MARS:
Number of predictors = 3, 2
Basis functions = 3, 1
Number of effective parameters = 7, 1
Min observation between knots = 0, 0
RF:
Number of predictors = 3, 2
Max terminal nodes = 8613, 8088
Trees grown = 500, 500
Parent node min cases = 2, 2
TN:
Number of predictors = 3, 2
Tree size = 6, 6
Tree grown = 200, 200
Last tree data fraction = 0.03, 0.44

V. Performance and Results Achieved

Table 2 summarises the comparative performances of the different CART, MARS, RF and TN in terms of error rate training, learning iteration and classification accuracy.

From the training, we find that the error rate training for TN model has resulted as the lowest rate with 0.001 for BU data. Simultaneously, CART and TN model is given 0 error rate for EL data. However, the highest error rate is 0.242 by using MARS model for EL data. The smallest number of training iteration is CART, 38 and 5 for BU and EL data.

TABLE II. Performance Comparison between CART, MARS, RF and TN for BU and EL Log Data

Technique	Error Rate (Training)		Learning Iteration/ Nodes Optimal		Accuracy (%)	
	BU	*EL*	*BU*	*EL*	*BU*	*EL*
CART	0.002	0.000	38	5	99.86	100
MARS	0.081	0.242	-	-	91.07	59.04
RF	0.007	0.003	500	500	99.50	99.45
TN	0.001	0.000	200	200	99.80	100

Testing process is done to determine the accuracy of the output generated by all Salford Systems tools. The accuracy is done base on the difference in results between the actual value and the generated value by CART, MARS, RF and TN. In this study, the accuracy is measured as shown in (5):

$$Accuracy = \frac{\text{Number of correct data}}{\text{Total data}} \times 100\% \qquad (5)$$

Based on this equation, the CART accuracy is 99.86% and 100% for BU and EL data as the best accuracy compare to other models. The second highest is TN, 99.8% (BU) and 100% (EL). It depicts that the CART model seems to classify better and less iteration than others.

Table 3 depicts the receiver operating characteristic (ROC) as a graphical plot of the sensitivity vs. (1 - specificity) for a binary classifier system as its discrimination threshold is varies. The ROC analysis recently has been introduced in various fields like medicine, radiology and others. Conversely, it has been introduced recently in other areas for instance data mining and machine learning. ROC approved that CART model is the best classifier measure up to other Salford Systems model.

TABLE III. ROC Comparison Between CART, MARS, RF and TN for BU and EL Log Data

Technique	ROC			
	BU		EL	
	Class1(0)	*Class2(1)*	*Class1(0)*	*Class2(1)*
CART	1.000	1.000	1.000	1.000
MARS	0.000	1.000	0.000	1.000
RF	0.989	0.996	0.999	0.995
TN	0.989	1.000	1.000	1.000

In addition, Table 4 shows the important level of three variables. However, EL log data only provided size and number of hits variable. The most significant variable is the size, followed by the number of hits and retrieval time for each BU and EL log data. It proves that size will construct an effective use of space for Web caching in the cache server.

TABLE IV. Variable Importance

Variable	Score							
	BU				EL			
	CART	*MARS*	*RF*	*TN*	*CART*	*MARS*	*RF*	*TN*
Size	100	94.17	100	100	100	-	100	100
Num_of _hits	87.50	100	56.85	59.75	26.14	-	25.82	50.20
Retrieval _time	41.19	49.27	13.57	34.51	-	-	-	-

VI. Discussion

In this research, an accomplishment of different machine learning approaches for Web caching technology promises alleviation of congestion of Internet access mainly for BU and EL. Therefore, this study proves that the classification of Web object through log mining by using CART, MARS, RF and TN.

In the literature, various methodology approaches to manage proxy cache have been proposed [2-10]. However, we applied statistical model to decide and classify the object on Web documents either to cache or not cache.

It is clear from the results presented here that CART and TN have a distinct advantage over MARS and RF in classifying the cache objects. This scenario happened related to the strengths and weaknesses of the models themselves. Subsequently, the data set to be modelled also can be considered as a suitability factor of the models.

Based on the experimental results in this study, some remarks can be discussed as follows:

- Intelligent Web caching is able to store ideal objects and remove unwanted objects, which may cause insufficiency cache. Thus, the caching insufficiency can be improved.
- Both CART and TN achieve correct classification accuracy in the range of 99.8% and 100% for testing data of BU and EL log data, and in the range of 0 and 0.002 for training error rate data for both data compared to MARS and RF respectively.
- ROC for CART is the highest sensitivity and specificity for testing. Consequently, CART is identified as the best classifier that is closest to the convex hull.
- In all conditions, MARS was the worst model to apply in classifying all log data because of MARS is highly sensitive to extrapolation caused by the local nature of the basis functions. A change in the predictor value toward the end of its range can cause the prediction to go largely off scale.

- Size as the most important variable is recognized to ensure that intelligent Web caching can be affected the performance of cache server.

VII. Conclusions and Future Work

In this research, an accomplishment of different machine learning approaches for Web caching technology promises alleviation of congestion of Internet access mainly for BU and EL. Therefore, this study proves that the classification of Web object through log mining by using CART, MARS, RF and TN models can be applied in cache server. Hence, this situation will affect the size of data in the cache server and time to retrieve the data from the cache server. In the future, we will evaluate and compare the performance analysis of machine learning approaches with other hybrid soft computing techniques for Web caching technology for BU and EL data.

Acknowledgment

This work is funded by Ministry of Higher Education (MOHE) under FRGS, vote 4F026 and supported by Research Management Centre (RMC), Universiti Teknologi Malaysia (UTM), MALAYSIA. The authors would like to express their deepest gratitude to the Boston University, Centre of Information and Communication Technology and Centre of Teaching and Learning, UTM especially to Mr. Azmi Kamis and Dr. Jamalludin Harun, Jo Ann Kelly from Salford Systems for her collaboration in making this research a successful one and Soft Computing Research Group (SCRG), K-Economy, UTM for their encouragement.

References

[1] M. Nottingham, “Caching tutorial for web authors and and webmasters,” June 2010, available from http://www.mnot.net/cache_docs/.

[2] V. N. Padmanabhan, and J. C. Mogul, “Using predictive prefetching to improve World Wide Web latency,” SIGCOMM Comput. Commun. Rev. 26, 3 (July 1996), pp. 22-36.

[3] A. Bestavros, and C. Cunha, “A prefetching protocol using client speculation for the WWW,” Technical Report. Boston University, Boston, MA, USA, 1995.

[4] G. Pallis, A.Vakali, and J. Pokorny, “A clustering-based prefetching scheme on a Web cache environment,” Comput. Electr. Eng. 34, 4 (July 2008), pp. 309-323.

[5] T. M. Kroeger, D. E. Long, and J. C. Mogul, “Exploring the bounds of web latency reduction from caching and prefetching,” In Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium on Internet Technologies and Systems (USITS'97). USENIX Association, Berkeley, CA, USA, 1997, 2-2.

[6] J. Xu, J. Liu, B. Li, and X. Jia, “Caching and prefetching for Web content distribution,” Computing in Science and Eng. 6, 4 (July 2004), pp. 54-59.

[7] A. S. Nair, and J. S. Jayasudha, “Dynamic Web pre-fetching technique for latency reduction,” In Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007) - Volume 04 (ICCIMA '07), Vol. 4. IEEE Computer Society, Washington, DC, USA, 2007, pp. 202-206.

[8] S.-H. Hung, C.-C.Wu, and C.-H. Tu., “Optimizing the embedded caching and prefetching software on a network-attached storage system,” In Proceedings of the 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing - Volume 01 (EUC '08), Vol. 1. IEEE Computer Society, Washington, DC, USA, 2008, pp. 152-161.

[9] W.-G. Teng, C.-Y. Chang, and M.-S. Chen, “Integrating web caching and web prefetching in client-side proxies,” IEEE Trans. Parallel Distrib. Syst. 16, 5 (May 2005), 2005, pp. 444-455.

[10] J. H. Lee and W.-K. Shiu, “An adaptive website system to improve efficiency with web mining techniques,” Advanced Engineering Informatics, 18 (2004), pp. 129-142.

[11] L. Breiman, J. Friedman, R. Olshen, and C. Stone, “Classification and regression trees,”Chapman & Hall/CRC Press, Boca Raton, FL, 1984.

[12] S. Gey, and E. Nédélec, “Model selection for CART regression trees,” IEEE Trans. Inf. Theory 51, 2005, pp. 658–670.

[13] J. H. Friedman, “Multivariate adaptive regression splines (with discussion)”. Annals of Statistics, 19, 1991, pp. 1–141.

[14] A. Abraham, and D. Steinberg, “MARS: Still an alien planet in soft computing?,” In: International Conference on Computational Science (Proceedings), Part II, vol. 2, Springer, San Francisco, California, USA, 2001, pp. 235-244.

[15] L. Breiman, “Random Forests,” Kluwer Academic Publishers, Machine Learning, 45, 2001, pp. 5–32.

[16] J.H. Friedman,"Stochastic gradient boosting", Computational Statistics & Data Analysis, vol. 38, 2002, pp. 367-378.

[17] R. Schapire, "A brief introduction to boosting," Proc. 16th International Joint Conference on Artificial Intelligence, 1999, pp. 1401-1405.

[18] J. Friedman and J. Meulman, "Multiple additive regression trees with application in epidemiology," Statistics in Medicine, vol. 22, 2003, pp. 1365-1381.

[19] M.O. Elish, K.O. Elish, “Application of TreeNet in predicting object-oriented software maintainability: a comparative study,” Software Maintenance and Reengineering, 2009. CSMR '09. 13th European Conference, 2009, pp. 69-78.

[20] C. Cunha, A. Bestavros, and M. Crovella, “Characteristics of WWW client-based traces,” Technical Report. UMI Order Number: 1995-010., Boston University, 1995.

[21] S. Sulaiman, S.M. Shamsuddin, F. Forkan, and A. Abraham, “Intelligent Web caching using neurocomputing and particle swarm optimization algorithm,” Second Asia International Conference on Modeling and Simulation, AMS 2008, IEEE Computer Society Press, USA, 2008, pp. 642-647.

[22] S. Sulaiman, S.M. Shamsuddin, and A. Abraham, “Rough Web caching,” Rough Set Theory: A True Landmark in Data Analysis, Studies in Computational Intelligence, Springer Verlag, Germany, 2009, pp. 187-211.

[23] S. Sulaiman, S.M. Shamsuddin, and A. Abraham,“An implementation of rough set in optimizing mobile Web caching performance,” Tenth International Conference on Computer Modeling and Simulation, UKSiM/EUROSiM 2008, Cambridge, UK, IEEE Computer Society Press, USA, 2008, pp. 655-660.

[24] S. Sulaiman, S.M. Shamsuddin, F. Forkan, and A. Abraham, S. Sulaiman, “Intelligent Web caching for e-learning log data,” Third Asia International Conference on Modelling and Simulation, AMS 2009, IEEE Computer Society Press, USA, 2009, pp. 136-141.

[25] A. Rousskov, and V. Soloviev, “On performance of caching proxies,” (extended abstract). SIGMETRICS Perform. Eval. Rev. 26, 1 (June 1998), pp. 272-273.

[26] M. Liu, F. Y. Wang, D. Zeng, and L. Yang, “An overview of World Wide Web caching,” IEEE International Conference on. Systems, Man, and Cybernatics, Volume 5, 2001, pp.3045-3050.

[27] F. Mohamed, “Intelligent Web caching architecture,” Master thesis, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, 2007.

[28] Koskela, T., “Neural network method in analysing and modelling time varying processes,” PhD dissertation, Helsinki University of Technology, 2004.