Journal of Theoretical and Applied Information Technology
20th September 2013. Vol. 55 No.2
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

# STUDY OF COMBINED WEB PRE-FETCHING WITH WEB CACHING BASED ON MACHINE LEARNING TECHNIQUE

**K R BASKARAN[1] DR. C.KALAIARASAN[2] A SASI NACHIMUTHU[3]**
[1] Associate Professor, Dept. of Information Tech., Kumaraguru College of Technology, Coimbatore, India
[2] Principal, Tamilnadu College of Engineering, Coimbatore, India
[3]Dept. of Information Tech., Kumaraguru College of Technology, Coimbatore, India
E-mail: [1]krbaski@yahoo.com [2]ckalai2001.yahoo.com

## ABSTRACT

High bandwidth utilization, reduced load on the origin server, high access speed are possible by combining Web caching and pre-fetching techniques. Pre-fetching is the process of fetching few Web pages in advance which will be assumed to be needed by the user in near future and those pages are cached in the memory. Lots of work has been reported for caching and pre-fetching of Web pages in the literature. In this paper, pre-fetching using clustering technique is combined with SVM (Support Vector Machine) – LFU algorithm, a machine learning technique for Web proxy caching .By using real dataset it will be shown that the SVM technique will be better than clustering based pre-fetching technique using caching policy like LFU considering bandwidth utilization and access latency.

**Keywords:** *Classification, Least Frequently Used (LFU), pre-fetching, Support vector machine (SVM)*

## 1. INTRODUCTION

With the wide spread use of WWW, there is high demand for computer networking resources. With unpredictable delays in retrieving Web pages and with more and more of web based applications being created by users, the increase in bandwidth does not address the delay problems [22]. The prediction algorithms available often predict relevant and irrelevant pages to the users. In caching the pre-fetched Web pages, efficient cache replacement methods have to be deployed to manage the cache content. The traditional cache replacement techniques used often fail to increase the cache hit ratio. Machine learning techniques are employed to increase the performance of Web proxy caching methods [5]. The remaining parts of the paper are organized as follows. Section 2 gives brief information of Web caching and Pre-fetching techniques. Section 3 contains the block diagram and description of it and the steps involved in the proposed method of combined clustering based pre-fetching technique with machine learning technique (SVM algorithm). It also gives the details of SVM and the algorithm for the combined intelligent Caching (SVM) and Pre-fetching. Section 4 discusses about performance evaluation and section 5 concludes the paper and suggests possible future works.

## 2. WEB CACHING AND PREFETCHING

Web caching is a technique for improving the performance of Web based systems. In Web caching, the Web objects that are likely to be accessed in the near future are kept closer to the user either in the client's machine or in the proxy server. Web caching reduces the latency perceived by the user, reduces bandwidth utilization and reduces the loads on the origin servers. If the cache is located in the client machine it is called browser cache. This type of cache is useful for a single user. If the cache is located in the proxy server it is called proxy cache. If the cache is located in the origin server it is called origin server cache. This will reduce redundant computations or redundant retrievals and it can reduce the server load. As Web proxy cache size is limited, efficient cache replacement methods are needed to handle the cache content. The conventional cache replacement policies are LRU, LFU, SIZE, GDS and GDSF [20].

The following are the factors (features) of Web objects that influence Web proxy caching:

a) Recency: object's last reference time.
b) Frequency: number of requests made to an object.
c) Size: size of the requested Web object.
d) Access latency of web object

The standard metrics used to analyze the performance of web caching methods are Hit Ratio

(HR) and Byte Hit Ratio (BHR) .HR is the percentage of number of requests that are served by the cache over the total number of requests.

BHR is the percentage of the number of bytes that correspond to the requests served by the cache over the total number of bytes requested. A high HR indicates the availability of the requested object in the cache most of the time and high BHR indicates reduced user-perceived latency and savings in bandwidth.

Web pre-fetching predicts the web objects that may be requested by the user in the near future. These predicted web objects are pre-fetched from the origin server during the browser idle time and stored in the cache. Pre-fetching increases the cache hits and reduces the user-perceived latency. The pre-fetching methods can be implemented on the client side or server side or on the proxy side.

The client-side pre-fetching helps in keeping track of the patterns of a single user across the various web servers.

The server-based pre-fetching helps in keeping track of the patterns of all the users accessing a particular web site. The proxy based pre-fetching helps in keeping track of the patterns of a group of users accessing many Web servers.

Pre-fetching algorithms are classified into content-based pre-fetching and history-based pre-fetching. The former method analyses the web page contents and predicts the HTML links to be followed by the clients. The latter method makes prediction based on observed page access behavior of the user in the past. The algorithms that come under this category are classified into four types namely approach based on dependency graph, approach based on Markov model, approach based on cost function and approach based on data mining. The data mining-based pre-fetching approach in turn is classified into association-based pre-fetching approach and clustering-based pre-fetching approach [20].If caching and pre-fetching methods are combined together, the hit ratio gets improved and the user-perceived latency gets reduced.

## 3. THE PROPOSED METHOD OF COMBINED CLUSTERING BASED PRE-FETCHING TECHNIQUE WITH MACHINE LEARNING TECHNIQUE

From the figure 1, Web pages that are referred by various clients are identified from the log file. The Web log file contents are preprocessed and trained using the features namely recency, frequency, retrieval time and size of web object. Dataset is created from the proxy log file.

A Web navigation graph (WNG) is constructed for each user independently using a user's session time interval of thirty minutes. It is considered that the difference between two subsequent requests should not be greater than 30 minutes [15]. At the end of each time interval new navigation graph are constructed for each user based on the content of log files. The WNG shows the navigations made between the various Web objects by each user. Each node in the WNG represents a Web object requested by the user and each edge represents user's transitions from one Web object to another and a weight is assigned to each edge which represents the number of transitions between those objects.

A clustering algorithm gets the contents of WNG as input. Support and Confidence are the parameters used to keep track of frequently visited pages by the user.

A threshold value is fixed for these parameters and those edges which have values less than this threshold are removed [20]. Support is defined as the frequency of navigation between two nodes $u1$ and$u2$. The confidence is defined as $freq(u1, u2)/\,pop(u1)$ where $pop(u1)$ is the popularity of u1. Popularity of a node is the number of incoming edges in to that node. The WNG is partitioned in to sub graphs by removing those edges having low support and confidence values. The nodes in each connected sub-graph are identified as a cluster. When a user requests any one of the nodes (Web objects) in a cluster and if it is found in the long- term cache or in the short-term cache, all the remaining nodes in that cluster can be pre-fetched in to the short-term cache by predicting that those objects will be requested by the user in the near future [20]. This pre-fetching is done during the browser idle time and this pre-fetching helps in reducing the user perceived latency time. If the access count of a Web object is greater than the threshold value then that Web object is moved from the short term cache to long term cache. LFU technique is used for removal of pages from the short term cache if sufficient space is not available for caching a new Web object.SVM classifier is used to classify the Web objects as class 0 or class1 [5] while moving them from short term cache to long term cache. When a user requests for a Web object, simultaneous search is made in short term cache as well as in the long term cache. If the requested Web object is available in the short-term cache, its access count is increased by one.

Journal of Theoretical and Applied Information Technology
20th September 2013. Vol. 55 No.2
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

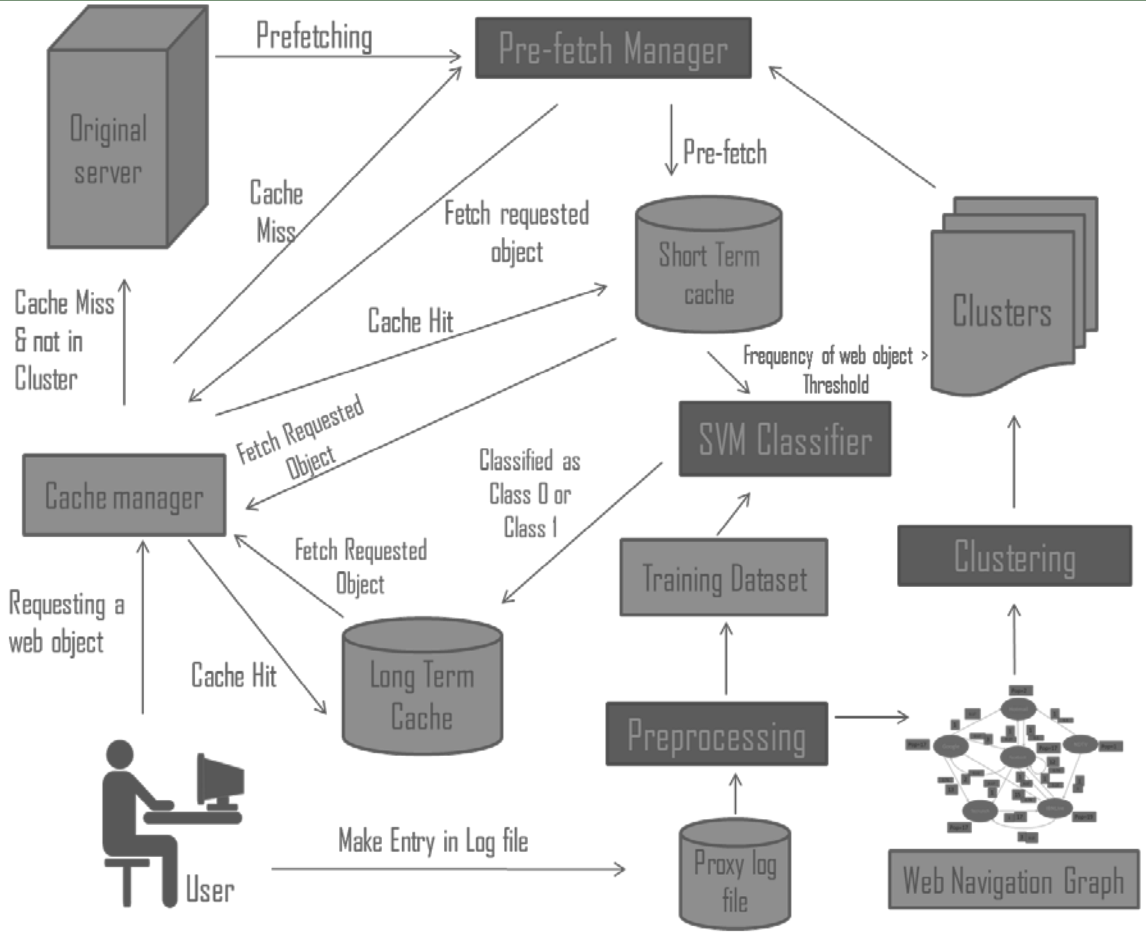


*Figure 1: Block Diagram*

If the access count is greater than the threshold value then that web object is given as input to SVM classifier for classification. If it is classified as class 0, it is moved to the bottom of long-term cache. If it is classified as class 1, it is moved to the top of long-term cache. If there is no space in the long-term cache, the Web object(s) placed at the bottom of the long-term cache are removed to provide space for this new Web object. Meanwhile a copy of this web object is given to the requested user. If that Web object is found in the long-term cache, classification of that Web object is done again.

If it is re-classified as class 1, it is moved to the top of the cache (long-term cache) else it is moved to the bottom of that cache. It is then given to the requested user and pre-fetching of other web objects if any belonging to that cluster in to the short-term cache is initiated.

If cache miss occurs, the requested object is searched in all the clusters generated by Clustering algorithm for that user.

If it is found in any one of those clusters**,** that Web object is fetched from the original server and it is sent to the user as well as placed in to the short cache. Other Web objects that are present in that cluster will be pre-fetched during browser idle time and will be placed in the short-term cache.

If the requested object is not found in any of the clusters of that user, then the required Web page is fetched from the original server. A copy of it is placed in the short term cache and it is sent to the requested user.

Thus by combining Web caching and pre-fetching it is possible to increase the hit ratio, decrease user perceived latency and reduce the origin server load.

### 3.1 Preprocessing Step

A log file generated by the proxy server consists of the following fields:

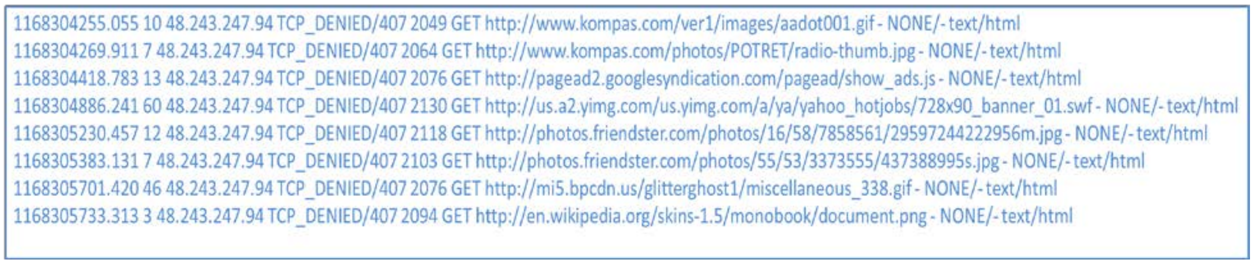
- Time stamp (Date and time) of the request made
- Time required (in milliseconds) to process the request
- Machine IP address making the request
- Cache hit/ Cache miss and Response code
- Size of the requested object measured in bytes.
- Type of method used
- URL of the requested page
- Information if the request is redirected to other server
- Content type

class of the web object. The data type of a1, a2, a3, a4 is numeric and the data type of b is nominal [5].

### 3.2. Web Navigation Graph (WNG)

A weighted directed Web graph $G(x,y)$ is used to represent the requests of each user, where each node $x$ represents a Web object and each edge $y$ represents a user's transition from one Web object to another. The weight of each edge represents the number of transitions in the set. To make the size of the WNG manageable, the edges are removed whose connectivity between two Web objects is lower than a specified threshold. Support and confidence are the two parameters that determine the connectivity between two objects [20].

Let $W\!:< xi, xj >$ be an edge from node $xi$ to node $xj$.



```
1168304255.055 10 48.243.247.94 TCP_DENIED/407 2049 GET http://www.kompas.com/ver1/images/aadot001.gif - NONE/- text/html
1168304269.911 7 48.243.247.94 TCP_DENIED/407 2064 GET http://www.kompas.com/photos/POTRET/radio-thumb.jpg - NONE/- text/html
1168304418.783 13 48.243.247.94 TCP_DENIED/407 2076 GET http://pagead2.googlesyndication.com/pagead/show_ads.js - NONE/- text/html
1168304886.241 60 48.243.247.94 TCP_DENIED/407 2130 GET http://us.a2.yimg.com/us.yimg.com/a/ya/yahoo_hotjobs/728x90_banner_01.swf - NONE/- text/html
1168305230.457 12 48.243.247.94 TCP_DENIED/407 2118 GET http://photos.friendster.com/photos/16/58/7858561/29597244222956m.jpg - NONE/- text/html
1168305383.131 7 48.243.247.94 TCP_DENIED/407 2103 GET http://photos.friendster.com/photos/55/53/3373555/437388995s.jpg - NONE/- text/html
1168305701.420 46 48.243.247.94 TCP_DENIED/407 2076 GET http://mi5.bpcdn.us/glitterghost1/miscellaneous_338.gif - NONE/- text/html
1168305733.313 3 48.243.247.94 TCP_DENIED/407 2094 GET http://en.wikipedia.org/skins-1.5/monobook/document.png - NONE/- text/html
```

*Figure 2 Sample Log File*

In pre-processing, the following are carried out on the contents of log file [20]:

- JavaScript files, Cascading style sheet, images that are present in the log files and not requested by the users are removed.
- Invalid requests from the proxy log file that refer to either internal sever errors and server side errors are also removed.
- It is necessary to remove all the requests that are not explicitly requested by the user from the Web proxy log file.

After removing all the unwanted entries from the log file

- Generate separate log file for individual users for creating inter-site Web clustering of Web objects.

#### 3.1.1 Training dataset creation

In order to create a training dataset, information is extracted from the traces of log file. Each log file record is converted to the training pattern in the format of <a1, a2, a3, a4, b> where a1 represents recency (time of since the last references to the object) of the web object, a2 represents frequency of the Web object, a3 represents the size of the Web object, a4 represents the retrieval time (access latency) of the Web object and b represents the

Support of $G$, denoted by $freq(xi, xj)$ is defined as the frequency of navigation steps between$xi to xj$. Confidence of g is defined as $freq(xi, xj)/pop(xi)$, where $pop(xi)$ is the popularity of$xi$.By this definition, the support value of the edge $(x3, x4)$ for the user X in Figure 3 is$(x3, x4) = 1$ and confidence value is $freq(x3, x4)/pop(x3) = 0.5$. If the support threshold chosen is very less, too many less important user's transitions for clustering may be included and if the chosen threshold value is high, many interesting transitions that occur at low levels of support may be missed.

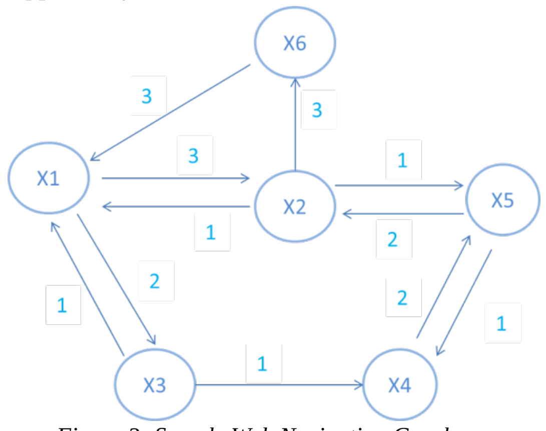


*Figure 3: Sample Web Navigation Graph*

### 3.3. Web Clustering Algorithm

The algorithm for clustering inter-site Web pages is described below [20]. A weighted directed Web graph $G(x,y)$ that represents the access patterns of a user is used. This graph is partitioned into sub graphs by filtering edges with low support and confidence values. The nodes in each connected sub graph in the remaining navigational graph will form a cluster. The inputs to this clustering algorithm will be Web navigational graph, the number of users, support threshold and confidence threshold. All the edges with support or confidence value less than the corresponding threshold values are removed. BFS (Breadth First Search) algorithm is applied to the navigational graph. BFS takes a node in the graph (called as source) and visits each node reachable from the source by traversing the edges. It outputs a sub-graph that consists of the nodes reachable from the source. This procedure is applied for all the nodes of the graph. All the nodes in each connected sub-graph forms a cluster. The time complexity of BFS is $O(|x| + |y|)$ where $|x|$the number of nodes is and $|y|$ is the number of edges in the graph [20].

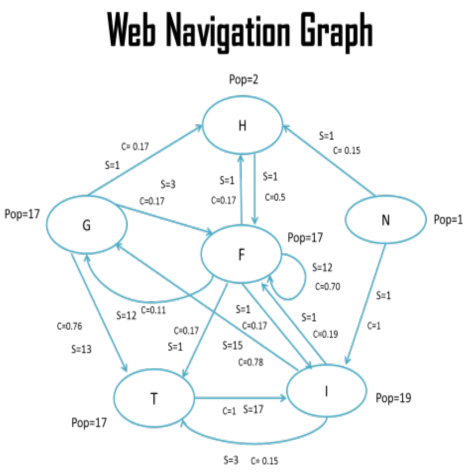


*Figure 4 Web Navigation Graph for User1*

In the above Web Navigation Graph

- G stands for Google Web object
- H stands for Hot Mail Web object
- N stands for NDTV Web object
- F stands for Face Book Web object
- I stands for IBN Web object
- T stands for Techrench Web object

User access pattern of above WNG is

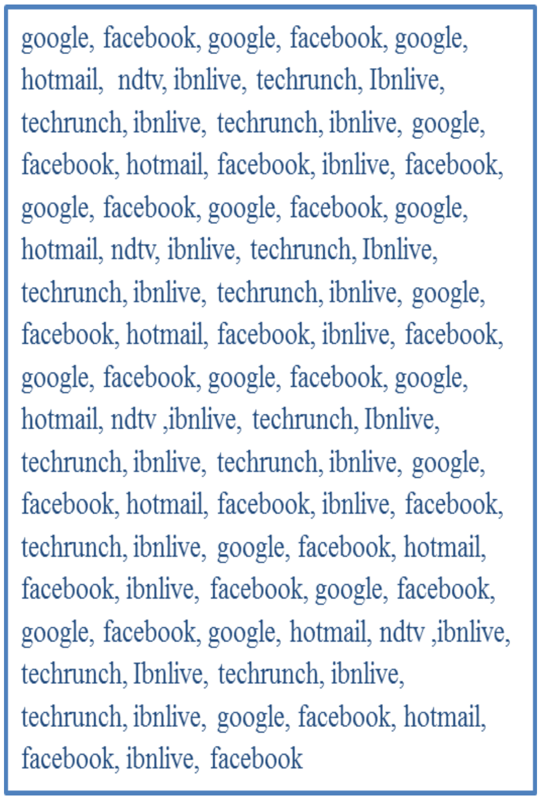

google, facebook, google, facebook, google, hotmail, ndtv, ibnlive, techrunch, Ibnlive, techrunch, ibnlive, techrunch, ibnlive, google, facebook, hotmail, facebook, ibnlive, facebook, google, facebook, google, facebook, google, hotmail, ndtv, ibnlive, techrunch, Ibnlive, techrunch, ibnlive, techrunch, ibnlive, google, facebook, hotmail, facebook, ibnlive, facebook, google, facebook, google, facebook, google, hotmail, ndtv ,ibnlive, techrunch, Ibnlive, techrunch, ibnlive, techrunch, ibnlive, google, facebook, hotmail, facebook, ibnlive, facebook, techrunch, ibnlive, google, facebook, hotmail, facebook, ibnlive, facebook, google, facebook, google, facebook, google, hotmail, ndtv ,ibnlive, techrunch, Ibnlive, techrunch, ibnlive, techrunch, ibnlive, google, facebook, hotmail, facebook, ibnlive, facebook

*Figure 5: User Access Pattern for User1*

The access pattern for the user1 consists of 6 different Web objects. From the access pattern information, WNG is constructed.

Support and Confidence value for each edge in the WNG is calculated as defined in section 3 and the popularity for each Web object is computed.

Support threshold value and confidence threshold values are assumed to be 2 and 0.6 respectively.

Those edges which have Support and Confidence values less than their threshold are removed. The threshold value chosen for Support and Confidence are critical in specifying the cluster size. It should be noted that the total size of all the objects in a cluster shall not exceed the total cache size.

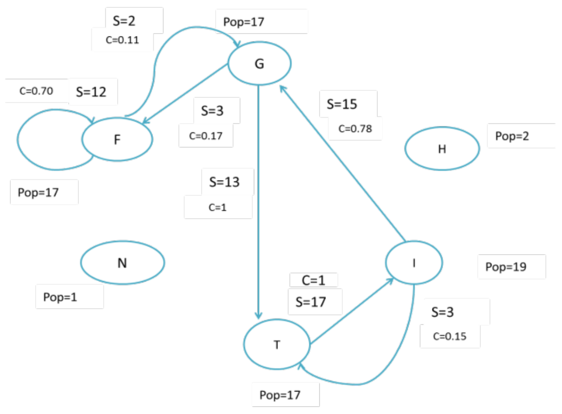


*Figure 6: Applying Support Threshold*

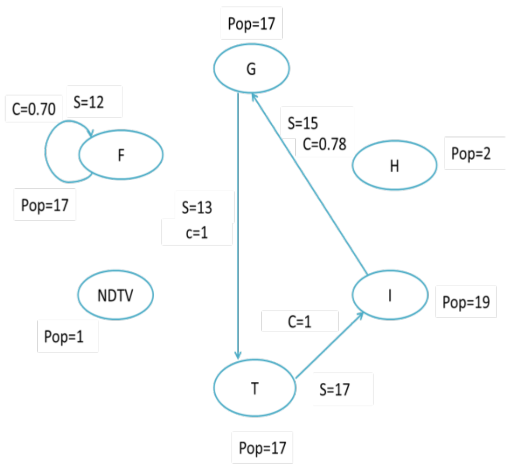


*Figure 7: Applying Confidence Threshold*

Then BFS algorithm is applied to the above navigational graph. For each node in the graph known as source, BFS algorithm tries to visit every other node that can be reached from the source by traversing the edges.

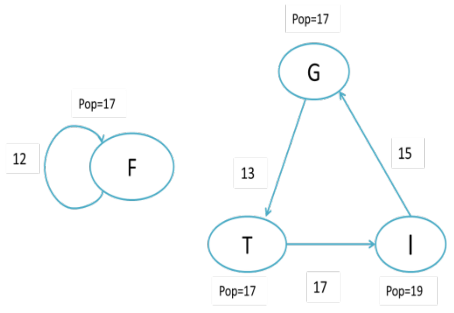


*Figure 8: Applying BFS*

It outputs a sub graph that consists of all the nodes reachable from the source. This procedure is iterated until BFS has traversed all the nodes of the initial graph. The nodes in every connected sub graph in the remaining graph forms a Web cluster.

### 3.4. Clustering Based Pre-fetching

The following are the steps that take place in the proposed pre-fetching method [20]:

- A user requests a web object.
- Using the IP address, the proxy identifies the user and maps the user to a particular user group. Given that the clusters of Web objects are known, the proxy searches inside the existing clusters of that user group to find in which cluster the requested object exists.
- All the remaining objects from the selected cluster are pre-fetched from the origin server by the proxy and they are loaded into the short-term cache during the browser idle time.
- The proxy sends to the user his/her requested object.

### 3.5. Intelligent Web Proxy Caching Algorithms

Compared to traditional caching approaches, intelligent Web caching methods are more efficient.

Details about intelligent caching methods are found in [4] and that of conventional replacement policies are found in [23]**.** In our work, the proxy cache is divided in to short-term cache and long-term cache. The Web objects requested for the first time are loaded in to short-term cache. LFU algorithm is used for managing short-term cache. Those objects visited more than once from the short- term cache are moved to long-term cache. In the literature many techniques were used for cache replacement. Back-propagation neural network has been used in NNPCR [10] and NNPCR-2[23]. The drawback was that BPNN learning process can be time consuming and BPNN classifier in cache replacement decision did not take into account the cost and object size in decision making. Combined BPNN as caching decision policy and LRU as replacement policy was proposed by Farhan(2007) [13]. However recency factor which is considered as an important factor was isgnored in the above technique. Farhan's approach was enhanced using particle swarm optimization by Sulaiman et al. (2008) [24]. This approach however did not incorporate enhanced classifier. Koskela et al. (2003) [16] used multilayer perceptron network (MLP) classifier in Web caching. This method used HTML structure of the document and HTTP responses of the server as inputs to MLP to predict the class of Web objects. This class value was integrated with LRU, called LRU-C to optimize the Web cache. This method however ignored the frequency factor in the replacement of cache contents. A logistic regression model to predict future requests was proposed by Foong et al. (1999) [14]. In his work, regardless of cost and size of the predicted object, objects with lowest re-access probability value were replaced first. From the above studies it is observed that intelligent caching technique can be employed either individually or can be combined with LFU technique. Both of these approaches predict Web objects that will be re-accessed in the near future without considering the cost and size of the predicted objects for replacement. Extra computational overheads and longer duration are required for training process. In our work SVM machine learning technique will be used that will classify Web objects and make more accurate predictions of those objects that will be re-accessed later.

### 3.5.1. Support vector machine (SVM)

SVM (Support Vector Machine) is a technique useful for data classification. SVMs are supervised learning models associated with learning algorithms for analyzing data and recognizing patterns which are used for classification and regression analysis. It takes a set of input data and predicts to which one of the two possible classes, the given input data belongs to. Given a set of training datasets, each marked as belonging to one of two classes, a SVM training algorithm constructs a model that classifies new data inputs into one category or the other. SVM determine the hyper-plane to do binary division that is used to find the linear boundary between two classes such as Positive (Class1) and Negative (Class0) [5]. The hyper-plane is placed in between two classes and it is oriented like, the distance between the plane and data point in each class is maximized. The nearest data points are called as Support Vectors. In an arbitrary dimensional space a separating hyper plane can be written

$$w.x + b = 0$$

Where, $b$ is bias, $w$ the weights and$x$is the input vector

The decision function can be written as

$$D(x) = sign\,(w.x + b)$$

If the sign of the decision function is positive the object is classified as Class 1 (Positive Class), otherwise the object is classified as Class 0 (Negative Class)

$$w.x + b = 1\,(Class\ 1)$$

$$w.x + b = -1(Class\ 0)$$

Thus,

$$w.(x1 - x2) = 2$$

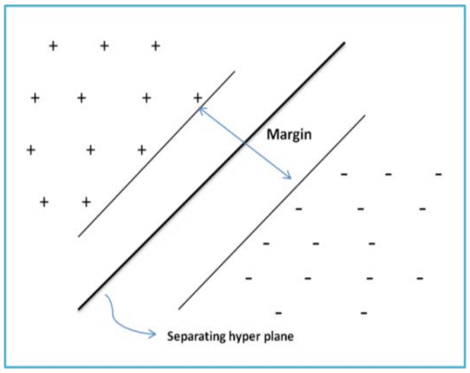


*Figure 9: SVM Hyper-plane Classifying Input Data*

For two support vectors on each side of the separating hyper plane, the margin will be given by the projection of the vector(x1-x2) onto the normal vector of hyper plane i.e. w/ ||w|| from which it is deduced that the margin is

$$Y = \frac{1}{2}||W||^2$$

The maximization of the margin is thus equivalent to the minimization of the functional

$$\Phi\,(w) = \frac{1}{2}(||W||^2)$$

Subjected to constraints

$$Yi[(w.x) + b] \geq 1$$

Thus, the task is to find an optimal of the primal objective function

$$\boldsymbol{L\,(w,b) = \frac{1}{2}(w.w) - \sum_{i=1}^{m} di[Yi((w.Xi) + b) - 1]}$$

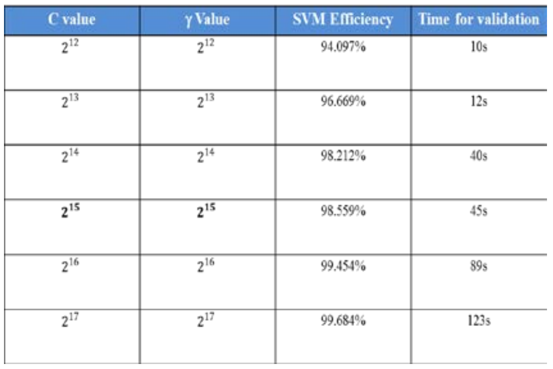
For most of the real-time application, it is not easy to find a hyper-plane to classify the data such as nonlinearly separable data. The input data is only transformed from the original space into much higher dimensional space called the feature space.

Several Kernel functions like polynomial, sigmoid, RBF can be used in SVM training. RBF kernel achieves a better performance compared to other kernel functions. Based on the recommendations of [26], SVM is trained as follows:

Preparing and normalizing the dataset, considering the RBF kernel, finding the best values for the parameters C (margin softness) and γ (RBF width) using cross-validation, using the best parameters to train the whole dataset and test [5]. The accuracy of a SVM model is largely dependent on the selection of the RBF Kernel parameters (c and gamma) [5]. 'C' value controls over-fitting of the model and gamma (γ) value controls the degree of nonlinearity of the model. The values of these training parameters C and gamma are determined by Grid search method.

Using all possible combination values for C and γ, the Grid search method finds out the optimal value for C and γ such that SVM will classify input data efficiently. Then, the performance of the constructed model is evaluated using 10-fold cross validation on the training data. If there are many training instances, more time will be taken by the grid search to compute optimum value for 'C' and gamma (γ) parameters because it does cross validation on it. Hence values for parameters(C and gamma) are selected on a small subset of the training data and use those values to train SVM model for the remaining training dataset.

From the data given in the table below, $2^{15}$ is found as the optimum value for C and γ by considering SVM efficiency and processing time for validation.



| C value | γ Value | SVM Efficiency | Time for validation |
|---|---|---|---|
| $2^{12}$ | $2^{12}$ | 94.097% | 10s |
| $2^{13}$ | $2^{13}$ | 96.669% | 12s |
| $2^{14}$ | $2^{14}$ | 98.212% | 40s |
| $\mathbf{2^{15}}$ | $\mathbf{2^{15}}$ | 98.559% | 45s |
| $2^{16}$ | $2^{16}$ | 99.454% | 89s |
| $2^{17}$ | $2^{17}$ | 99.684% | 123s |

*Figure 10: SVM Efficiency Calculation for Various Values of C and Gamma*

### 3.6. Algorithm for Combined Intelligent Caching and Pre-fetching

```
Begin
For each web object m requested by the user
If m is in short-term cache
Begin
Cache hit occurs
Fetch the requested object m from short-term cache
Update the information of m
If the frequency of m> threshold limit
Begin
While no space in the long-term cache for m
Begin
        Expel f from long-term cache such that f is
        in bottom of the long-term cache
End
Class of m=apply-svm(Common features)
If class of m=1
        Move m to top of the long-term cache
Else
        Move m to the long-term cache
End
End
Else if m is in long-term cache
Begin
        Cache hit occurs
        Fetch the requested object m from the
        long-term cache
        Update the information of m
        Recalculate the class of m
        If class of m =1
                Move m to the top of the long-
                term cache
        Else
                Move m to the long –term cache
        End
Else
Begin
        Cache miss occurs
        Fetch m from original server
        Cluster c=getClusterForUser(m)
        If c is not NULL
        Begin
                While no space in short-term
                cache for web objects in c
                Begin
                Expel object using LFU from
                short-term cache
                End
        Load the all cluster into the short-term
        cache
        End
End

   Procedure getClusterForUser(m)
    Begin
        For each cluster p for the user
                If m is in cluster p
                        Return p
        If m is not in any of the cluster
                return NULL
    End
End
```

### 3.6.1 Clustering Algorithm

```
Begin
For each client IP address
Begin
Construct web navigation graph G (U, V)
End
For each G (U, V)
Begin
        Calculate Support, Confidence and Popularity of the node U
        If Confidence < Threshold limit of Confidence
        Begin
                Remove U
        End
        If Support< threshold limit of Support
        Begin
                Remove V
        End
        Cluster C = Apply BFS for G (U, V)
End
```

### 3.6.2 Breadth first search

```
Begin
Input: Graph G (U, V)
Choose some starting node u1
Mark u1 as visited
Initialize list x1 with u1
Initialize sub-graph T with u1
While x1 is not empty
Begin
        Choose node x2 from front of the list
        Visit x2
End
For each unmarked neighbor y
Begin
        Mark y
        Add y to the end of the list x1
        Add x2->y to sub-graph T
End
Find all neighbors of the node u1
Visit each neighbor and mark the visited node
End
```

## 4. PERFORMANCE EVALUATION

### 4.1. Dataset

The scheme explained in this paper is tested with a dataset. The dataset is obtained from a proxy server installation ftp://ftp.ircache.net/Traces/DITL-2007-01-09/. The filename of the dataset used for testing of the scheme is rtp.sanitized-access.20070109.gz under the website. The raw proxy server log file for the dataset contained the details of more than 3 million requests. After the log cleaning process was applied on the dataset, the dataset contained about 610,634 entries fit for analysis. The inner details about the data set are as explained below:

Total Number of Items: 610634
Total Size of Bytes for Dataset: ~ 49 GB
Total Number of Items used for *Clustering*: 427443
Total Size of Bytes Used for *Clustering*: ~36 GB
Total Number of Items used for *Testing*: 183191
Total Size of Bytes Used for *Testing*: ~ 13GB

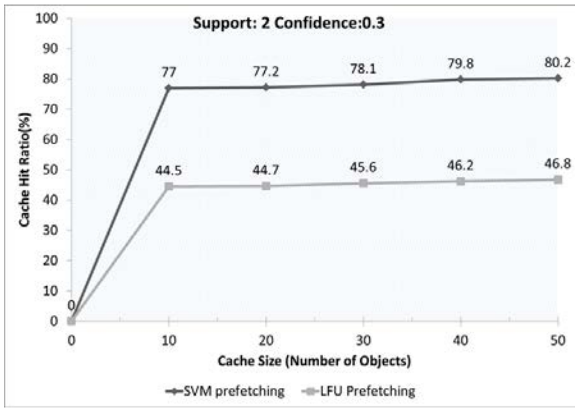
70% of all the requests ordered by time have been used for the user's access pattern analysis, creating training dataset and testing [20].The remaining 30% of the requests were used for testing the scheme.

### 4.2. Performance Metrics

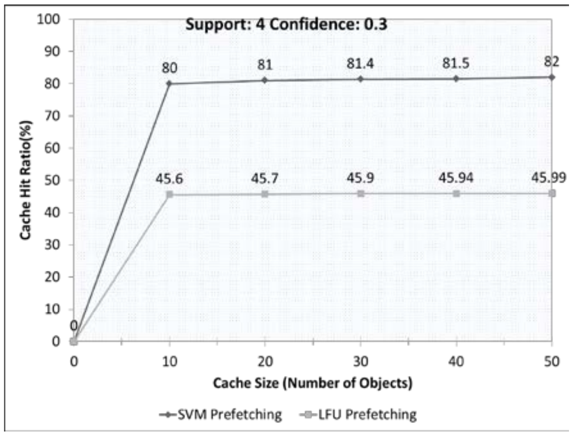
- **Hit Ratio (HR):**
  HR is the percentage of the total number of requests served by the cache over the total number of requests given.
- **Byte Hit Ratio (BHR):**
  BHR gives the percentage of the number of bytes corresponding to the requests served by the cache over the total number of bytes requested.

### 4.3. Experimental Results

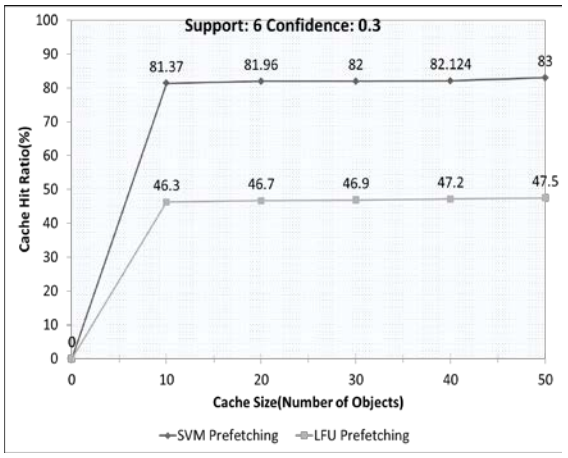
#### 4.3.1. Hit ratio analysis



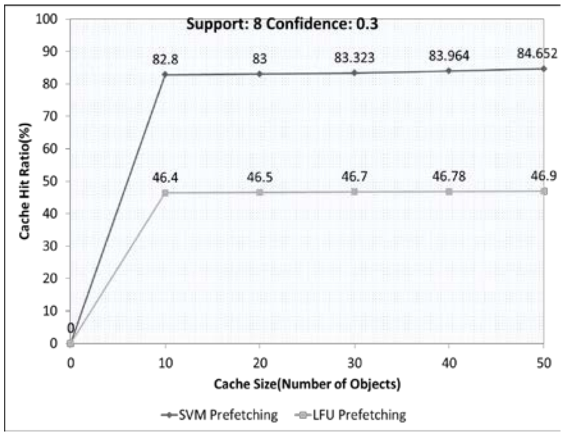
*(a) Support: 2 Confidence: 0.3*



(b) Support: 4 Confidence: 0.3



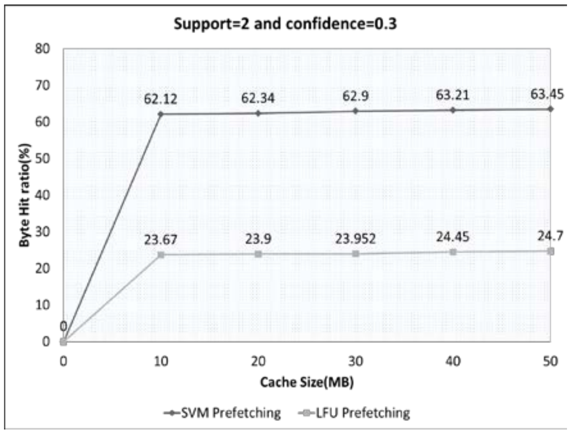
(c) Support: 6 Confidence: 0.3



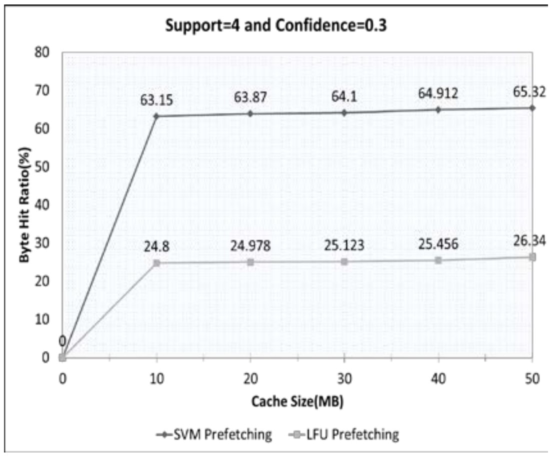
(d) Support: 8 Confidence: 0.3

*Figure 11: Analysis of HR Using SVM and LFU Pre-fetching on Different Values of Support and Confidence*

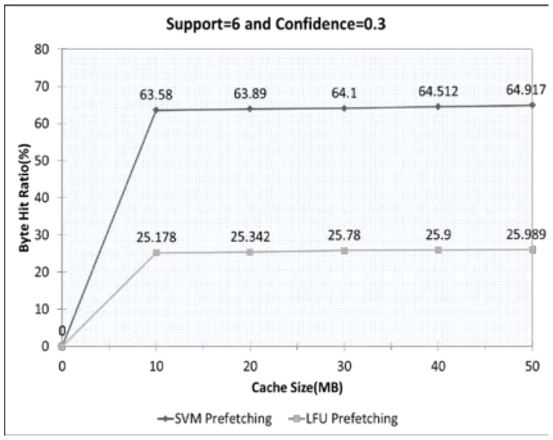
### 4.3.2. Byte hit ratio analysis



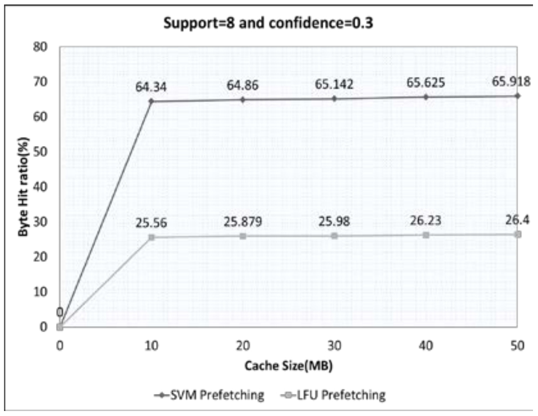
(a) Support: 2 Confidence: 0.3



(b) Support: 4 Confidence: 0.3



(c) Support: 6 Confidence: 0.3



*(d) Support: 8 Confidence: 0.3*

*Figure 12: Analysis of BHR Using SVM and LFU Pre-fetching on Different Values of Support and Confidence*

HR and BHR are calculated using different values of Support and Confidence using SVM and LFU pre-fetching. The performances for both of these methods for various cache sizes are plotted. Results inferred from the graphs are stated below:

**1.** Increase in Confident and Support value leads to increase in number of clusters created and decrease in number of Web objects in the created clusters.

2. Increase in Support and Confidence values leads to increase in HR for increasing cache sizes.

**3.** SVM pre-fetching leads to significant increase in the HR independent of the cache size, Support and Confident values in comparison with LFU pre-fetching.

**4.** Considering BHR, it is found that using SVM pre-fetching on an average 78% of the total size of the information requested is found fetched from cache due to cache hit and 23% of the total size of the information requested is found fetched from cache using LFU pre-fetching and the rest of the information fetched from the original server. Thus the superiority of SVM pre-fetching is established. **.**

**5.** SVM pre-fetching leads to decrease in the network bandwidth utilization as a result of more cache hits and an increase in computer network performance.

**6.** Users experience decreased access latency while using SVM pre-fetching because of increase in percentage of cache hits. This ensures lesser load on the original server.

## 5. CONCLUSION AND FUTURE WORK

In this paper, a user's log file is preprocessed and a Web navigation graph is constructed. Clustering of the inter Web objects in the WNG is done using a clustering algorithm. Frequency of frequently used Web objects is monitored by Support and Confidence values. Availability of requested Web object in the short-term cache or long-term cache leads to pre-fetching of all other Web objects in that cluster during browser idle time. Using access count a Web object in short-term cache is moved to long-term cache after classification using SVM algorithm. A cache miss in both the caches leads to fetching of that object from the original server and a copy of it being placed in the short-term cache. Comparison of the efficiency of SVM pre-fetching with that of .FU-pre-fetching is done using real data set and it ; inferred that SVM pre-fetching has high HR as well as high BHR for various values of Support, Confidence and cache sizes.

This work can be extended by clustering intra Web objects along with inter Web objects and comparison of efficiency of SVM pre-fetching with that of LFU pre-fetching can be experimented.

## REFERENCES

[1] Abrams M, Standridge C R, Abdulla, Fox G A, Williams S, "Removal Policies in
Network Caches for World-Wide Web Documents", *ACM*, 1996, pp. 293–305.

[2] Ali W, Shamsuddin S M "Intelligent client-side web caching scheme based on least recently used algorithm and neuro-fuzzy system", *in: W. Yu, H. He, N. Zhang (Eds.), Advances in Neural Networks — ISNN 2009, Springer, Berlin/Heidelberg, 2009,* pp. 70–79.

[3] Ali W, Shamsuddin S M, Ismail A S, "Web proxy cache content classification based on support vector machine", *Journal of Artificial Intelligence* 4 (2011) 100–109.

[4] Ali W, Shamsuddin S M, Ismail A S," A survey of Web caching and prefetching", *International Journal of Advances in Soft Computing and Its Applications* 3 (2011) 18.

[5] Ali W, Shamsuddin S M, Ismail A S, "Intelligent Web proxy caching approaches based on machine learning techniques", *Decision Support Systems 53 (2012)* 565-579.

[6] Chen C, Hsieh C, "Web page classification based on a support vector machine using a weighted vote schema", *Expert Systems with Applications* 31 (2006) 427–435.

[7] Chen T, "Obtaining the optimal cache document replacement policy for the caching system of an EC website", *European Journal of Operational Research* 181(2007) 828–841.

[8] Chen X, Zhang X. "Popularity-based PPM: an effective web prefetching technique for high accuracy and low storage." *In: Proceedings of the international conference on parallel processing. Canada, Vancouver; 2002.*

[9] Chen Y, Qiu L, Chen W, Nguyen L, Katz R H. "Efficient and adaptive Web replication using content clustering." *IEEE J Selected Areas Communication 2003*; 21(6):979–94.

[10] Cherkasova L, "Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy", *HP Technical Report, Palo Alto*, 1998.

[11] Cobb J, ElAarag H, "Web proxy cache replacement scheme based on back-propagation neural network", *Journal of Systems and Software* 81 (2008) 1539–1558.

[12] Deshpande M, Karypis G. "Selective Markov models for predicting Web-page accesses." *In: Proceedings of the 1st SIAM international conference on data mining. Chicago, USA; 2001.*

[13] Farhan, "Intelligent Web Caching Architecture", *Faculty of Computer Science and Information System, UTM University, Johor, Malaysia, 2007.*

[14] Foong A P, Yu-Hen H, Heisey D M, "Logistic regression in an adaptive Web cache", *IEEE Internet Computing* 3 (1999) 27–36.

[15] Jyoti, Sharma A, &Goel A. (2009). "A novel approach for clustering web user sessions using RST" *Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT*, 2(1), 656-661.

[16] Koskela T, Heikkonen J, Kaski K, "Web cache optimization with nonlinear model using object features", *Computer Networks* 43 (2003) 805–817.

[17] Kumar C, "Performance evaluation for implementations of a network of proxy Caches", *Decision Support Systems* 46 (2009) 492–500.

[18] Pallis G, Angelis L, Vakali A. "Validation and interpretation of Web users' sessions clusters." *Information Processing and Management 2007;*43(5):1348–67.

[19] Pallis G, Vakali A. "Insight and perspectives for content delivery networks. " *Communication ACM (CACM) 2006*;49(1):101–6.

[20] Pallis G, Vakali A, Pokorny J, "A clustering-based pre-fetching scheme on a Web cache environment", *Computers and Electrical Engineering 34 (2008)* 309-323.

[21] Pallis G, Vakali A, Sidiropoulos E. "FRES-CAR: An adaptive cache replacement policy." *In: Proceedings of the 1st IEEE international workshop on challenges in Web information retrieval and integration (WIRI'05) in cooperation with the 21st IEEE conference on data engineering ICDE 2005 Tokyo, Japan; 2005.*

[22] Podlipnig S, Boszormenyi L. "A survey of Web cache replacement strategies" *ACM Computer Surveys 200*; 35(4):374–98.

[23] Romano S, ElAarag H,"A neural network proxy cache replacement strategy and its implementation in the Squid proxy server", *Neural Computing and Applications* 20(2011), 59-78.

[24] Sulaiman S, Shamsuddin S M, Forkan F, Abraham A, "Intelligent Web caching using neurocomputing and particle swarm optimization algorithm", *Modeling & Simulation, 2008. AICMS 08. Second Asia International Conference on, 2008*, pp. 642–647.

[25] Xing D, Shen J. "Efficient data mining for web navigation patterns" *Inform Software Technology 2004*; 46(1):55–63.

[26]Web reference: http://www.wikipedia.com/svm

[27] Yang Q, Zhang H. "Integrating Web pre-fetching and caching using prediction models." *World Wide Web 2001*;4(4):299–321.