Int. J. Advance. Soft Comput. Appl., Vol. 3, No. 1, March 2011
ISSN 2074-8523; Copyright © ICSRS Publication, 2011
www.i-csrs.org

A Survey of Web Caching and Prefetching

Waleed Ali , Siti Mariyam Shamsuddin, and Abdul Samad Ismail

Soft Computing Research Group, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia,
81310 Skudai, Johor
email: waleedalodini@gmail.com
Soft Computing Research Group, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia,
81310 Skudai, Johor
email: mariyam.utm.my:
Department of Communication and Computer Systems, Faculty of Computer Science and Information Systems, Universiti Teknologi Malaysia,
81310 Skudai, Johor
email: abdsamad@utm.my

Abstract

Web caching and prefetching are the most popular techniques that play a key role in improving the Web performance by keeping web objects that are likely to be visited in the near future closer to the client. Web caching can work independently or integrated with the web prefetching. The Web caching and prefetching can complement each other since the web caching exploits the temporal locality for predicting revisiting requested objects, while the web prefetching utilizes the spatial locality for predicting next related web objects of the requested Web objects. This paper reviews principles and some existing web caching and prefetching approaches. The conventional and intelligent web caching techniques are investigated and discussed. Moreover, Web prefetching techniques are summarized and classified with comparison limitations of these approaches. This paper also presents and discusses some studies that take into consideration impact of integrating both web caching and web prefetching together.

Keywords: *Web caching, Web prefetching, Intelligent techniques, Markov model, Association rules and Clustering.*

1 Introduction

Web caching is a well-known strategy for improving the performance of Web-based system by keeping Web objects that are likely to be used in the near future in location closer to user. The Web caching mechanisms are implemented at three levels: client level, proxy level and original server level [1, 2]. Significantly, proxy servers play the key roles between users and web sites in lessening of the response time of user requests and saving of network bandwidth. Therefore, for achieving better response time, an efficient caching approach should be built in a proxy server.

The cache replacement is the core or heart of the web caching; hence, the design of efficient cache replacement algorithms is crucial for caching mechanisms achievement [2]. Thus, cache replacement algorithms are also called web caching algorithms [3]. Due to the limitation of cache space, an intelligent mechanism is required to manage the Web cache content efficiently. The traditional caching policies are not efficient in the Web caching since they consider just one factor and ignore other factors that have impact on the efficiency of the Web caching [4, 3, 5]. In these caching policies, most popular objects get the most requests, while a large portion of objects, which are stored in the cache, are never requested again. This is called cache pollution problem. Therefore, many Web cache replacement policies have been proposed attempting to get good performance. However, combination of the factors that can influence the replacement process to get wise replacement decision is not an easy task because one factor in a particular situation or environment is more important than other environments [1, 6]. Hence, the difficulty in determining which ideal web objects will re-accessed is still a big challenge faced by the existing Web caching techniques. In other words, what Web objects should be cached and what 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 [7, 1, 8, 3].

Unfortunately, the cache hit ratio is not improved much with caching schemes. Even though with a cache of infinite size, the hit ratio is still limited only at the range from 40% to about 50%, regardless of the caching scheme [9, 10, 11]. This is because most people browse and explore the new web pages trying to find new information. In order to improve the hit ratio of cache, Web pre-fetching technique is integrated with web caching to overcome these limitations.

The web prefetching is a hot research topic that has gained increasing attention in recent years. The cache prefetching fetches some web objects before users actually request it. Thus, the cache prefetching helps on reducing the user-perceived latency. Many studies have shown that the combination of caching and prefetching doubles the performance compared to single caching [12, 13, 14, 15, 16]. According to [12, 13] , a combination of web caching and prefetching can potentially improve latency up to 60%, whereas web caching alone improves the latency up to 26%. However, the main drawback of systems enhanced with

prefetching policy is that the users may not eventually request some prefetched objects. In such a case, the prefetching scheme increases the network traffic as well as the Web servers' load. Moreover, the cache space is not used optimally [13, 14, 15, 16]. Therefore, the prefetching approach should be designed carefully in order to overcome these limitations.

[17, 6] provided good survey papers of Web caching and replacement. Currently, many recent methods and algorithms have been suggested in web caching area. [18] presented a survey on applications of neural networks and evolutionary techniques in web caching. However, some intelligent web caching approaches have not been discussed and investigated in [18]. This paper differs about the previous survey papers. This study reviews extensively principles and some existing web caching and prefetching approaches. More significantly, this paper surveys the conventional and intelligent web caching techniques. Moreover, prefetching approaches are summarized and classified and with more focus on approaches based on data mining since they are the most common approaches in the web prefetching. Besides, this survey introduces some studies that discussed integration of web caching and web prefetching together. Thus, this review can be contributed in understanding and investigating new solutions of web caching and prefetching.

The remaining parts of this paper are organized as follows. Some principles of Web caching are presented in Section 2. The existing works of conventional and intelligent web caching techniques are also discussed in Section 2. Section 3 describes types and approaches of Web prefetching and surveys some of the representative techniques for each approach. Section 4 elucidates some studies that discussed integration of web caching and web prefetching together. Finally, Sections 5 and 6 concludes the paper and future work.

2 Web Caching

Web caching is one of the most successful solutions for improving the performance of Web-based system. In Web caching, the popular web objects that likely to be visited in the near future are stored in positions closer to the user like client machine or proxy server. Thus, the web caching helps in reducing Web service bottleneck, alleviating of traffic over the Internet and improving scalability of the Web system. The Web caching has three attractive advantages to Web participants, including end users, network managers, and content creators [13]:

a. The Web caching decreases user perceived latency.
b. The Web caching reduces network bandwidth usage.
c. The Web caching reduces loads on the origin servers.

2.1 Basic Types of Web Cache

Web caching keeps a local copy of Web pages in places close to the end user. Caches are found in browsers and in any of the Web intermediate between the user agent and the origin server. Typically, a cache is located in client (browser cache), proxy server (proxy cache) and origin server (cache server) [1, 19] as shown in Fig.1.

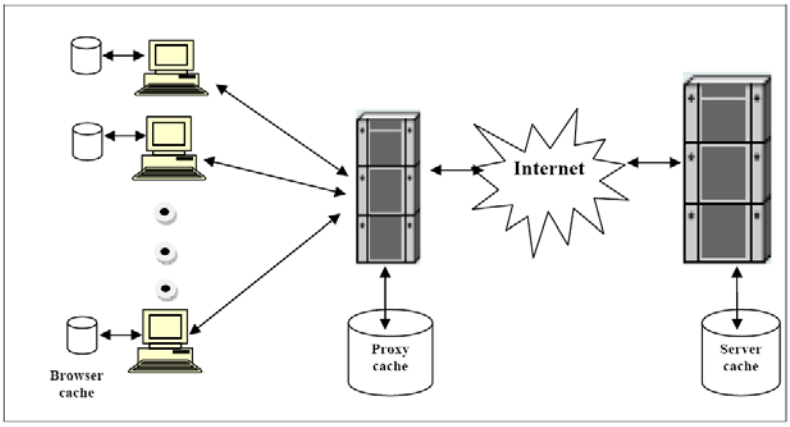

Fig.1: Locations of web cache

a. *Browser cache:* It is located in the client. The user can notice the cache setting of any modern Web browser such as Internet Explorer, Safari, Mozilla Firefox, Netscape, and Google chrome. This cache is useful, especially when users hit the "back" button or click a link to see a page they have just looked at. In addition, if the user uses the same navigation images throughout the browser, they will be served from browsers' caches almost instantaneously.

b. *Proxy server cache:* It is found in the proxy server which located between client machines and origin servers. It works on the same principle of browser cache, but it is a much larger scale. Unlike the browser cache which deals with only a single user, the proxies serve hundreds or thousands of users in the same way. When a request is received, the proxy server checks its cache. If the object is available, it sends the object to the client. If the object is not available , or it has expired, the proxy server will request the object from the origin server and send it to the client. The object will be stored in the proxy's local cache for future requests.

c. *Origin server cache:* Even at the origin server, web pages can be stored in a server-side cache for reducing the need for redundant computations or database retrievals. Thus, the server load can be reduced if the origin server cache is employed.

The proxy caching is widely utilized by computer network administrators, technology providers, and businesses to reduce user delays and to alleviate Internet congestion [20, 21, 7].

2.2 Web Caching Algorithms

Cache replacement policy plays an extremely important role in Web caching. Hence, the design of efficient cache replacement algorithms is required to achieve highly sophisticated caching mechanism [2, 22]. In general, cache replacement algorithms are also called Web caching algorithms [3].

As cache size is limited, a cache replacement policy is needed to handle the cache content. If the cache is full when an object needs to be stored, the replacement policy will determine which object is to be evicted to allow space for the new object. The optimal replacement policy aims to make the best use of available cache space, to improve cache hit rates, and to reduce loads on the origin server.

Most Web proxy servers still concern with traditional caching policies [4, 23]. These conventional policies are suitable in traditional caching like CPU caches and virtual memory systems, but they are not efficient in Web caching area [3]. The simplest and most common cache management approach is Least-Recently-Used (LRU) algorithm, which removes the least recently accessed objects until there is sufficient space for the new objects. LRU is easy to implement and proficient for uniform size objects, like in the memory cache. However, it does not perform well in Web caching since it does not consider the size or the download latency of objects [3]. Least-Frequently-Used (LFU) is another common web caching that replaces the object with the least number of accesses. LFU keeps more popular web objects and evicts rarely used ones. However, LFU suffers from the cache pollution in objects with the large reference accounts, which are never replaced, even if they are not re-accessed again.

The SIZE policy is one of the common web caching approaches that replaces the largest object(s) from cache when space is needed for a new object. Thus, the cache can be polluted with small objects which are not accessed again. To alleviate the cache pollution, [24] suggested Greedy-Dual-Size(GDS) policy as extension of the SIZE policy. The algorithm integrates several factors and assigns a key value or priority for each web object stored in the cache. When cache space becomes occupied and new object is required to be stored in cache, the object with the lowest key value is removed. When user requests an object p, GDS algorithm assigns key value K(p) of object p as shown in Eq.(1):

$$K(p) = L + \frac{C(p)}{S(p)} \tag{1}$$

where C(p) is the cost of fetching object p form server into the cache; S(p) is the size of object p; and L is an aging factor. L starts at 0 and is updated to the key value of the last replaced object. The key value K(p) of object p is updated using the new L value since the object p is accessed again. Thus, larger key values are assigned to objects that have been visited recently.

[24] proved that the GDS algorithm achieved better performance compared with some traditional caching algorithms. However, the GDS algorithm ignores the frequency of web object. [25] enhanced GDS algorithm by integrating the frequency factor into of the key value K(p) as shown in Eq.(2). The policy is called Greedy- Dual-Size-Frequency (GDSF).

$$K(p) = L + F(p) * \frac{C(p)}{S(p)} \tag{2}$$

where F(p) is frequency of the visits of object p. Initially, when p is requested by user, F(p) is initialized to 1. If p is in the cache, its frequency is increased by one. Table 1 depicts well-known replacement policies [3, 26].

Table 1: Conventional replacement policies

Policy	Brief description	Advantage	Disadvantage
LRU	The least recently used objects are removed first.	simple and efficient with uniform size objects, such as the memory cache.	ignores download latency and the size of web objects
LFU	The least frequently used objects are removed first.	simplicity	ignores download latency and size of objects and may store obsolete web objects indefinitely.
SIZE	Big objects are removed first	prefers keeping small web objects in the cache, causing high cachet hit ratio.	▪ sores small web objects even if these object are never accessed again. ▪ Low byte hit ratio.
GD-size	It assigns a key value to each object in the cache as equation(2.1) . Consequently, the object with the lowest key value is replaced when cache space becomes occupied	overcomes the weakness of SIZE policy by removing objects which are no longer requested by users.	not take into account the previous frequency of web objects
GDSF	It extends GDS algorithm by integrating the frequency factor into of the key value K(p) as shown in Eq.(2)	overcomes the drawback of GD-size	not take into account the predicted accesses in the future

In fact, there are few important factors (characteristics) of Web objects that can influence the replacement process [1, 27, 28]:

a. Recency: time of (since) the last reference to the object.
b. Frequency: number of requests to an object.
c. Size: size of the Web object.
d. Cost of fetching the object: cost to fetch an object from its origin server.
e. Access latency of object.

These factors can be incorporated into the replacement decision. Depending on these factors, the replacement policies can be classified into five categories [1, 6, 27]: Recency-based polices, Frequency-based polices, Size-based polices, Function-based polices and Randomized polices. Table 2 presents the replacement policy categories [6].

Table 2: Replacement policy categories

Category	Brief description	Available replacement policies	Representative policy	disadvantage
Recency-based policies	These policies use recency factor as the primary factor to remove Web object	LRU, LRU-threshold, LRU*, LRU-hot, LRU-LSC, SB-LRU, SLRU, HLRU, Pitkow/Recker, EXP1, value-aging, generational replacement.	LRU	not consider frequency, size and download latency of Web objects
Frequency-based policies	These policies use object popularity (or frequency count) as the primary factor to remove Web object	LFU, LFU-Aging, LFU-DA, Window-LFU, swLFU, AgedswLFU, a-Aging, HYPER-G.	LFU	not consider recency, size and download latency of Web objects
Size-based policies	These policies use object size as the primary factor for removing Web object.	SIZE, LRU min, partitioned caching, PSS, CSS, LRU-SP	SIZE	not consider recency, frequency and download latency of Web objects
Function-based policies	These policies associate each object in the cache with a utility value. The value is calculated	GD-Size, GDSF, GD*, PGDS, server-assisted cache replacement,	GD-SIZE	Choosing appropriate weights of factors is difficult task

	based on a specific function incorporating different factors such as time, frequency, size, cost, and latency, and different weighting parameters. The object with the smallest value is evicted first.	TSP, Bolot/Hoschka, MIX, M-Metric, HYBRID, LNCR-W3, LRV, LUV, LR, N-gram		
Randomized policies	These policies use randomized decisions to find an object for replacement. .	RAND, HARMONIC, LRU-C, LRU-S, randomized policies using utility functions.	HARMONIC.	More cumbersome to evaluate.

As it can be observed from Table 2, many Web cache replacement policies have been proposed for improving performance of web caching. However, it is difficult to have an omnipotent policy that performs well in all environments or for all time because each policy has different design rational to optimize different resources. Moreover, combination of the factors that can influence the replacement process to get wise replacement decision is not an easy task because one factor in a particular situation or environment is more important than other environments [1, 6]. Hence, there is a need for an effective approach to intelligently manage the web cache which satisfies the objectives of Web caching requirement. This is motivation in adopting intelligent techniques in solving Web caching problems.

2.3 Intelligent Web Caching Algorithms

Availability of Web access logs files that can be exploited as training data is the main motivation for adopting intelligent Web caching approaches [29, 1]. The second motivation, since Web environment changes and updates rapidly and continuously, an efficient and adaptive scheme is required in Web environment. The machine learning techniques can adapt to the important changes through a training phase. Although there are many studies in Web caching, enhancement of Web caching performance using intelligent techniques is still fresh. Recent studies have shown that the intelligent approaches are more efficient and adaptive to the Web caching environment compared to other approaches [29, 1].

In our previous work ICWCS [30], the neuro-fuzzy system has been employed to predict Web objects that can be re-accessed later. Although the simulation results have proven that ICWCS helps in improving the performance in terms of the hit ratio(HR), the performance in terms of the byte hit ratio (BHR) is not good enough since ICWCS did not take into account the cost and size of the predicted objects in the cache replacement process. Moreover, the training process requires long time and extra computational overhead.

In NNPCR[4] and NNPCR-2[23], ANN has been used for making cache replacement decision. An object is selected for replacement based on the rating returned by ANN. However, the objects with the same class are removed without any precedence between these objects.

An integrated solution of ANN as caching decision policy and LRU technique as replacement policy for script data object has been proposed in [26]. However, the most important factor in web caching, i.e., recency factor, was ignored in caching decision. [31] enhanced [26] using PSO for improving neural network performance. ANN has also been used in [3] depending on syntactic features from HTML structure of the document and the HTTP responses of the server as inputs. However, this method ignored frequency factor in web cache replacement decision. On another hand, it hinged on some factors that do not affect on performance of the web caching.

[32] proposed a logistic regression model(LR) to predict the future request. The web objects are classified into two categories; the object will be re-accessed in a given forward looking window (Y =1), or not (Y =0). In the LR model, the objects with the lowest re-access probability value are replaced first regardless of cost and size of the predicted object. In the similar way, [33] presented design of an intelligent predictor that uses back-propagation neural network instead of the LR model to predict the future access of web objects. However, he ignored the recency factor in Web caching. Table 3 summarizes the existing intelligent Web caching techniques and their limitations.

Table 3: Some intelligent web caching approaches

Title of the study	Author	Briefly description of study	limitations
Intelligent Client-side Web Caching Scheme Based on Least Recently Used Algorithm and Neuro-Fuzzy System	Ali & Shamsuddin (2009)[30]	Neuro-fuzzy system has been used for classifying web objects into either cacheable or uncacheable objects. Then, the trained neuro-fuzzy system has been employed with LRU Algorithm in cache to predict Web objects that can be re-accessed later.	▪The training process requires long time and extra computational. ▪Implemented in client level
Web proxy cache replacement scheme based on back-propagation neural network (NNPCR)	Cobb & Elaarag (2008)[4]	ANN has been used for making cache replacement decision. An object is selected for replacement based on the rating returned by ANN	The objects with the same class are removed without any precedence between these objects.
NNPCR-2	ElAarag & Romano(2009) [23]		

Intelligent Web Caching Architecture.	Farhan(2007)[26]	The author proposed an integrated solution of ANN as caching decision policy and LRU technique as replacement policy for script data object.	▪It ignored recency factor. ▪It is similar to SIZE- policies that suffer from cache pollution. ▪limited to VB.Net script
Intelligent web caching using neurocomputing and particle swarm optimization algorithm	Sulaiman et al. (2008)[31]	The author enhanced Farhan's approach[26] using PSO for improving neural network performance.	In training phase, getting of target output is not clear.
Web cache optimization with nonlinear model using object feature	Koskela et al. (2003)[3]	ANN has also been used depending on syntactic features from HTML structure of the document and the HTTP responses of the server as inputs.	▪Frequency factor is ignored in Web cache replacement decision. ▪It hinged on some factors that do not affect on Web caching
An Adaptive Web Cache Access Predictor Using Neural Network	Tian *et al.* (2002)[33]	The authors presented design of an intelligent predictor that used back-propagation neural network to improve the performance of Web caching by predicting the most likely re-accessed objects (based on history of Web accesses in a Web site) and then keep these objects in the cache.	It ignored the most important factor in Web caching (recency)
Logistic regression in an adaptive web cache	Foong et al. (1999)[32]	The author proposed a logistic regression model (LR) to predict the future request.	The objects with the lowest re-access probability value are replaced first regardless of cost and size of the predicted object

Although the previous studies produced better results compared to conventional web caching, the learning process can be time consuming. Moreover, they did not take into account the cost and size of the predicted objects in the cache replacement process.

[34] proposed a replacement algorithm based on fuzzy logic. This method ignored latency time in replacement decision. In addition, as fuzzy logic depends on expert knowledge, which may not be always available in Web caching, this method analyzed proxy workloads to identify parameters of each membership

function. This process is complicated and inaccurate. Moreover, this scheme is not adaptive with Web environment that changes and updates rapidly and continuously.

Several studies suggested cache replacement approaches based on evolutionary algorithms. [28] introduced the concept of applying genetic algorithms and evolutionary programming to web cache replacement process. A Web cache is modeled as a population of information objects and the aim is to improve the cache population regarding its reliability, and accessibility. Each cached object is considered to be an individual in the cache population. This approach prefers the strongest cached objects such that objects strength is determined by specific criteria related to object's staleness, access frequency and retrieval cost. [35] presented genetic algorithm based cache replacement policy. [35] defined the individuals as sets of cache objects with different ordering, while finesse function was defined as the sum of the cost of objects that tend to be removed from the cache. [36] presented also a model for applying evolutionary computing technique to the web cache replacement policy. Genetic algorithm and genetic programming were the two mechanisms used in their approach. They developed fitness function of genetic programming technique. Generally, the difficulties in the web cache replacement approaches based on evolutionary algorithms are in defining the individuals and fitness functions properly.

2.4 Performance Measures

There are standard metrics to analyze the efficiency and the performance of Web caching techniques. Hit Ratio (HR), Byte Hit Ratio (BHR) and Latency Saving Ratio (LSR) are the most widely used metrics in evaluating the performance of Web caching [4, 6, 3]. HR is defined as the percentage of requests that can be satisfied by the cache. Note that HR only indicates how many requests hit and does not indicate how much bandwidth or latency has been saved. BHR is the number of bytes satisfied from the cache as a fraction of the total bytes requested by user. LSR is defined as the ratio of the sum of download time of objects satisfied by the cache over the sum of all downloading time. Let N be the total number of requests (objects) and $\delta_i = 1$ if the request i is in the cache, while $\delta_i = 0$ otherwise. Mathematically, this can be written as follows:

$$HR = \frac{\sum_{i=1}^{N} \delta_i}{N} \tag{3}$$

$$BHR = \frac{\sum_{i=1}^{N} b_i \delta_i}{\sum_{i=1}^{N} b_i} \tag{4}$$

where b_i the size of the ith request ,

$$LSR = \frac{\sum_{i=1}^{N} t_i \delta_i}{\sum_{i=1}^{N} t_i} \quad (5)$$

where t_i is the time to download the ith referenced object from its server to the cache

3 Web Prefetching

Web prefetching is another very effective technique, which is utilized to complement the Web caching mechanism. The web prefetching predicts the web object expected to be requested in the near future, but these objects are not yet requested by users. Then, the predicted objects are fetched from the origin server and stored in a cache. Thus, the web prefetching helps in increasing the cache hits and reducing the user-perceived latency.

3.1 Types of Web Prefetching

The prefetching techniques can be implemented on server, proxy or client side. The client-based prefetching concentrates on the navigation patterns of a single user across many Web servers. On another hand, the sever-based prefetching concentrates on the navigation patterns of all users accessing a single website.

The proxy-based prefetching concentrates on the navigation patterns of a group of users across many Web servers. Thus, this approach can reflect a common interest for user's community. In other words, the prefetching contents can be shared by many users. Table 4 summarizes types of prefetching according to location.

Table 4: Types of prefetching based on location[37]

Prefetching Location	Data for Prediction Model	Advantages	Disadvantages
Client	Historical and current user requests	Easy to partition user session and realize personalized prefetching.	▪ not share prefetching content among users. ▪ needs a lot of network bandwidth.
Proxy	Proxy log and current user requests	▪ reflects common interests for a group of users. ▪ shares prefetching content from different servers among users.	not reflect common interests for a single Website from all users.
Server	Server log and current user requests	▪ records a single website access information from all users and better reflect all users' common interests.	▪ not reflect users' real browsing behavior . ▪ difficult to partition user session. ▪ needs additional communications between clients and servers for deciding prefetching content.

In recent years, considerable attention has been paid to proxy-based prefetching since it is more effective and more accurate in predicting the correlated pages of many Websites in the similar interests for more homogeneous users [38 , 29 ,14, 15].

3.2 Web Prefetching Approaches

The exiting prefetching algorithms can be classified into two main categories according to the data taken for prediction: content-based prefetching and history-based prefetching.

The content-based prefetching category predicts future user requests depending on the analysis of web page contents to find HTML links that are likely to be followed by the clients. [39] used ANN to predict future requests depending on keywords in anchor text of URL. The keywords extracted from web documents

were given as inputs to ANN that predict whether the URL needs to be prefetched or not. The content-based prefetching techniques are not recommended to be implemented at the server side because the high load for parsing every web page served can negatively affect the server service time [38].

The history-based prefetching category predicts future user requests depending on observed page access behaviors in the past. Many studies were interested in enhancing the techniques of this category. The algorithms of this category can be classified into four approaches: approach based on dependency graph, approach based on Markov model, approach based on cost function and approach based on data mining [37].

3.2.1 Dependency Graph Based Prefetching Approach

Dependency graph algorithm was firstly used in web prefetching in [40]. In this approach, dependency graph (DG) is constructed for prediction of next page. The dependency graph consists of nodes that represent Web pages, and arcs that indicate that the target node is accessed after the original node within a time window. Every arc is assigned to weight that represents the probability that the target node will be the next one. The dependency graph based prefetching approach predicts and prefetches the nodes whose arcs connect to the current accessed node and have weights higher than a threshold. Recently, [38] has presented double dependency graph (DDG) prediction algorithm that considers the characteristics of the current web sites to improve the performance achieved by web prefetching. It is based on DG algorithm, but it differentiates two classes of dependences: those among objects of the same page, and those among objects of different pages. Depending on this differentiation, DDG targets at predicting not only the following page but also its embedded objects.

Although the web prefetching approach based on DG can help in reducing the latency time, the network traffic is increased with this approach. Another drawback is that only pair dependencies between two web pages are examined, so the prediction accuracy is low [41].

3.2.2 Markov Model Based Prefetching Approach

Markov based prefetching approach is one kind of effective models for predicting the next Web page by matching the user's current access sequence with the user's historical Web access sequences. Suppose we have sequence of pages (states) $x_1 \to x_2 \to ... \to x_{k-1}, k \geq 2$. In the first-order Markov based approaches, the next state x_k depends only the current state of the chain x_{k-1}. If the occurrence of x_k depends on the latest occurred 2-length consecutive sequence $x_{k-2} \to x_{k-1}$, this is called a second order Markov. In general, if x_{k+1} depends on the pervious consecutively accessed k-length consecutive pages $x_1 \to x_2 \to ... \to x_k$, this is called a k-th order Markov Model.

In the lower order Markov models, the prediction is usually not accurate, since they do not look far enough into the past browsing behavior of users. As a consequence, higher order Markov models are used to obtain higher prediction accuracy, but at the expense of lower coverage. To overcome the problem of low coverage, [42] proposed the all k-th order Markov models. In this approach, the prediction is done by combining different order Markov models. Initially, predictions are attempted by using the highest order Markov. If no matching is found, the predictions are done by using lower order Markov. The order of Markov is decreased until the state is covered. Unfortunately, all-kth order Markov models exacerbate the problem of complexity since the states of all Markov models are added up.

[43] proposed Prediction-by-Partial-Matching (PPM) algorithm. PPM is one of the most successful Web prefetching techniques, which depends on high-order Markov mode. In PPM, prefetching decisions are made based on historical user requests in a Markov prediction tree. However, as more pages are requested and added into the history structure, its size will grow linearly. Several researchers proposed different methods to limit the size of the history structure. [42] modified PPM algorithm and proposed algorithm called LRS(Longest Repeating Sub-sequences). [44] also proposed a variant of the PPM algorithm called Popularity-based PPM (PB-PPM) to further reduce the size of the history structure and improve prediction accuracy. The LRS algorithm reduces the size of the history structure by only storing long branches formed by frequently requested pages, which are called Longest Repeating Sub-sequences. Since the majority of pages in Web servers are requested infrequently, keeping only frequently requested sequences would not give noticeable effect on the prefetching performance, but can significantly reduce the memory cost. The PB-PPM algorithm reduces the memory cost of PPM by only allowing the most frequently requested pages to become tree roots in the history structure. By doing so, the size of the history structure is reduced, while the overall prefetching performance is only slightly affected.

There are two shortcomings in both LRS and PB-PPM [29]. First, as the predictions are based solely on the history structure built in the last time period, the history structure used to make predictions can not include any new patterns discovered during the current time period. This lack of ability to adjust to the latest changes negatively affects the efficiency of these algorithms. For that reason, [45] presented an online PPM model to capture the changing patterns and fit the memory. Their approach only keeps the most recent requests using a sliding window. The prediction model in this approach combines entropy, prediction accuracy rate and the longest match rule. Second, LRS and PB-PPM are not designed to work on client machines or proxies. Unlike web servers, client machines and proxies can receive requests for pages on the entire Internet instead of a single Web server. Therefore, the difference in requesting frequency between pages is significantly smaller than the case when all pages came from the same

Web server, and, thus, removing slightly unpopular pages might result in a big loss of useful information. Therefore, these algorithms are not so useful for prefetching on client machines and proxies [29].

3.2.3 Cost Function Based Prefetching Approach

This approach prefetches some web objects into the cache depending on some factors such as popularity and lifetime of web pages. Several Web prefetching techniques based on cost function approach are presented in the literature. The algorithms under this approach are summarized as follows.

- *Prefetch by Popularity:* [46] suggested Top-10 prefetching approach, which keeps in cache the ten most popular objects from each web server.
- *Prefetch by Lifetime:* [47] suggested prefetching approach called Prefetch by Lifetime. Prefetch by Lifetime approach selects n objects that have the longest lifetime, and thus intends to minimize the bandwidth.
- *Good Fetch:* Prefetching approaches such as Good-Fetch[48] and APL [47] attempted to balance object popularity and object update rate, thereby they can get good hit ratio improvement at a modest expense of bandwidth.
- *Objective-Greedy Algorithms:* [49] proposed a family of prefetching algorithms, Objective-Greedy prefetching, wherein each algorithm greedily prefetches those web objects that give the highest performance as per the metric that it aims to improve.

3.2.4 Data Mining Based Prefetching Approach

The data mining-based prefetching approach can be classified into association rules-based prefetching approach and clustering-based prefetching approach.

3.2.4.1 Association Rules Based Prefetching Approach

Association rule discovers groups of homogeneous pages that are commonly accessed together in same user session. The user session is defined as the sequence of pages made by a single user during a visit to a particular site. In association rules, support and confidence are two important measures for benchmarking strength of any association rule. Support is defined as the discovery of frequent pages, while confidence is defined as the discovery of association rules from these frequent pages.

Since a user session may include many pages, it is very difficult to find matching rule antecedents. So, sliding window is used in association rule algorithms. The size of sliding window is iteratively decreased until an exact match with a rule antecedent. [50] presented four kinds of sequential association rules:

1) Subsequence rules: The association rules take into consideration the order of the pages.

2) Latest subsequence rules: The association rules take into consideration the most recent pages and order of the pages.
3) Substring rules: The association rules take into consideration the adjacency and the order of the pages.
4) Latest substring rules: The association rules take into consideration the most recent pages, the adjacency and the order of the pages.

Several researches used association rules techniques in web prefetching. [51] used an N-gram mining method and extended the GDSF function to improve hit ratios of caching and prefetching. [52] developed an A priori-based mining method to deduce a rule table for predicting and prefetching the highest document into a proxy buffer. However, too many useless rules are produced in the rule table. Recently, [14] have developed an access sequence miner to mine popular surfing 2- sequences with their conditional probabilities from the proxy log, and stored them in the rule table. Then, according to buffer contents and the rule table, a prediction-based buffer manager achieved appropriate actions such as document caching, document prefetching, and even cache/prefetch buffer size adjusting to achieve better buffer utilization.

The main problem associated with association rule mining-based approaches is that too many useless rules are produced in the rule table of the frequent pages with a high frequency. This causes inconsistent predictions, especially when the dataset is large [53]. In other words, the existing association rule based Web prefetching algorithms predict a particular page depending on patterns observed from all users' references. As a consequence, the prediction based on too many patterns is inaccurate and inefficient [53, 54].

3.2.4.2 Clustering Based Prefetching Approach

Clustering is utilized for finding similarity groups in data, called clusters, such that the data instances in the same cluster are similar to each other while data instances in different clusters are very different from each other. In other words, the intra-clusters distance is minimized, while the inter-clusters distance is maximized. Many studies used the clustering for improving the efficiency and scalability of the real-time personalization tasks [53, 55, 56, 57, 58].

The clustering approaches can be classified into model-based approaches and distance-based approaches. The model-based clustering approach often specifies the model type a priori. Moreover, the model structure can be determined by model selection techniques and parameters estimated using maximum likelihood algorithms, e.g., the Expectation Maximization (EM) [59]. The distance-based clustering groups the similar sessions together into clusters depending on a distance measure between pairs of pages. There are two types of distance-based clustering techniques: partitional clustering and hierarchical clustering. In the partitional clustering method, the data is partitioned into K groups or clusters. The most common partitional clustering method is k-means algorithm. In k-means

algorithm, the user needs to specify the number of clusters k in advance. The hierarchical method has a number of desirable properties, which make it popular. It builds a hierarchical set of nested clusters.

The clustering techniques can be utilized in either Web page clustering or user session clustering [53]. The page clustering is achieved by grouping pages having similar content as suggested in [60, 61]. On the other hand, the clustering techniques are used for grouping of the homogenous sessions in clusters. The user session clustering is widely used in many fields such as web page prediction, personalization and web prefetching.

[15] have introduced an algorithm for clustering inter-site Web pages in proxy, called clustWeb. The clustWeb algorithm has generated the clusters by partitioning the Web navigational graph using association rule mining techniques. This approach depended on the content of clusters. If each cluster contains a large amount of objects, it will result in an overaggressive prefetching policy. Moreover, grouping of IPs according to their domains may fail when IP changes dynamically.

[62] presented a prefetching scheme using ART1 network based clustering technique to group users (hosts) and then prefetch their requests according to the prototype vector of each group. This method may cause a considerable increase in network traffic since all objects in the same cluster are fetched. Moreover, the experiment only focused on the prefetching and did not address the interaction between web caching and prefetching.

[63] proposed three-dimensional cube model that represented sequential user session data and allowed different data analyses to be easily conducted, such as summary statistical analysis, clustering and sequential association analysis. Then, [63] applied the clustering results using the data cube model to the problem of integrated web caching and prefetching. The authors used a variant of the k-means algorithm for clustering, so the clustering requires both a prior knowledge about number of clusters and initializing cluster centroids randomly.

[64] presented page rank-based prefetching mechanism for clustered web page accesses. The pages linked to a requested page are ranked, and the rank is used to determine the pages to be prefetched. Moreover, page rank is used as a replacement mechanism, i.e. those pages with the lowest rank are replaced with new ones. The authors heuristically defined any Web directory with 200 or more files under it as a candidate cluster.

The approaches based on data mining and Markov model are the most common approaches in the web prefetching. However, the Markov model based algorithms are not so useful for prefetching on the proxy server and negatively affected with any new patterns discovered during the current time period [29]. In recent years, the data mining techniques have been utilized effectively in web proxy prefetching area [29, 14, 15].

3.3 Performance Measures

There are several metrics used to analyze the efficiency and the performance of Web pre-fetching techniques. The following measures are frequently used for evaluating performance of web prefetching [65, 14] :

- *Precision (Pc):* The ratio of prefetch hits to the total number of objects prefetched as shown in Eq. (6).

$$Pc = \frac{\#\,PrefetchHits}{\#\,Prefetches} \tag{6}$$

- *Byte Precision (Pc_B):* Byte precision measures the percentage of prefetched bytes that are subsequently requested. It can be calculated by replacing the number of prefetched objects with their size in bytes in Eq. (6).

- *Recall (Rc):* The ratio of prefetch hits to the total number of objects requested by users as shown in Eq. (7).

$$Rc = \frac{\#\,PrefetchHits}{\#\,Requests} \tag{7}$$

- *Byte Recall (Rc_B):* Byte recall measures the percentage of demanded bytes that were previously prefetched and subsequently requested.

- *Traffic Increase (ΔTr):* The bytes transferred through the network when prefetching is employed divided by the bytes transferred in the non-prefetching case.

- *Latency per page ratio:* The latency per page ratio is the ratio of the latency that prefetching achieves to the latency with no prefetching. The latency per page is calculated by comparing the time between the browser initiation of an HTML page GET and the browser reception of the last byte of the last embedded image or object for that page. This metric represents the benefit perceived by the user, which will be better as lower its value is.

4 Integrating Web Caching and Web Prefetching

Web proxy caching and prefetching are the most popular techniques which play a key role in improving the Web performance. Since the Web proxy caching exploits the temporal locality and the web prefetching utilizes the spatial locality of the Web objects, Web proxy caching and prefetching can complement each other. Thus, combination of the caching and the prefetching helps on improving hit ratio and reducing the user-perceived latency. However, if the web caching and prefetching are integrated inefficiently, this might cause increasing the network traffic as well as the Web servers' load. Moreover, the cache space is not used optimally [13, 14, 15, 16]. Therefore, the prefetching approach should be designed carefully in order to overcome these limitations.

Basically, the web prefetching requires two steps: anticipating future pages of users and preloading them into a cache. This means the web prefetching involves also the caching. However, the web caching and prefetching are addressed separately by many researchers in the past.

It is important to take into consideration the impact of these two techniques combined together. Few studies were discussed integration of web caching and web prefetching together. [12] studied effect of a combination of caching and prefetching on end user latency. They concluded that the combination of web caching and prefetching can potentially improve latency up to 60%, whereas web caching alone improves the latency up to 26%. [51] suggested an application of web log mining to obtain web-document access patterns and used these patterns to extend the well-known GDSF caching policies and prefetching policies. [66] proposed cache replacement algorithm called IWCP for integrating Web caching and Web prefetching in client-side proxies. They formulated a normalized profit function to evaluate the profit from caching an object either a nonimplied object or an implied object according to some prefetching rule.

Similar to [39], [13] used ANN in both prefetching policy and Web cache removal decision. This approach depended on the keywords of URL anchor text to predict the user's future requests.The most significant factors (recency and frequency) were ignored in web cache replacement decision. Moreover, since the keywords extracted from web documents were given as inputs to ANN, applying ANN in this way may cause extra overhead on the server.

[67] presented a compact set of algorithms for integrating web caching and prefetching for wireless local area network, including sequence mining based prediction algorithm, context-aware prefetching algorithm and profit-driven caching replacement policy. [68] proposed a framework for combining Web caching and prefetching on mobile environment. They proposed hybrid technique (Rough Neuro-PSO) based on combination of ANN and PSO for classification Web object. Then, rules from log data are generated by Rough Set technique on the proxy server. In prefetching side, prefetching approach based on XML is

suggested to be implemented on mobile device to handle communication between client and server.

In summary, the previous works integrated the web prefetching with caching; However, these approaches are still not efficient enough [29]. Most previous works used association rules for prefetching approach, which are inaccurate and inefficient since these works predict a particular page depending on patterns observed from all users' references [53, 54]. Moreover, these approaches employ the conventional replacement policies that are not efficient in web caching.

5 Conclusion

Web caching and prefetching are two effective solutions to lessen Web service bottleneck, reduce traffic over the Internet and improve scalability of the Web system. The Web caching and prefetching can complement each other since the web caching exploits the temporal locality for predicting revisiting requested objects, while the web prefetching utilizes the spatial locality for predicting next related web objects of the requested Web objects. Thus, combination of the web caching and the web prefetching doubles the performance compared to single caching. This paper reviews principles and some the existing web caching and prefetching approaches. Firstly, we have reviewed principles and existing works of web caching. This includes the conventional and intelligent web caching. Secondly, types and categories of prefetching have presented and discussed briefly. Moreover, the history-based prefetching approaches have been concentrated and discussed with review of the related works for each approach in this survey. Finally, this survey has presented some studies that discussed integration of web caching and web prefetching together.

6 Future Work

In recent years, machine learning techniques have been applied successfully in data mining tasks such as classification and clustering. Artificial immune system and support vector machine are two common intelligent techniques for achieving data mining tasks. In the future, we will concentrate on constructing an intelligent web proxy caching and prefetching approach based on support vector machine and artificial immune system.

ACKNOWLEDGEMENTS.

Authors would like to thank Research Management Centre (RMC), Universiti Teknologi Malaysia, for the research activities and *Soft Computing Research Group* (SCRG) for the support and incisive comments in making this study a success.

References

[1] H.T. Chen, *Pre-fetching and Re-fetching in Web caching systems: Algorithms and Simulation*, Master Thesis, TRENT UNIVESITY, Peterborough, Ontario, Canada(2008).

[2] T.Chen, "Obtaining the optimal cache document replacement policy for the caching system of an EC Website", *European Journal of Operational Research*.181(2),(2007), pp. 828. Amsterdam.

[3] T. Koskela, J. Heikkonen, ,and K. Kaski, (2003). "Web cache optimization with nonlinear model using object feature", *Computer Networks journal, elsevier* , 43(6), (2003), pp. 805-817.

[4] J. Cobb, and H. ElAarag, "Web proxy cache replacement scheme based on back-propagation neural network", *Journal of System and Software*, 81(9), (2008), pp. 1539-1558.

[5] R. Ayani, Y.M. Teo, and Y.S. Ng, "Cache pollution in Web proxy servers", *International Parallel and Distributed Processing Symposium (IPDPS'03)*, 22-26 April 2003, pp.7.

[6] A.K.Y. Wong, " Web Cache Replacement Policies: A Pragmatic Approach", *IEEE Network magazine*, 20(1), (2006), pp.28–34.

[7] C. Kumar and J.B. Norris, "A new approach for a proxy-level Web caching mechanism", *Decision Support Systems, Elsevier*, 46(1), (2008), pp.52-60.

[8] I. R. Chiang, P. B. Goes, and Z. Zhang, "Periodic cache replacement policy for dynamic content at application server", *Decision Support Systems, Elsevier*, 43 (2), (2007), pp. 336- 348.

[9] H.k. Lee, B.S. An, and E.J. Kim, "Adaptive Prefetching Scheme Using Web Log Mining in Cluster-Based Web Systems", *2009 IEEE International Conference on Web Services (ICWS)*, (2009), pp.903-910.

[10] L. Jianhui, X. Tianshu, Y. Chao. "Research on WEB Cache Prediction Recommend Mechanism Based on Usage Pattern", *First International Workshop on Knowledge Discovery and Data Mining(WKDD)*, (2008), pp.473-476.

[11] A. Abhari, S. P. Dandamudi, and S. Majumdar , "Web Object-Based Storage Management in Proxy Caches", *Future Generation Computer Systems Journal* , 22(1-2), (2006). pp. 16-33.

[12] T. M. Kroeger, D. D. E. Long, and J. C. Mogul, "Exploring the bounds of web latency reduction from caching and prefetching", *Proceedings of the USENDC Symposium on Internet Technology and Systems*, (1997), pp. 13-22.

[13] U. Acharjee, *Personalized and Artificial Intelligence Web Caching and Prefetching.* Master thesis, University of Ottawa,Canada(2006).

[14] Y.f. Huang and J.M. Hsu, "Mining web logs to improve hit ratios of prefetching and caching". *Knowledge-Based Systems*, 21(1), (2008), pp. 62-69.

[15] G. Pallis, A. Vakali, and J.Pokorny, "A clustering-based prefetching scheme on a Web cache environment", *Computers and Electrical Engineering*, 34(4), (2008). pp.309-323.

[16] W. Feng, S. Man, and G. Hu, "Markov Tree Prediction on Web Cache Prefetching", Software Engineering, Artificial Intelligence(SCI), Springer-Verlag Berlin Heidelberg, 209,(2009). pp. 105–120.

[17] J. Wang, " A Survey of Web Caching Schemes for the Internet", *ACM Comp. Commun. Review*, 29(5), Oct. 1999, pp. 36–46.

[18] P. Venketesh, R. Venkatesan, "A Survey on Applications of Neural Networks and Evolutionary Techniques in Web Caching", *IETE Tech Rev*, 26, (2009), pp.171-80 .

[19] B. Krishnamurthy, J. Rexforrd, *Web protocols and practice:HTTP/1.1, networking protocols, caching and traffic measurement*. Addison-Wesley, (2001).

[20] C. C. Kaya, G. Zhang, Y. Tan , and V.S Mookerjee, "An admission-control technique for delay reduction in proxy caching", *Decision Support Systems*, 46(2), (2009), pp 594-603.

[21] C. Kumar, "Performance evaluation for implementations of a network of proxy caches", *Decision Support Systems*, 46(2), (2009), pp. 492-500.

[22] H. Che, Y. Tung, and Z. Wang, "Hierarchical Web Caching Systems: Modeling, Design and Experimental Results", *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, 20(7), (2002), 1305-1314.

[23] H. ElAarag and S. Romano, "Improvement of the neural network proxy cache replacement strategy", *Proceedings of the 2009 Spring Simulation Multiconference,(SSM'09)*, San Diego, California, (2009), pp: 90.

[24] P. Cao and S. Irani, "Cost-Aware WWW Proxy Caching Algorithms", *In USENIX Symposium on Internet Technologies and Systems*, Monterey, CA(1997).

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

[26] Farhan, *Intelligent Web Caching Architecture. Master thesis. Faculty of Computer Science and Information System*, UTM University, Johor, Malaysia, (2007).

[27] S. Podlipnig, and L. Böszörmenyi, "A survey of Web cache replacement strategies", *ACM Computing Surveys*, 35(4), (2003), pp. 374-39.

[28] A. Vakali, "Evolutionary Techniques for Web Caching", *Distributed and Parallel Databases*, Springer Netherlands, 11(1), (2002), 93-116.

[29] Q. Liu, *Web Latency Reduction With Prefetching*, PhD thesis, University of Western Ontario, London(2009).

[30] W. Ali, and S.M. Shamsuddin, "Intelligent Client-side Web Caching Scheme Based on Least recently Used Algorithm and Neuro-Fuzzy System", *The sixth International Symposium on Neural Networks(ISNN 2009), Lecture Notes in Computer Science (LNCS)*, Springer-Verlag Berlin Heidelberg , 5552, (2009), pp. 70–79.

[31] S. Sulaiman, S.M. Shamsuddin, F. Forkan, and A. Abraham, "Intelligent web caching using neurocomputing and particle swarm optimization algorithm", *Proceedings of the 2008 Second Asia International Conference on Modelling & Simulation (AMS 08), IEEE Computer Society*, (2008), pp. 642-647.

[32] A. P. Foong, Y.-H. Hu, and D. M. Heisey, "Logistic regression in an adaptive web cache", *IEEE Internet Computing*, 3, (1999), 27-36.

[33] W. Tian, B. Choi, and V.V. Phoha,"An Adaptive Web Cache Access Predictor Using Neural Network". *Proceedings of the 15th international conference on Industrial and engineering applications of artificial intelligence and expert systems: developments in applied artificial intelligence, Lecture Notes In Computer Science(LNCS)*, Springer- Verlag London, UK 2358, (2002).450-459.

[34] Calzarossa, and G. Valli, "A Fuzzy Algorithm for Web Caching", *SIMULATION SERIES Journal*. 35(4), (2003), 630-636.

[35] Y. Chen, Z. Li, and Z. Wang, "A GA-Based Cache Replacement Policy", *In Proceedings of the third International Conference on Machine Learning and Cybernetic*, (2004).

[36] K. Tirdad, F. Pakzad, and A. Abhari, "Cache replacement solutions by evolutionary computing technique", *Proceedings of the 2009 Spring Simulation Multiconference, San Diego, California, Society for Computer Simulation International*,(2009), pp. 1-4.

[37] B. Zhijie, G. Zhimin, and J. Yu, "A Survey of Web Prefetching", *Journal of computer research and development*, 46(2), (2009), pp. 202-210.

[38] J. Domenech, J. A. Gil, J. Sahuquillo, and A. Pont, "Using current web page structure to improve prefetching performance", *Computer Network Journal*, 54(9), (2010), 1404-1417.

[39] T.I. Ibrahim, and C-Z. Xu, "A Keyword-Based Semantic Pre-fetching Approach in Internet News Services", *IEEE Trans. Knowl. Data Eng.*, 16 (5), (2004). 601-611.

[40] V. N. Padmanabhan, and J. C. Mogul , "Using predictive prefetching to improve World Wide Web latency", *Computer Communication Review*, 26(3), (1996). pp. 22–36.

[41] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos, "A data mining algorithm for generalized Web prefetching", *IEEE Trans on Knowledge and Data Engineering* , 15 (5) ,(2003). Pp. 11552-1169.

[42] J. Pitkow and P. Pirolli, "Mining longest Repeating subsequences to Predict World Wide Web Surfing", *Proceedings USENIX Symposium on Internet Technologies and Systems(USITS'99),*(1999).

[43] T. Palpanas and A. Mendelzon, "Web prefetching using partial match prediction*", In Proceedings of the 4th International Web Caching Workshop. San Diego, USA*, (1999).

[44] X. Chen and X. Zhang, "Popularity-based PPM: An effective web prefetching technique for high accuracy and low storage*", In Proceedings of the International Conference on Parallel Processing*, (2002), pp. 296-304.

[45] Z. Ban, Z. Gu, "An online PPM prediction model for web prefetching", *Proceedings of the 9th annual ACM international workshop on Web information and data management*, Lisbon, Portugal, ACM, (2007), pp. 89-96.

[46] E. P. Markatos and C. E. Chronaki , "A Top-10 approach to prefetching on the Web ", *Proceedings of INET'98 Geneva, Switzerland*, (1998), pp. 276-290.

[47] Y. Jiang, M.Y Wu, and W. Shu, "Web prefetching : Costs , benefits and performance", *Proceedings of the 11th International World Wide Web Conference, New York, ACM*, (2002).

[48] A. Venkataramani, P. Yalagandula, and R. Kokku , "The potential costs and benefits of long-term prefetching for content distribution", *Computer Communications*, 25(4) ,(2002). pp. 367-375.

[49] W. Bin, and A. D. Kshemkalyani, " Objective-greedy algorithms for long-term Web prefetching", *Proceedings of Third IEEE International Symposium on Network Computing and Applications(NCA 2004),* (2004).

[50] Q. Yang, T. Li, and K. Wang, "Building association-rule based sequential classifiers for web-document prediction", *Journal of Data Mining and Knowledge Discovery*. 8(3), (2004), 253–273.

[51] Q.Yang, H. Zhang, and T. Li, "Mining web logs for prediction models in WWW caching and prefetching", *Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining*, (2001), pp. 473–478.

[52] B. Lan, S. Bressan, B.C Ooi, and K.L. Tan, "Rule-assisted prefetching in web-server caching", *Proceedings of the 9th ACM International Conference on Information and Knowledge Management*, (2002), pp. 504–511.

[53] F. Khalil, J. LiAn, and H. Wang, "Integrated model for next page access prediction", *Int. J. Knowledge and Web Intelligence*, 1(1-2), (2009), pp. 48-80(33).

[54] J. Xiao, Y. Zhang, X. Jia, and T. Li, "Measuring similarity of interests for clustering Web-users", 12th Australasian Database Conference(ADC), (2001), pp.107-114.

[55] M. Rigou, S. Sirmakesses, and G. Tzimas, "A method for personalized clustering in data intensive web applications", *APS'06, Denmark*, (2006), pp. 35–40.

[56] N. K. Papadakis, and D. Skoutas, "STAVIES: A system for information extraction from unknown web data sources through automatic web warpper generation using clustering techniques", *IEEE Transactions on Knowledge and Data Engineering* , 17(12), (2005), 1638–1652.

[57] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White, "Model-based clustering and visualization of navigation patterns on a web site", *Data Mining and Knowledge Discovery*. 7(4), (2003), 399–424.

[58] G. Adami, P. Avesani, and D. Sona, "Clustering documents in a web directory", *WIDM'03, USA* , (2003), pp. 66–73.

[59] S. Zhong, and J. Ghosh, "A unified framework for model-based clustering", *Machine Learning Research*, 4, (2003), pp. 1001–1037.

[60] N. Tang and R. Vemuri, "An artificial immune system approach to document clustering", *Proceedings of the Twentieth ACM Symposium on Applied Computing. SantaFe, New Mexico, USA* , (2005) ,918-922.

[61] L. Xu, H. Mo, K. Wang, and N. Tang, " Document Clustering Based on Modified Artificial Immune Network", *Rough Sets and Knowledge Technology. G. Wang, J. Peters, A. Skowron and Y. Yao, Springer Berlin / Heidelberg*, 4062, (2006), 516-521.

[62] S. K. Rangarajan, V. V. Phoha, K. Balagani, R. R. Selmic, and S.S. Iyengar, "Web user clustering and its application to prefetching using ART neural networks", *IEEE Computer*, (2004).

[63] Q. Yang, J.Z. Huang, and M. Ng, "A data cube model for prediction-based web prefetching", *Journal of Intelligent Information Systems*, 20(1), (2003), pp.11-30.

[64] V. Safronov, and M. Parashar, "Optimizing Web servers using page rank prefetching for clustered accesses", *Information Sciences* , 150(3-4), (2003), pp.165-176.

[65] J. Domenech, A. Pont-Sanju´an, J. Sahuquillo, and J. A. Gil, "Evaluation, Analysis and Adaptation of Web Prefetching Techniques in Current Web", *Web-based Support Systems*, Springer, London, (2010). 239-271.

[66] W. Teng, C. Chang, and M. Chen, "Integrating Web Caching and Web Prefetching in Client-Side Proxies", *IEEE Transaction on Parallel and Distributed Systems*, 16(5), (2005), pp 444-455.

[67] B. Jin, T. Sihua, C. Lin, X. Ren, and Yu. Huang, "An Integrated Prefetching and Caching Scheme for Mobile Web Caching System". *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing(SNPD)*, (2007).

[68] S. Sulaiman, S. M Shamsuddin, and A. Abraham, " Rough Neuro-PSO Web caching and XML prefetching for accessing Facebook from mobile environment", *World Congress on Nature & Biologically Inspired Computing(NaBIC 2009), (2009).*