Abstract—Cache memory plays a central role in improving the performance of web servers, especially for big data transmission, which response time is constrained. It is necessary to use an effective method, such as web cache. Browsers’ cache has a significant role according to less bandwidth use, response time and traffic load as well as beneficial if the internet connection is slow. Due to the space limitations, modern browsers companies attempt to use a method to store a great number of web objects and to advance the effectiveness of web browsers. Many scientists have been working to discover and recommend various techniques for this purpose. This study consequently reviews the recent likelihood probabilistic methods, to figure out how browsers store web objects in their caches, and which methods are used to load more speedily and to store a great number of web objects. The comparison between numerous browsers performed to pick and recommend the utmost one for usage. The result has shown that each browser using RI (Ratio Improvement) has powerful performance; to be discussed later. It has proposed using Google Chrome browser because web objects are placed in its cache through the RI technique that correlated with browsers’ effectiveness.

Index Terms—Browser Catch, Web Content Mining, Overloading, Probability Techniques. Cache Improvement.

I. INTRODUCTION

A cache is a high-speed memory that temporarily saves data and content from a website, for example texts, images and some other web objects, so that the next time the site is visited, that content is displayed much faster. It helps a web page load much faster for a better user experience. When data and content get stored into a cache memory, it’s referred to as “caching.”. The web cache servers are widely deployed in many places throughout the Web [1]. For web cache, there are three different web cache patterns: client-side caching [2], [3], server-side caching [4], and proxy caching [5]. Client-side caching refers to caches that user-side stores users web browsers page addresses and other information, and it can reach the specified server with the information. However, it is only for single user. Server-side caching refers to establishing the cache on the web server side. The purpose of it is decreasing the number of requests for the server, so it can reduce the server load [6]. Proxy caching usually serves as the middle connection of user and center servers [7]. When the user sends a request to the server via a proxy server, server will response the data to the user according to the original request path. During this process, proxy server will decide whether to store a copy in its cache or not because this data may be requested in the future. Fig. 1 shows the overview of web and browser cache. Probabilistic techniques for web

![Fig. 1. Overview of web and browser cache [28]](http://dx.doi.org/10.24018/ejers.2020.5.7.1976)

Browsers are a common way to explore the intent for web objects in the context of probabilities, for loading web pages faster and to lessen network traffic loads, objects with high probability will be used rather than which have low probability.

Objects with low probability will be set in a queue or will be deleted to free up space for newly signed objects. Web browsers use such techniques to store their cached content over a while. Hence, when it is requested later the same content will be accessible easily and faster. A web browser is an application program widely used by internet users to retrieve information from the World Wide Web (WWW). The web cache has an important role to search for information on the internet, and it is one of the best

Published on July 6, 2020.
A. S. Kakar, Department of Information Systems, Faculty of Computer Science, Kandahar University, Afghanistan. (e-mail: sattarka@gmail.com)
M. S. Rohei, Department of Information Systems, Faculty of Computer Science, Kandahar University, Afghanistan. (e-mail: sadiqlahah52@gmail.com)
mechanisms for web pages fast loading and storing web content to lessen over-load problems for the users. Common types of web cache which are used for augmenting browser over-loading issues are proxy cache, browser cache, inverted proxy cache, and transparent proxy cache. While users are visiting a web page at the same time its links and contents are stored in the browser cache, on the spot storing of this web page information will be supportive to load the same web page quickly from the first visit by the same user [8]. Browser caching is a typical cost-effective method of improving the performance of the WWW. The cache setting of any modern web browser such as Google Chrome, Mozilla Firefox, Internet Explorer, Netscape, and Safari can be manipulated by the user. The cache is more helpful for the users, mostly in case of moving backward from a web page or going to a new webpage by clicking a link for revisiting the page recently visited by the user. Furthermore, if the same navigation images throughout the browser are used by the user, it will be served by browsers cache almost instantaneously [9].

Web system scalability can be enhanced by the browser cache using deferent possible techniques for objects storage to lessen bandwidth usage, response time, and network traffic. The browser cache contains limited space, then the development of efficient techniques is needed to manage web cache content effectively. The purpose of this study is to review existing probability techniques used to store contents in the browser cache and comprise of them to select the best. Besides, the background for the web cache is studied to understand the core mechanisms of the browser cache and to highpoint the optimum probability for contents stored in it which can be valuable in the development of efficient cache management techniques and can advise future guidelines.

II. ART STATE ON PROBABILITY TECHNIQUES

The foremost concept behind the probability techniques used for content storing in browser cache is an effective way of mining structured data such as text, image, and multimedia across the web page, weblogs and browsers. These facts are attracting the attention of researchers to concentrate on web pages mining, web usage, and log services mining for content. We have stated review on some probabilistic techniques such as Probabilistic Latent Semantic Analysis (PLSA), Hidden Markova Model (HMM), fuzzy logic and Least Recently Used replacement policy (FL & LRU) and Ratio Improvement (RI) which are used for storing contents in the browser cache and their performance. We choose these techniques because modern browsers’ companies try to improve browser performance and capacity of browser cache to store a great number of contents as well as we perform comparison among them, which enable us to easily decide which one is better and give us guidelines about developing effective technique in the future.

A. Probabilistic Latent Semantic Analysis (PLSA)

PLSA is an old statistical technique developed by T. Hofmann [10] which deals with the co-occurrence of data, this technique is used for web usage mining [11] to find out the hidden semantic relationship between two co-occurrence factors such as web users and web objects in cache environments. The relationship was probabilistic which discovers the classification of web users in performed tasks at cache environments. When users navigate for web pages in browsers, all the web objects are stored in browser cache instantaneously and for the next navigation of same web page, related contents are already in browsers’ cache so the users can view the page contents much faster without any delay or loading of network traffic. Processes for storing page viewed objects and using the same objects next time depends on user’s navigation session matrix with page views, provides such a matrix:

\[
\text{Set of } n \text{ pages views } \rightarrow P = \{p_1, p_2, p_3, \ldots, p_n\}
\]

\[
\text{Set of } m \text{ users } \rightarrow U = \{u_1, u_2, u_3, \ldots, u_m\}
\]

\[
PU = m \times n
\]

Consequently, PU represents the occurrences and duration of pages that are viewed in relevant sessions [12]. PLSA discovered the relationship between these two co-occurrences of data in the form of observation of hidden factor such that

\[
K = \{k_1, k_2, k_3, \ldots, k_n\} \text{ access to web resources depends on each observation in a particular session, for making relationship through PLSA join probability is vital}
\]

\[
Pr\left(\frac{p_{ij}}{pu}\right) = \sum_{k=1}^{n} pr\left(\frac{p_{ij}}{2k}\right) \times pr\left(\frac{pu}{2k}\right)
\]

The requirement to characterize the web users' performed tasks in a particular session at the cache environment. The whole process is called page identification and user sectorization respectively that provides user interest in web cache environments; As well as reduces traffic delay for browsing information for the next time. As shown in Fig 2 the page weight and user session where Kij has joined the probability or relationship model of page Pij in a particular Uij user session. [13]

1) Advantages of PLSA for web caching

PLSA technique is much important for making a relationship between users and the pages that are browsed by these users. The main advantage of PLSA is to generate probabilities that make the relationship between two co-occurrence actions such as visiting page and performed
tasks by the user in a particular session and driving that relationship on the base of probabilistic inferences among users, pages, and between both of them. Hence, this intellectual framework provides a flexible interest of users in the web cache environment. PLSA also has taken advantage of the development of new algorithms or techniques for web cache and has provided space dimension interpretation, which may help in reducing consumption space in browser cache [14]. PLSA interprets space dimensions by characterizing documents in a corpus, where each document is characterized in probabilistic space [15]. The coordination of a document consists of word unigram probabilities, and sub-space constructed in resulting new document is projected to space in a probabilistic manner. Word probabilities for this document will be mapped from original space representation with high dimensionality to new representation with low dimensionality latent semantic space. This process is called dimension reduction, which is extremely helpful in consumption space in browser cache that is much better for cache due to its space limitation. This interpretation of PLSA is seldom applied to large data set with its high computational complexity in computer memory cache. It also takes advantage of adding the latent variable to PLSA to generate a more sophisticated latent semantic structure than the flat structures. PLSA can use the usual methods to prevent overfitting, which can lead to models that are more general.

2) **Deficiency of PLSA**

PLSA is not a procreative technique for modern web browser cache. Modern web browsers cache initially has two types of caches, instant and durable cache [16]. A web object is stored in instant cache and the web objects that are visited more than the pre-specified threshold value will be moved to the durable cache. Other objects are removed by the LRU algorithm, as the instant cache is full, then fuzzy logic is used to improve durable cache replacement decisions. Then PLSA not fully employed to make a relationship between web users and pages that are browsed by these users, as well as PLSA can't remove uncatchable objects from cache environments. PLSA has a major drawback in a computer cache environment that is the theatrical consumption of computing recourse, in terms of both execution time and internal memory [17]. This drawback limits the practical application of the technique only to document collections of modest size, but for web, cache making relationships between web users and browsed pages is a computational procedure that may lead to time consumption. PLSA uses many parameters for making any model of relationship which leads to high complexity models. Due to this issue, modern browsers use other different techniques to create a relationship between users and browsed pages and to improve their cache to store a great amount of data as well as to reduce response time, bandwidth usage, and loading of network traffic.

**B. Hidden Markova Model (HMM)**

A Hidden Markova Model (HMM) is a tool for representing probability distributions over sequences of observations and is a statistical, timeline-based, and intelligent technique for web cache management, which is most effective in static web page contents but less effective in dynamic contents. This model is working on hidden transitive probability [18]. If the user visits the web pages, the most visited URLs or (higher probability URLs) are listed in the cache and less-visited URLs (fewer probability URLs) are not cleared in the cache. By using HMM these are transferred to a new list, and the list is based on time. If the time is expired, then the URLs and its objects are cleared from the cache. As shown below in Fig. 3. HMM also provides mapping documents or contents in multi-dimension space that extremely helps in reducing space for cache which is the main necessity of browsers cache due to its space limitations.

**1) Advantages of HMM**

Easily computational model than PLSA, an HMM is usually used for situations where you have discrete "hidden" states, mapping document in multi-dimensional space and using time slots between probability transition. As well as used for reducing time consumption for user request in browser amounts of cash and computing resources in computer memory. Modern browsers use HMM for the cache of their browsers, which have the capability for a great amount of data in their cache as well as response time is much faster than previous versions. HMM is especially used in web mining, structural analysis, and pattern discovery.

**2) Deficiency of HMM**

HMM is less effective in dynamic content mining. The dynamic data of web pages hitting in web cache required special techniques called hit and byte hit ratio improvements that will be discussed later. HMM only performs the acts of transitive probability between webpages that have static contents. HMM also has a large number of unstructured parameters. Therefore, the transition matrix ends up being very large, and this leads to severe overfitting. A solution to this is to use other types of HMMs, such as factorial or hierarchical HMMs.

**C. Fuzzy logic and LRU replacement policy**

Fuzzy logic is similar to Boolean logic dealing with truth condition that denotes the extent to which a proposition is true. Fuzzy logic is much useful when a noise environment is established which handles truth decisions for replacement. Fuzzy logic works in three variable stages: the variable that describes the first visit of the web page is called the input variable or "fuzzified" stage and the second is set of fuzzy control rules which are applied to the input variable called fuzzy control stage. After applying fuzzy control, rules

---

**Fig. 3. Web page architecture based on HMM**

---

**DOI:** http://dx.doi.org/10.24018/ejers.2020.5.7.1976
produce output which is called the output variable or “defuzzified” stage. Fuzzy controls check the input variable for noise occurrences and applied specific probability rule to remove the object with lower noise probability from the input variable and then give it as an output [19, 13]. Fuzzy logic is applied in a noisy environment to remove some objects from the cache and to free up space for incoming objects. When users browse some specific web pages, instant cache directly stores all web objects for short time, and the objects that are visited more such as a hyperlink or pdf file will be relocated to the durable cache for a longer time, this facilitates the easy access for users for next time visiting. If the durable cache is full and saturates, then fuzzy logic is employed to divide each of the objects inside the durable cache whether cacheable or uncacheable objects. The uncacheable objects that contain the oldest state will be removed initially from the durable cache to free up space for incoming objects. For pages, if the cache is full one or more pages will be evicted by fuzzy logic form cache to free up space. As cache size is limited, a cache replacement policy is needed to handle the cache contents. One of the most popular cache replacements policy is LRU for cache management in a web browser. LRU specifies the set of documents at any point to hold in the cache when needed free space in the cache. LRU immediately replaces the page or document that has not been used for the longest period [20]. So Fuzzy logic is employed. If all objects are cacheable then LRU policy is used for replacement purposes that optimize the performance of browser cache.

1) LRU threshold

An estimated value for the replacement of cache content based on time interval, a threshold value is calculated from the current size of the cache for the low and high watermarks [21, 18]. When the current size of the cache is closer to a high watermark, then the threshold gets a higher value and LRU purges more objects from cache otherwise fewer objects. The probabilistic calculation of threshold values for LRU depends on the number of documents in the cache. Higher probability means to purge more objects from cache, medium probability means to purge a half number of objects, and low probability means to purge a smaller number of objects from the cache. As shown below in Fig. 4.

![Fig. 4. Probabilistic model for LRU replacement [21].](image)

2) LRU based policies

There are many policies developed based on the study of LRU, we find that all policies have the same performance for web browser cache. We have started a review of all policies in tabular form to know about the usage of policies as well as to reduce time complexity for selecting good policy among them [22]. Fig. 5 displays time complexity for each policy, and Table I displays categories and usage of LRU types.

As have been seen that LRU is a very good replacement policy but some LRU base policies have a complex data structure that is not applicable for high performance of web browsers because they use more bandwidth and consuming the time for caching web objects in the browser cache.

3) Advantages of LRU

The main advantage of LRU, compare to PLSA is the provision of LRU threshold, which calculates the probability of purged document in the browser cache, and for cacheable objects, LRU policy is used for replacement purpose which enhances the performance of browser cache as well as LRU can lessen time complexity for caching document in browser cache.

![Fig. 5. Category of LRU base policies and their representation [15].](image)

| TABLE I: OVERALL PERFORMANCE FOR EACH CATEGORY [22] |
|--------------------------------------------------|
| Recently based | Features | Users Interest |
| LRU, S,-LRU, HLRU, SB-LRU | The recently referenced object will be referenced once again shortly. | Many users interest the same objects in the same unit of time. |
| Size-based | Features | Users Interest |
| LRU, LRU-SP | Removing larger object to free up space for small incoming objects | Users want to be having access to information-based web sites |
| Randomized | Features | Users Interest |
| LRU-C, LRU-S | Complex data structure and high computation overhead are not necessary | Computer memory, as the system has limited processing time and memory resources. |
4) Deficiency of LRU

The major disadvantage of the LRU is that for certain web objects, it only considers the time of the last reference and has no indication of the number of references [23]. If the time of this object is expired, then it is removed automatically from the cache and for the next request, most of the bandwidth usage is needed for caching. LRU has the issue of low overheating and cache pollutions for a larger document to be cached. That’s why there is no efficient approach to achieve a complete goal. To explicitly exploit both access frequency and object size in LRU, there are more advanced extensions of LRU to get rid of the above issues. Moreover, for their performance to operate on a large document and for the browser cache performance to store a large amount of data, another technique has been proposed which is the improvement of hit and byte hit ratio. Browsers' caches performance depends on connected clients to it. Therefore, the higher probability of hitting documents in browser cache provides a quick loading facility of web content for users. However, higher probability is used by modern browsers for caching documents to quickly respond for users' requests and for the increased probability to hit a documents' both caches (client and proxy) cooperate.

Cooperative caching is a technique used in browsers caching to improve the efficiency of information access by reducing access latency and bandwidth usage. When browsing hypermedia documents from a remote URL, if no proxy servers are built in the passing path, the documents will be transmitted site by site from the remote URL. This is not only increasing the response time of requests but also wastes network bandwidth. Thus, proxy servers need to be built in the passing path to cache frequently accessed documents in the buffer. Cache replacement policy plays a significant role in response time reduction by selecting suitable subsets of items for eviction from the cache. We have already made a review of the existing cache replacement mechanism proposed for cooperative caching in the browser’s cache. A good replacement mechanism is the improvement of hit and byte hit ratio for a document.

D. Ratio Improvement (RI)

The fourth technique that is used extremely in a modern browser is the improvement of hit and byte hit ratio for the evaluation of the best performance of web browsers caches [24].

Hit ratio: Is the ratio of the total amount of objects found in the cache to the total number of objects requested.

\[
HR = \frac{\sum_{i=1}^{N} ri}{N}
\]

Where \( ri \) is a state of the object. It is a boolean value, where '0' represents miss and '1' represents hit. \( N \) is the sum of requests. When all requests are finished, we could count the hit times by summing up all \( ri \) value.

Byte hit ratio: it is defined as the number of bytes found in the cache divided by the total number of bytes requested within the observation period.

\[
BHR = \frac{\sum_{i=1}^{N} ri \times si}{\sum_{i=1}^{N} Si}
\]

Where \( ri \) has the same meaning as that described in Hit Rate. The parameter \( si \) is the size of object \( i \). We record the size of each cache object. When all requests are finished, we can calculate the total object size, and then we use the above formula to calculate the value of BHR.

1) Average Access Latency

The AAL’s (Average Access Latency) value represents the average access time during the whole dataset. AAL can be described by the following equation:

\[
AAL = \frac{\sum_{i=1}^{N} Ti}{SUM}
\]

Where the parameter \( Ti \) is the time when the program finished accessing dataset \( i \) and the parameter \( SUM \) is the number of datasets. The AAL is smaller, the performance of the cache replacement algorithm is better. It means users can get objects they requested without waiting for a long time.

2) Probabilistic analysis model for RI

The formula for

\[
BHR = \sum_{i \in R} \frac{H(i)}{\sum_{i \in R} H(i) \times F(i)}
\]

where:

\( H(i) \): corresponding to the object hits at time \( t \)
\( F(i) \): corresponding to the request of object \( i \)

We still need to make an assumption modelling for cache hit ratio improvement:

- \( a \) value range for
  \( H(i) \) (for instance \( \{u1, u2, u3\} = [\{lower, average, upper\}] \))
  - transition probability function
  \( P(H(i)|F(i)) \) (for instance \( P(ui|ui) = \alpha=2/3 \) and \( P(ui+1|ui) = 1-\alpha \))

If we assume object or document’s hit with high cache hit ratio to respect of time then an output probability function

\[
P(H(i)|F(i)) \text{ (i.e. in above notations } P \text{ (success } [ui] \text{ for all its objects) [25].}
\]

Advanced caching algorithms lead to a higher cache hit ratio but at different rates. Moreover, hit ratio improvement with the size of the cache is going to be larger size while \( x \) is the original size of web cache or computer memory cache. Doubling the size \( 2x \) would significantly increase the hit ratio, and tripling \( 3x \) would further increase the hit ratio. A high cache hit ratio greatly improves the browsing experience while reducing costs in terms of energy, bandwidth, and network traffic, on the other hand, byte hit ratio improvement reduces the latency of caching documents between client and web servers and saves bandwidth of the
network.

3) A probabilistic model for caching document and RI

Let \( x \) = travel time of document, \( y \) = length of document, and \( Pr \) = the probability of caching documents

\[
Pr(y) = \frac{5}{6}, \text{if } y = 102 \text{ (bytes), then } \\
Pr(y) = \frac{1}{6}, \text{if } y = 104 \text{ (bytes)} \\
Pr(x/102) = \frac{1}{2}, \text{if } x = 10^{-2} \text{ (sec) and } \\
Pr(x/102) = \frac{1}{6}, \text{if } x = 1 \text{ (sec), and for } \\
Pr(x/104) = \frac{1}{2}, \text{if } x=1, \text{otherwise } \frac{1}{6}, \text{if } x=100.
\]

Respectively from the above assumption, we could figure out if the document has a larger size, then the caching probability is less than the document that has a smaller size. On the other hand, the traveling time of \( x \) document depends on its length of \( y \). Hence, we have seen probabilistically the time for caching document in browsers’ cache through RI where byte hit ratio for a smaller sized document is more than large-sized documents [26]. Consequently, the large-sized document should be removed by replacement policy from web caching. When a single large document is evicted from the cache, space is freed up for many smaller documents. Besides, the majority of requests from smaller sized documents are very high. Finally, the relationship results between hit and byte hit ratio also describes that “small files seem to be more prevalent than the large ones, but it is mainly a consequence of the choices related to cache replacement policies [27]. The relationship holds by HR and BHR.

\[
\frac{HR}{BHR} = \frac{\text{Mean of Response Size}}{\text{Mean Hit Response Size}}
\]

then the result with a high percentage (asset is cached properly) and with low percentage (no caching). Computers memory cache \( HR = BHR \) because all the responses (hit and misses) have the same size. For a web caching system, it depends on relationship result between HR and BHR. This means that the probability for caching smaller size documents is more than the probability for caching greater files that is most effective in the strengthening of the content storing in the cache. This strength is evaluated by the consideration of the cached objects’ retrieval rates as well as their frequency of access and their “freshness”.

4) Advantages of RI

LRU threshold has to calculate the probability for purge documents from cache. If a large document has lower probability then according to the LRU threshold it will have remained in the cache which is space consuming. The advantage of RI is providing a higher probability for caching of smaller size documents. If the document has larger size, then the caching probability is less mean that document should be removed by LRU which helps for providing more space for incoming objects.

5) Deficiency for RI

Maximizing the hit and byte hit ratio at a high level takes the drawback of overheating and overlapping of content in the web cache. It might have casuals’ effect on work loading and loss of some contents’ references, where for the user’s next request cache will not have the reference of requested objects. Maximizing the hit and byte hit ratio at a high level will get rid of some web pages due to low bandwidth, and users need to re-access the page with a different frequency which leads to work loading and postponement of access to information.

III. RESULT AND DISCUSSION

In this study, we have reviewed the four probabilistic techniques such as PLSA, HMM, FL-LRU and RI for storing content in a web cache, specifically in the browser cache. We have explored their advantages, deficiencies, and usages for both computer memory cache and web cache. We have attained our objective that how web objects (documents) are cached in a web cache, especially in the browser cache and how they store these web objects. Due to limited space of web browser cache we have studied how we can improve the capacity of our browser cache to store a great amount of data, as a result, we have considered that different ways are used by different techniques to free up space in the browser’s cache. LRU is used to calculate the probability to purge documents from the cache where higher probability means to purge more documents from cache. RI is used to calculate the probability for caching web objects, the web objects that have a large size have a lower probability for caching then automatically removed through LRU to free up space. Furthermore, for the performance of browsers cache, we have stated the following comparisons based on the study of four techniques to know which technique is much better. Modern browsers use powerful techniques nowadays to lessen response time, bandwidth usage, and loading of network traffic at the time of browsing information across WWW.

A. Experiment Results

In this section techniques are compared on HR, BHR and AAT on large data set of entity-based objects and represent the performance of each technique, as shown below.

Based on Fig. 6 as constructed on ETD (Entity based Data) which contain web objects, based on RI, and we know which technique is most powerful and has a significant role in the performance of browsers. The first graph shows the HR (Hit Rate) performance of the techniques, which IR has the best HR, the second graph show the BHR (Byte Hit Rate) of the mentioned techniques, which again IR has the best BHR and finally the third graph shows the AAT (Average Access Time) among the techniques and shows that IR has the less AAT. Old versions of browsers were used with old techniques where they had the drawback of low storage, overhead, low response time, bandwidth usage and much work loading. For instance, the old version of the Internet Explorer browser had all the above-mentioned issues for browsing information across WWW. Modern browsers’ companies try to improve the performance of their browsers and the capacity of browsers caches to store a large amount of content. They use the latest proposed techniques such as improvement of hit and byte hit ratio and upright replacement policies to remove the old objects from the cache and to free up space for new objects. That is why, modern browsers’ caches can add extra extensions and plugins to browsers as well as great download history for
web objects such as text, images and multimedia files.

Google company has already improved both hit and byte hit ratio. In addition to that, they have used best replacement policies for their Google Chrome browser which has the capability of storing a large amount of data and the performance of browsers regarding lessening of response time, bandwidth usage and a load of network traffic is very high. For that reason, it is alive on an extraordinary of 2 billion devices across the globe.

IV. CONCLUSION AND FUTURE CHALLENGES

The comparison review has been conducted on the four different techniques utilizing probability for web browsers’ caching. Besides, the document caching methods in browser cache have been highlighted. The study in a nutshell:

The techniques used for web caching are PLSA, HMM, FL-LRU, RI. All of the stated techniques use following probabilistic methods such as comparable probabilities, transitive probabilities and probabilistic algorithms for caching documents. Two methods are generally used to free up space for incoming new objects in web cache:

(a) Removal and replacement algorithm. (b) RI.

Throughout the comparisons, it is revealed and proved that the RI (Ratio Improvement) is most attractive and useful for caching web objects and for the performance of web browsers.

Our future work is to propose new probabilistic techniques and algorithms for web browsers’ caching to improve the cache capacity for a large amount of data as well as for efficient browsing across the WWW.

REFERENCES

[1] D. Singh, S. Kumar, and S. Kapoor. “An explore view of Web caching techniques,” Int. J. Adv. Eng. Sci., vol. 1, no. 3, pp 38–43, 2011.
[2] S. M. Shamsuddin and W. A. Ahmed. “Integration of least recently used algorithm and neuro-fuzzy system into client-side Web caching,” Int. J. Comput. Sci. Secur., vol. 3, no. 1, pp. 1–15, 2009.
[3] W.-G. Teng, C.-Y. Chang, and M.-S. Chen. “Integrating Web Caching and Web prefetching in client-side proxies,” IEEE Trans. Parallel Distrib. Syst., vol. 16, no. 5, pp. 444–455, May 2005.
[4] K. Kim and D. Park. “Reducing outgoing trafﬁc of proxy cache by using client-cluster,” J. Commun. Netw., vol. 8, no. 8, pp. 330–338, 2006.
[5] X. Wu, H. Xu, X. Zhu, and W. Li. “Web cache replacement strategy based on reference degree,” in Proc. IEEE Int. Conf. Smart City/SocialCom/SustainCom, Chengdu, China, Dec. 2015, pp. 209–212.
[6] S. Hiranpongsin and P. Bhattacharosol. “Integration of recommender system for Web cache management,” Maejo Int. J. Sci. Technol., vol. 7, no. 2, pp. 232–247, 2013.
[7] T. Ma et al., “KDVE, A k-degree anonymity with vertex and edge modification algorithm,” Computing, vol. 70, no. 6, pp. 1336–1344, 2015.
[8] Aswini, S., G. ShanmugaSundaram, and P. Iyappan. “Optimizing the Performance in Web Browsers Through Data Compression: A Study,” Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics, Springer India, 2016.
[9] Ali, Wahed, Siti Mariyam Shamsuddin, and Abdul Samad Ismail. “A survey of Web caching and prefetching,” Int. J. Advance. Soft Comput. Appl. 3.1 (2011): 18–44.
[10] T. Hofmann. Probabilistic latent semantic analysis. In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 1999
[11] Jin, Xin, Yanzan Zhou, and Bamshad Mobasher. “Web usage mining based on probabilistic latent semantic analysis.” Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2004.
[12] Cooley, R., Mobasher, B., & Srivastava, J. (1999). Data preparation for mining world wide web browsing patterns. Knowledge and information systems, 1(1), 5-32.
[13] Xu, Guandong, Yanchun Zhang, and Xiaofang Zhou. “Using probabilistic latent semantic analysis for Web page grouping.” 15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications (RIDE-SDMA’05). IEEE, 2005.
[14] Hofmann, Thomas. “Unsupervised learning by probabilistic latent semantic analysis.” Machine learning 42.1–2 (2001): 177–196.
[15] Akita, Yuya, and Tatsuya Kawahara. “Language model adaptation based on PLSA of topics and speakers.” INTERSPEECH. 2004.

Fig. 6. Techniques comparison based on HR, BHR and AAT

DOI: http://dx.doi.org/10.24018/ejers.2020.5.7.1976
[16] Muralidhar, K., and Dr. N. Geethanjali. “Improving the performance of the browsers using fuzzy logic.” International Journal of Engineering Research and Technology 3.1 (2012).

[17] Wan, Raymond, Vo Ngoc Anh, and Hiroshi Mamitsuka. “Efficient probabilistic latent semantic analysis through parallelization.” Asia Information Retrieval Symposium. Springer Berlin Heidelberg, 2009.

[18] Badodia, Sujit Kumar, Sachin Patel, and Rakesh Pendit. "Dynamic Web Cache Management and Browsing Performance." International Journal of Computer Applications 70.14 (2013).

[19] Calzarossa, Maria Carla, and Giacomo Valli. "A fuzzy Algorithm for web caching," SIMULATION SERIES 35.4 (2003): 630-636.

[20] Mookerjee, Vijay S., and Yong Tan. "Analysis of a least recently used cache management policy for Web browsers." Operations Research 50.2 (2002): 345-357.

[21] Vakali, A. I. "LRU-based algorithms for Web cache replacement." International conference on electronic commerce and web technologies. Springer Berlin Heidelberg, 2000.

[22] Wong, Kin-Yeung. "Web cache replacement policies: a pragmatic approach." IEEE Network 20.1 (2006): 28-34.

[23] Cheng, Kai, and Yahiko Kambayashi. “LRU-SP: a size-adjusted and popularity-aware LRU replacement algorithm for web caching.” Computer Software and Applications Conference, 2000. COMPSAC 2000. The 24th Annual International. IEEE, 2000.

[24] Arora, Kapil, and Dhawaleswar Rao Ch. "Web Cache Page Replacement by Using LIRU and LFU Algorithms with Hit Ratio: A Case Unification." ICSIT) International Journal of Computer Science and Information Technologies 5.3 (2014): 3232.

[25] Alexandre Halm (http://math.stackexchange.com/users/177651/alexandre-halm), How to calculate a changing probability situation based on possible improvement? Can this problem not be solved precisely, just estimated?, URL (version: 2014-10-13)

[26] Bertsekas, Dimitri P., and John N. Tsitsiklis. Introduction to probability. Vol. 1. Belmont, MA: Athena Scientific, 2002.

[27] Murta, Cristina Duarte, and Virgílio AF Almeida. “Using performance maps to understand the behavior of Web caching policies.” Internet Applications, 2001. WIAPP 2001. Proceedings. The Second IEEE Workshop on. IEEE, 2001

[28] Keycdn (2018). "Web cache Guid ". 2020, from https://www.keycdn.com/support/web-cache.

Abdul Sattar Kakar is born 20-12-1982 in Kandahar, Afghanistan. Kakar has a Master’s degree in Computer Science from TU-Berlin, Berlin, Germany.

He is Assistant Lecturer in Kandahar University at the faculty of Computer Science in the department of Information Systems, and is currently the Dean of the Faculty of Computer Science.

Mr Muhammad Sadiq Rohei is a lecturer in Kandahar University, Kandahar since May, 2014. He graduated from Nangarhar University, Afghanistan in 2013, and has completed MSc in Applied Computing from University of Malaya, Malaysia. He has 6+ years of experience in the field of web and database development, MIS specialist, managing and coordinating government projects with national and international organization such as UNICEF, MAP2U SDN BHD, and ZiiK-berlin program at Kandahar University. He has also worked with Systems Innovations (Si3). His research interest is in IoT, sensing Technologies, data Analytics and pervasive computing.