Mining Web Logs with PLSA Based Prediction Model to Improve Web Caching Performance

Chuibi Huang
Department of Automation, USTC
Key laboratory of network communication system and control
Hefei, China
Email: hcb@mail.ustc.edu.cn

Jinlin Wang, Haojiang Deng, Jun Chen
Institute of Acoustics, Chinese Academy of Sciences
National Network New Media Engineering Research Center
Beijing, China
Email: {wangjl, denghj, chenj}@dsp.ac.cn

***Abstract*— Web caching is a well-known strategy for improving the performance of web systems. The key to better web caching performance is an efficient replacing policy that keeps in the cache popular documents and replaces rarely used ones. When coupled with web log mining, the replacing policy can more accurately decide which documents should be cached. In this paper, we present a PLSA based prediction model to predict the user access patterns and interest to extend the well-known NGRAM-GDSF caching policy. Extensive experiments are conducted on the publicly available web logs datasets. The result shows that our approach gets better web-access performance.**

***Index Terms*—PLSA, web caching, web caching algorithm, prediction model, web log mining**

I. INTRODUCTION

As the Internet and user scale are growing at a very rapid rate, the performance requirements of web systems become increasingly high. Web caching is one of the most successful solutions for improving the performance of web systems. The core idea of web caching is to maintain the popular web documents that likely to be revisited in near future in a cache, such that the performance of web system can be significantly improved since most of later user requests can be directly replied from the cache. Lying in the heart of web caching algorithms is the cache replacement policy. To improve the performance of web caching, researchers have proposed a number of cache replacement policies [1, 21], many of which have been covered in the comprehensive surveys by [16]. These traditional algorithms take into account several factors and assign a key value or priority for each web document stored in the cache. 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 [14, 15].

Due to these constraints, 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 the Web caching algorithms. Another motivation to intelligent Web caching algorithms is the availability of web access logs files, which can be exploited as training data. In a few previous studies, the intelligent approaches have been applied in web caching algorithms or other web service [2, 11, 12, 13, 17, 18, 19, 20]. These studies typically build prediction model by training the web logs. By making use of the prediction model, the caching algorithms become more efficient and adaptive to the web caching environment compared to the traditional web caching algorithms. However, these studies didn't take into account the user access patterns and interest when building the prediction model. Since the users are the source of all the web access actions, it is necessary to build a prediction model which can well modeling the user access patterns and interest. In this paper, we use the web access logs to train a probabilistic latent semantic analysis (PLSA) based prediction model for user access patterns and interest to extend NGRAM-GDSF [6]. Based on the model, we can obtain user's topics of interest and mine rules for future access prediction. We then proposed the PLSA-based NGRAM-GDSF caching algorithm, which incorporate our PLSA-based prediction model into NGRAM-GDSF to improve its web caching performance. We also improve the PLSA model to get the more accurately topics. The experiment shows that our PLSA-based prediction model indeed improve the system performance over NGRAM-GDSF.

The organization of the paper is as follows. In the next section, we review the related work in web caching. In Section 3, we give a brief introduce of the PLSA model.

doi:10.4304/jcp.8.5.1351-1356

Data process is described in Section 4. Then in Section 5, we introduce the formal PLSA-based prediction model and show how it integrates with the caching algorithms. Experiment results are presented in Section 6. Finally, we conclude our paper in Section 7.

II. Related Work

Web caching plays a key role in improving web systems performance. The heart of web caching is so-called "replacement policy", which measure the popularity of all the previously visited documents, keeps in the cache those popular documents and replaces rarely used ones. The basic idea of the most well-known caching algorithms is to assign each document a key value computed by factors such as size, frequency and cost. Using this key value, we could rank these documents according to corresponding key value. When a replacement is to be carried, the lower-ranked documents will be evicted from the cache. Among these key value-based caching algorithms, GDFS [1] is the most successful one. It assigns a key value to each document in the cache as $K(h) = L + F(h) * C(h)/S(h)$, where L is an inflation factor to avoid cache pollution, $C(h)$ is the cost to fetch h, $F(h)$ is the past occurrence frequency of h and $S(h)$ is the size of h.

Availability of web access logs files that can be used as training data promotes the emergence of intelligent web caching algorithms [2, 3, 4, 11, 12]. In [11], the neuro-fuzzy system has been employed to predict web objects that can be re-accessed later. [12] proposes a logistic regression model to predict the future request. However, these algorithms didn't make use of the web logs to mine the useful information that can reveal the user behavior pattern. Then a lot of study has been conducted to use web log mining to improve the performance of web caching. Web log mining extract useful knowledge from large-scale web logs for future research and application. In [5], Pitknow and Pirolli studied the pattern extraction techniques to predict the web use's access path. An n-gram model to predict future requests was proposed in [7]. [8] has proposed sequential data mining for web transaction data, but they didn't apply the algorithm in caching. In [6], Qiang Yang discusses an integrated model by combining association-based prediction and the well-known GDSF caching algorithm (NGRAM-GDSF) in a unified framework. They first train the web logs to build a set of association rules, and then apply these rules to give prediction of future visits for each session. However, the prediction didn't take into account the interest of current user. In contrast, in this paper, we propose a PLSA-based prediction model, by which we can make predictions based on the current active user's interest, to improve the association prediction algorithm in [6].

III. The PLSA Model

The PLSA model [9] was originally developed for topic discovery in a text corpus, where each document is represented by its word frequency. The model assumes that, under the texts we observed there is another latent level: the topic level. A document has a certain probability distribution on a variety of topics, and similarly, topics also have different distribution on a set of words. Therefore, PLSA introduces a latent topic variable $z_k \in \{z_1, \ldots, z_K\}$ between the document $d_i \in \{d_1, \ldots, d_N\}$ and the word $w_j \in \{w_1, \ldots, w_M\}$. Then the PLSA model is given by the following generative scheme:

(1) Select a document d_i with probability $p(d_i)$.
(2) Pick a latent topic z_k with probability $p(z_k|d_i)$.
(3) Generate a word w_j with probability $p(w_j|z_k)$.

As a result, we generally get an observation pair (d_i, w_j), while the latent topic variable z_k is hidden. This generative model can be expressed by the following probabilistic model:

$$P(w_j, d_i) = P(d_i) P(w_j \mid d_j) \tag{1}$$

$$P(w_j \mid d_i) = \sum_{k=1}^{K} P(w_j \mid z_k) P(z_k \mid d_i) \tag{2}$$

Expectation maximization (EM) algorithm [10] is applied to learn the unobservable probability distribution $P(z_k|d_i)$ and $P(w_j|z_k)$ from the complete dataset. The log-likelihood of the complete dataset is:

$$L = \sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) \log P(d_i, w_j) \tag{3}$$

$$\propto \sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) \log \sum_{k=1}^{K} P(w_j \mid z_k) P(z_k \mid d_i) \tag{4}$$

Where $n(d_i, w_j)$ is the number of occurrences of word w_j in document d_i. In order to maximize the log-likelihood function L, we should first initial the PLSA probability model parameters $P(z_k|d_i)$ and $P(w_j|z_k)$ with random number, then perform iterative calculations by alternating implementation of the E-step and M-step. When the change of L is less than a threshold value, we stop the iterative calculations. In E-step, the priori probability of z is calculated:

$$P(z_k \mid d_i, w_j) = \frac{P(w_j \mid z_k) P(z_k \mid d_i)}{\sum_{l=1}^{K} P(w_j \mid z_l) P(z_l \mid d_i)}. \tag{5}$$

In M-step, the following formulas are used to re-estimate the model parameters:

$$P(w_j \mid z_k) = \frac{\sum_{i=1}^{N} n(d_i, w_j) P(z_k \mid d_i, w_j)}{\sum_{i=1}^{N} \sum_{j=1}^{M} n(d_i, w_j) P(z_k \mid d_i, w_j)} \tag{6}$$

$$P(z_k \mid d_i) = \frac{\sum_{j=1}^{M} n(d_i, w_j) P(z_k \mid d_i, w_j)}{\sum_{j=1}^{M} n(d_i, w_j)} \quad (7)$$

IV. Data Process

In this section, we present our approach to establish a PLSA-based prediction model on a large-scale web log. Our goal is to model user access patterns to get the user's interest, based on which we can predict future requests for each current active user.

A. Extracting Embedded Objects

HTML documents consist of variety of embedded web objects, such as images, audio and video files. Exactly, they contain the linkage structure of these web objects. References to embedded objects are usually preceded by their HTML container. Therefore, from the web logs we can see, they always appear as a burst of requests from the same user shortly after an HTML access. If an object is observed that its request always come immediately after access to certain HTML documents, it can be labeled as its containers.

Generally web users are only interested in HTML page, however, they know nothing about the linkage structure information. Therefore, we deal with HTML documents and embedded objects differently. While building PLSA-based model to get the interest of user, we do not take embedded objects into considerations. Instead, we just associate them to their corresponding HTML containers. After extracted from web logs, these embedded objects are stored in HTML-OBJECT Hash Table, by which we can get the embedded objects of a HTML documents quickly.

B. Building User-html Matrix

After filtering the embedded objects, only HTML document remain in a request sequence. When we build the user-html matrix, the session list has to be generated first. Here, the generation of session list consists of three major steps. First, the records of the web logs are sorted according to the access time. Then, web set a reasonable session time threshold. We assume that the duration of any session won't exceed the session time threshold. Finally, according to the threshold, we sequentially build the sessions for each user. Then we can get the user-html matrix (see Fig.1) by traversing the session list. Each row in the matrix represents a user and $n(u_n, h_m)$specifies the number of times the html file h_m accessed by user u_n.

$$\begin{bmatrix} n(u_1,h_1) & \cdots & n(u_1,h_m) & \cdots & n(u_1,h_M) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ n(u_n,h_1) & \cdots & n(u_n,h_m) & \cdots & n(u_n,h_M) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ n(u_N,h_1) & \cdots & n(u_N,h_m) & \cdots & n(u_N,h_M) \end{bmatrix}$$

Figure 1. The user-html matrix.

V. PLSA-Based Prediction Model

A. Overview

Although the PLSA model was originally developed for topic discovery in text corpus, it has been applied in many scientific fields, such as multimedia, data mining and image recognition. Here we apply the PLSA model to improve web caching algorithm. However, in the application process, we found that there is a problem with the PLSA model: it doesn't consider the user similarity when calculate the model parameters. In order to more accurately obtain the latent topics, we have made some improvements on the iterative calculations to address the similarity between the users. We introduce a new similarity layer in PLSA model such that the topic distributions of the given user can be updated by that of users similar to him. After building the improved PLSA model, we can made prediction for each current active user based on the users' interest which is derived from the model. Further, we make use of the prediction model to improve the caching algorithm described in [6].

As shows in Fig.2 we depict an overview of the improved PLSA model in our application scenario. Given the user $u_i \in \{u_1, \ldots, u_N\}$, the HTML file $h_j \in \{h_1, \ldots, h_M\}$ and the latent topic $z_k \in \{z_1, \ldots z_K\}$, we adopt the same generative scheme as that of PLSA. In addition, we introduce a similarity layer between user and HTML file.

1. Select a user u_i with probability $P(u_i)$
2. Pick a latent topic z_k with probability $P(z_k|u_i)$
3. Access a HTML file h_j with probability $P(h_j|z_k)$
4. $P(z_k|u_i)$ can be updated by that of the similar users: $P(z_k|u_i) = \sum_{l=1}^{N} P(z_k|u_l)P(u_l|u_i)$
5. $P(u_l|u_i)$ is similarity between u_l and u_i, it also can be considered as the conditional probability

The user similarities are parameterized by the user similarity matrix S which described in the following section. As we have incorporated the user similarities into the PLSA model, the similar users can have similar topic distributions and we will get more accurate latent topics than the original PLSA model.

B. Users Similarites

With the user-html matrix introduced as Fig.1 shows, we compute the user similarity matrix S by cosine similarity. For each pair of users in the matrix, we first compute their cosine similarity as follows:

$$Sim_{ih} = \frac{\vec{u_i} \cdot \vec{u_l}}{\left|\vec{u_i}\right| \cdot \left|\vec{u_l}\right|} \quad (8)$$

where $\vec{u_i}$ is the i-th user and represented by the i-th row in the user-html matrix. Then we can get a similarity matrix S where $S_{il} = Sim_{il}$. Further, we should normalize the matrix S such that its row adds up to 1.

$$S_{i1} = \frac{S_{i1}}{\sum_{h=1}^{N} S_{ih}} \quad (9)$$

Therefore, each element of S can be considered as the conditional probability $P(u_l|u_i)$ and the topic distribution of a given user can be updated by the topic distributions of the users that are similar to the given user.

$$P(z_k \mid u_i) = \sum_{l=1}^{N} P(z_k \mid u_l) P(u_l \mid u_i) \quad (10)$$

C. Parameter Estimating

According to the maximum likelihood principle, we estimate the parameters $P(z_k|u_i)$ and $P(h_j|z_k)$ by maximizing the log-likelihood function:

$$L = \sum_{i=1}^{N} \sum_{j=1}^{M} n(u_i, h_j) \log \sum_{k=1}^{K} P(h_j \mid z_k) P(z_k \mid u_i) \quad (11)$$

Then we used EM algorithm to estimate the parameters. In order to incorporate the user similarity, we renew the probability $P(z_k|u_i)$ by equation (9) at each end run of M-step, thus resulting in a variation of EM algorithm through the following E-step and M-step:

The E-step is given by

$$P(z_k \mid u_i, h_j) = \frac{P(h_j \mid z_k)\overline{P(z_k \mid u_i)}}{\sum_{l=1}^{K} P(h_j \mid z_l)\overline{P(z_l \mid u_i)}} \quad (12)$$

and the M-step is given by

$$P(h_j \mid z_k) = \frac{\sum_{i=1}^{N} n(u_i, h_j) P(z_k \mid u_i, h_j)}{\sum_{i=1}^{N} \sum_{j=1}^{M} n(u_i, h_j) P(z_k \mid u_i, h_j)} \quad (13)$$

$$P(z_k \mid u_i) = \frac{\sum_{j=1}^{N} n(u_i, h_j) P(z_k \mid u_i, h_j)}{\sum_{j=1}^{M} n(u_i, h_j)} \quad (14)$$

$$\overline{P(z_k \mid u_i)} = \sum_{l=1}^{N} P(z_k \mid u_l) P(u_l \mid u_i) \quad (15)$$

Iteratively perform E-step and M-step until the probability values are stable.

D. Prediction Algorithm

The process of building the PLSA-based model is called training. Once the training is finished, we can make use of the model to give predictions of future visits. Specifically, for user u_i, we will assume that he is interested in the topic z_k, if $p(z_k|u_i)$ is the biggest one among the sets $\{P(z_1|u_i), \ldots P(z_K|u_i)\}$. Likewise, for topic z_k, we only keep N largest value among sets. Then we assume that the N html pages corresponding to the N values are likely to be accessed under the topic z_k.

In the previous section, we introduced the intelligent web caching algorithm [6] (NGRAM-GDSF) which aims to improve the GDSF caching algorithm. NGRAM-GDSF is one of the best intelligent caching replacement algorithms. Our PLSA-based predictive caching algorithm (PN-GRAM) is an extension and enhancement of NGRAM-GDSF by incorporating a factor of predictive interest frequency. When we use the improved PLSA model, the corresponding PLSA-based predictive caching algorithm is called IPN-GRAM.

Normally, there simultaneously exist a number of active users who are accessing a web server. Based on the pre-trained improved PLSA model, our prediction model can predict each active user's interested topic and the HTML file corresponding to the topic. Different users will give different prediction to future HTML files. Since our prediction of a HTML file comes with a probability $P(h_j|z_k)$, we can combine all the current active users' interest predictions to calculate the future interest-based occurrence frequency of a HTML file. Let h_j denote a HTML file on the server, u_i be an active user on a web server, $P_{i,j}$ be the probability predicted by an active user u_i, who are interested in topic z_k, for HTML h_j. If $P_{i,j} = 0$, it indicates that HTML file h_j is not predicted by u_i. Let I_j be the future interest based frequency of requests to HTML h_j. If we assume all the users on a web server are independent to each other, we can obtain the following equation:

$$I_j = \sum_i P_{i,j} \quad (16)$$

To illustrate (16), we map two users in Figure 2. Each user yields a set of predictions to HTML files according to his interest. Since users are assumed independent to each other, we use (16) to compute the interest-based frequency of HTML file h_j. For example, The HTML file h_1 is predicted by three active users with a probability of 0.7, 0.6 and 0.5, respectively. From (16), $I_1 = 1.3$. This means that based on users' interest, the HTML file h_j will be accessed 1.3 times in the near future.

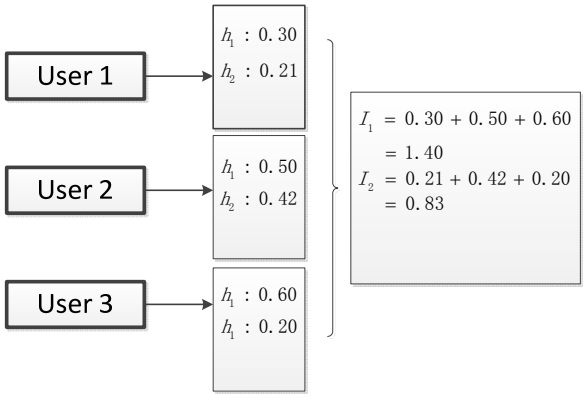

Figure 2. Prediction of interest-based frequency.

Once the future interest-based frequency $I(h)$ of a HTML h can be predicted, we extend NGRAM-GDSF [6] to incorporate the $I(h)$:

$$K(h) = L + (w(h) + F(h) + I(h)) * C(h) / S(h) \quad (17)$$

We add $w(h)$, $F(h)$ and $I(h)$ together in (17), which implies that the key value of a HTML h is determined not only by its past occurrence and session-based future frequency, but also affected by its interest. The more likely the active users are interested in it, the greater the key value will be. The rationale behind our extension is that we take into account the users interest obtained by training the web logs and adjust the replacement policy.

VI. EXPERIMENTS

A. Simulation Model

We have conducted a series of experimental comparisons with the web log that we are able to get. In the experiments, the EPA data set contains a full day of HTTP requests to the EPA web server which located at Research Triangle Park, NC. Before the experiments, we removed the records of uncacheable URLs from the web logs. A URL is considered uncacheable when it contains dynamically generated content such as CGI scripts. We also filtered out the records with unsuccessful HTTP response code. In our experiments, we use two objective performance parameters to evaluate the performance of our extended caching algorithm. The hit rate is the percentage of all requests that can be replied directly by searching the cache for a copy of the requested document. The byte hit rate represents the percentage of bytes that are transferred directly from the cache rather than from the origin server. Let N be the total number of request and $\delta_i = 1$ if the request i is hit in the cache, while $\delta_i = 0$ ohterwise. Mathematically, the two parameters can be calculated as follows:

$$HR = \frac{\sum_{i=1}^{N} \delta_i}{N} \quad (18)$$

$$BHR = \frac{\sum_{i=1}^{N} b_i \delta_i}{\sum_{i=1}^{N} b_i} \quad (19)$$

where b_i is the size of the i-th request.

B. Experimental Results

In the experiments, the algorithms under comparison are NGRAM-GDSF, PN-GRAM and IPN-GRAM. With respect to any one of the three algorithms, from Fig.3 and Fig.4, we can see that both the hit rate and byte hit rate are growing in a log-like fashion as a function of the cache size. This suggests that hit rate or byte hit rate does not increase as much as the cache size does, especially when cache size is large enough. In other words, as the cache size becomes infinite, the performance difference between these three algorithms will disappear. So we should compare the performance of caching algorithms under limited cache size.

We observed the cache hit rates and cache byte hit rates under different cache sizes. As shows in Fig.3 and Fig.4, IPN-GRAM and PN-NGRAM both perform better than NGRAM-GDSF. This result reveals that the interest-based frequency, predicted by the PLSA-based prediction model, is workable. Due to the validity of interest-based frequency, IPN-GRAM and PN-GRAM both build a more accurately prediction model than N-GRAM, so that the according cache replacement policies get a better performance. Besides, we found that IPN-GRAM is a little better than PN-GRAM, since it gets a more accurately latent topic when building the prediction model. In addition, we can see that when cache sizes are large few replacement decisions are needed and cache pollution is not a factor so the policies have similar performance. When cache sizes are very small, adding a single large document can result in the removal of a large number of smaller documents reducing the effects of cache pollution. At this case, a good caching algorithm could significantly improve the caching performance.

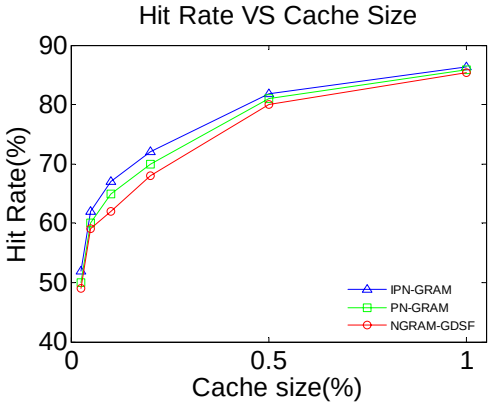

Figure 3. Hit rate comparison on EPA data

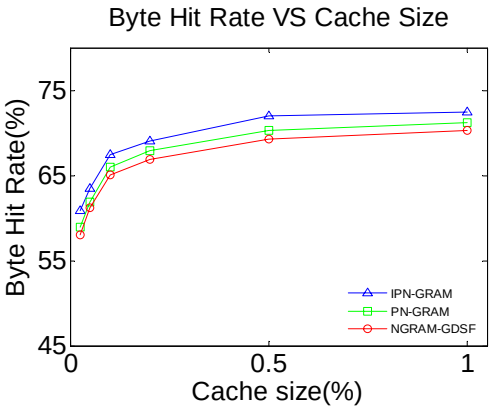

Figure 4. Byte hit rate comparison on EPA data

In order to prove the validity of the improved PLSA model, we compare the convergence rate between IPN-GRAM and PN-GRAM. PN-GRAM adopts the PLSA model and IPN-GRAM uses the improved PLSA model instead. As shows in Fig.5, the log-likelihood of IPN-GRAM convergence faster than PN-GRAM. Since IPN-

GRAM introduce similarity layer in PLSA model, topic distributions of the given user can be updated by that of users similar to him during the iterative process of training. By this way, IPN-GRAM could estimate more accurate probability model of the topic distributions in an iteration, which will accelerate the convergence rate of iterations.

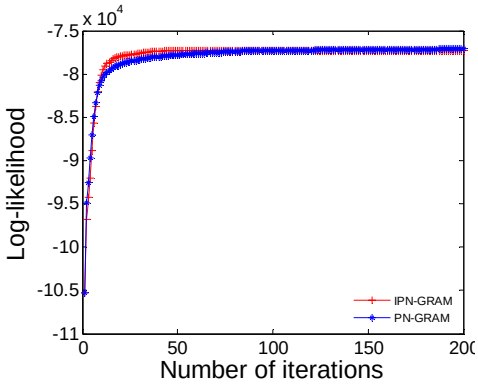

Figure 5. Log-likelihood convergence rate

VII. CONCLUSION

In this paper, we applied the PLSA-based prediction model build from the web logs to improve the NGRAM-GDSF caching algorithm. By taking into account the interest-based frequency in caching algorithm, it is possible to dramatically improve both the hit rate and byte hit rate. In the future, we would like to study on how to reduce the building time of PLSA model, so that we can dynamically update the model parameters on line. In addition, in future we can consider incorporating the clustering approach to process web logs, so that a more accurate user interest model could be obtained by building PLSA model.

ACKNOWLEDGMENT

This work and related experiment environment is supported by the National High Technology Research and Development Program of China under Grant No. 2011AA01A102, the National Key Technology R&D Program under Grant No. 2012BAH18B04 and the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDA06010302. We are sincerely grateful to their support.

REFERENCES

[1] L. Cherkasova, "improving www proxies performance with greedy-dual-size-frequency caching policy," In HP Technical Report, Palo Alto, November 1998.
[2] Q. Liu, "Web Latency Reduction with prefetching," PhD thesis, University of Western Ontario, London, 2009.
[3] H.T. Chen, "Pre-fetching and Re-fetching in Web caching system: Algorithms and Simulation," Master Thesis, TRENT UNIVERSITY, Peterborough, Ontario, Canada, 2008.
[4] W. Ali, 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, Lecture Notes in computer Science, Springer-Verlag Berlin Heidelberg, pp. 70-79, 2009.
[5] Pitkow J, Pirolli P, "Mniing longest repeating subsequences to predict www surfing," In Proceedings of the 1999 USENIX Annual Technical Conference, 1999.
[6] Qiang Yang, Haining Henry Zhang, Tianyi LI, "Mining web logs for prediction models in WWW caching and prefetching," In Proceedings of International Conference on Knowledge Discovery and Data Mining KDD'01, San Francisco, California, USA, 2001.
[7] R. Agrawal and R. Srikant, "Mining Sequential Patterns," In Proceedings of International Conference on Data Engineering, Taipei, Taiwan, 1995.
[8] Z. Su, Q. Yang, Y. Lu, H. Zhang, "Whatnext: A prediction system for web requests using n-gram sequence models," In Proceedings of the First International Conference on Web Information System and Engineering Conference, pp. 200-207, Hong Kong, June 2000.
[9] T. Hofmann, "Probabilistic latent semantic indexing," In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pp. 50-57, ACM, 1999.
[10] S. Deerwester, "Maximum likelihhod from in complete data via the EM algorithm," Journal of the Royal Statistical Society B 39, pp. 1-38
[11] 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), pp.473-476, 2008.
[12] T.M. Kroeger, D.D.E. Long, J.C. Mogul, "Exploring the bounds of web latency reduction from caching and prefetching." Proceedings of the USENDC Symposium on Internet Technology and Systems, pp. 13-22, 1997.
[13] Tao. Tan, Hongjun. Chen, "A personalization recommendation method based on Deep web data query," Journal of Computers, v 7, n 7, p 1599-1606, 2012.
[14] H.T. Chen, "Pre-fetching and Re-fetching in Web caching systems: Algorithms and Simulation," Master Thesis, TRENT UNIVERSITY, Peterborough, Ontario, Canada, 2008.
[15] A.K.Y. Wong, "Web Cache Replacement Policies: A Pragmatic Approach," IEEE Network magazine, 20(1), pp. 28-34, 2006.
[16] J Wang, "A survey of web caching schemes for the internet," ACM SIGCOMM Computer Communication Review, 1999.
[17] Jingli. Zhou, Xuejun. Nie, Leihua. Qin, Jianfeng, Zhu, "Web clustering based on tag set similartiy," Journal of Computers, v 6, n 1, p 59-66, 2011.
[18] Yanjuan. Li, Maozu. Guo, "Web page classification using relational learning algorithm and unlabeled data," Journal of Computers, v 6, n 3, p 474-479, 2011.
[19] Wei. Huang, Liyi. Zhang, Jidong. Zhang, Mingzhu Zhu, "Semantic focused crawling for retrieving E-commerce information," Journal of Software, v 4, n 5, p 436-443, July 2009.
[20] Xiangfeng. Luo, Kai. Yan, Xue. Chen, "Automatic discovery of semantic relations based on association rule," Journal of Software, v 3, n 8, p 11-18, November 2008.
[21] P. Cao, S. Irani, "Cost-Aware WWW Proxy Caching Algorithms," USENIX Symposium on Internet Technologies and Systems, Monterey, CA, 1997.