Web Access Latency Reduction Using CRF-Based Predictive Caching

Yong Zhen Guo, Kotagiri Ramamohanarao, and Laurence A.F. Park

Department of Computer Science and Software Engineering
University of Melbourne, Australia
{yzguo,rao,lapark}@csse.unimelb.edu.au

Abstract. Reducing the Web access latency perceived by a Web user has become a problem of interest. Web prefetching and caching are two effective techniques that can be used together to reduce the access latency problem on the Internet. Because the success of Web prefetching mainly relies on the prediction accuracy of prediction methods, in this paper we employ a powerful sequential learning model, Conditional Random Fields (CRFs), to improve the Web page prediction accuracy for Web prefetching. We also propose a predictive caching scheme by incorporating CRF-based Web prefetching and caching together to reduce the perceived waiting time of Web users further. We show in our experiments that by using CRF-based Web predictive caching, we can achieve higher cache hit ratio and thus reduce more access latency with less extra transmission cost when compared with the predictive caching methods based on the well known Markov Chain models.

Keywords: Web Page Prediction, Conditional Random Fields, Web Predictive Caching.

1 Introduction

The World Wide Web is a vast information source in which many people turn to for daily news and general knowledge. The easy and convenient access to information in remote locations has attracted more and more people to use the Internet, which also results in a rapid increase in the Network traffic. The popularization and convenience of wireless connections has turned a large amount of Web users to use handheld devices (such as mobile phones or PDAs) to surf the Internet, although the transmission speed of these wireless connections are usually slow. These low bandwidth Web users may spend lengthy times waiting for requested Web pages to be transferred to them from Internet, which leads to intolerable access delays.

The perceived access latency of Web users comes from several aspects. First, Web servers need to process user requests. When a Web server is overloaded, the latency caused by its processing time is noticeable. Second, Web clients need to spend time on parsing the data received (*i.e.*, interpreting a segment of script or running a Java program) and displaying the contents to Web users. Third,

W. Liu et al. (Eds.): WISM 2009, LNCS 5854, pp. 31–44, 2009.

the transmission of a file from a Web server to a user will consume a period of time. Due to the rapid development of computer hardware, the processing power of both Web servers and clients have improved dramatically, therefore, the access latency caused by the processing time of Web servers and clients is negligible. Because the network bandwidth (especially the wireless network) is limited, the transmission of a large amount of data across a long distance over narrow bandwidth will take a relatively long time, thus the transmission time from Web servers to users results in most of the perceived latency of Web users.

Researchers have proposed many techniques to decrease the Internet access latency, among which *caching* and *prefetching* are two main techniques. In Web prefetching if most prefetched Web pages are not visited by a Web user in his subsequent accesses (implying that the prefetching method has predicted the user's actions poorly), the limited network bandwidth and server resources will not be used efficiently, increasing the access delay to the user-requested pages and worsening the access latency problem. Consequently, the success of Web prefetching relies mainly on the Web page prediction accuracy. In this paper, we introduce the use of a powerful prediction algorithm, Conditional Random Fields (CRFs) [1], to improve the prediction accuracy of Web prefetching. Furthermore, although a prefetched Web page is not the page a user wants to visit currently, it might be requested by the user in his subsequent access, we can save these prefetched pages in the user's cache to reduce the waiting time further. In this way caching and prefetching can be combined together as predictive caching to improve the cache hit ratio and provide better performance in reducing Web access latency. In this paper we propose the CRF-based predictive caching of this kind for Web access latency reduction by combining CRF-based prefetching and caching together. We show in our experiments the merits of CRF-based Web predictive caching in reducing Web access latency over the one based on the popular Markov Chain models [2].

The rest of this paper is organized as follows: In Section 2 we briefly describe the Web page prediction method based on Conditional Random Fields. In Section 3 we illustrate how we combine CRF-based prefetching and caching together as predictive caching for Web access latency reduction. In Section 4 we compare the performance of CRF and Markov Chain-based predictive caching in improving cache hit ratio and reducing transmission cost. In this section we also show the impact of CRF-based predictive caching in reducing access latency by simulations. Finally, we conclude this paper in Section 5.

2 Conditional Random Field-Based Web Page Prediction

The prediction algorithm employed to estimate the probability of each Web page being requested by a Web user in the immediate future is very important for Web prefetching. In this section we will first briefly review the basic principle of Conditional Random Fields (CRFs) and then explain how to use the CRF-based Web page prediction in Web prefetching.

2.1 Conditional Random Fields

A Conditional Random Field [1] is an undirected graphical model, it defines a conditional probability distribution of a label sequence $Y = (y_1, y_2, \cdots, y_n)$, given an observation sequence $X = (x_1, x_2, \cdots, x_n)$. Although theoretically the structure of a Conditional Random Field can be an arbitrary undirected graph that obeys the Markov property, for the tasks of labeling the most common graphical structure is an undirected linear chain of first-order among label sequence Y. A linear chain CRF has the form as below:

$$P_\theta(Y|X) = \frac{1}{Z(X)} \exp\left[\sum_{t=1}^{T}\left(\sum_i \lambda_i f_i(y_{t-1}, y_t, X, t) + \sum_j \mu_j s_j(y_t, X, t)\right)\right] \tag{1}$$

Where $f_i\,(y_{t-1}, y_t, X, t)$ is a transition feature function between the states (labels) at position $t-1$ and t, while $s_j\,(y_t, X, t)$ is a state feature function of the state at position t. $Z(X)$ is a global normalization factor over all possible label sequences. The parameters $\theta = (\lambda_i, \mu_j)$ can be estimated by maximizing the log-likelihood of the training data by using approach such as L-BFGS [5]. In terms of the size of training data, the CRF training using L-BFGS usually requires many iterations, each of which calculates the log-likelihood and its derivative [3]. The larger number of sequences a dataset has, the more iterations it needs for the L-BFGS algorithm to converge. The time complexity of such an iteration is $O(L^2NT)$, where L is the number of labels, N is the number of sequences (*e.g.*, Web page sessions) and T is the average length of sequences. Therefore, the time complexity of a CRF training is quadratic with respect to the number of unique labels. When there are a large number of unique labels, the CRF training can become very slow or intractable even with efficient training algorithm like L-BFGS. However, in the next section we will describe how to reduce the training complexity of CRFs. After the parameters are trained, the Viterbi [4] algorithm can be used to label the testing data and perform the prediction.

Because CRFs directly model the conditional distribution $P(Y|X)$, they do not need to model the visible observation sequences X, which results in the relaxation of unwarranted independence assumptions over observation sequences. Actually, CRFs can model long-range dependencies between observation elements. Moreover, due to the conditional nature, CRFs are able to model arbitrary features of observation sequences, regardless of the relationships between them. Because of these advantages, CRFs can yield more accurate prediction than other popular prediction methods such as Markov Chain models.

2.2 Grouped ECOC-CRFs for Web Page Prediction

In the Web page prefetching scenario, each unique Web page of a Website is regarded as a unique label. Then we can treat each previous user session of the Website as an observation sequence, and in a user session we can consider each

pageview's subsequent pageview as its label. In this way, we can make use of all the previous uesr sessions to obtain observation sequences and the corresponding label sequences which can be used to train a CRF model for making prediction on this Website. It has been shown in [8] that CRFs can be used efficiently in Web page prediction on Websites whose number of unique Web pages is small.

In a Website where there are a large number of unique Web pages, the direct use of CRF-based prefetching is infeasible due to the inherent high complexity of CRF training (quadratic to the number of unique labels). In this case, we can reduce the overall training complexity of CRFs for Web prefetching by using Error Correcting Output Coding (ECOC) [6] method to decompose a multi-label CRF training into a set of binary-label sub-CRF trainings, and then combine the output of all the sub-CRFs to obtain the possible original result [9]. Since all the sub-CRFs are binary, they can be trained very fast and efficiently.

Moreover, although by using ECOC method we can achieve faster training time for CRF-based Web prefetching, we also drastically reduced the class information given to each sub-CRF, which will decrease the overall prediction accuracy slightly when compared to a multi-label CRF training. We can improve the prediction accuracy of ECOC-CRFs by using the grouping technique [10], in which the columns of the ECOC code matrix are divided into several groups and each group is used to train a sub-CRF. By doing this, each sub-CRF can obtain more refined class information and the prediction accuracy can be enhanced further. In this way, we can utilize grouped ECOC-CRFs to obtain satisfactory prediction accuracy for Web page prefetching on large-size websites.

3 CRF-Based Predictive Caching

Caching techniques save the copies of the Web pages that a user has visited in his local cache, in the future if this user wants to visit these pages again his browser can display these pages immediately from the cache. Every time when a user requests a Web page, if this page is in his cache, then there is a cache hit; otherwise there is a cache miss, and this page will be put into the user's cache. The cache hit ratio is then defined as *cache hits / (cache hits + cache misses)*. The more requested Web pages can be found in cache, the higher the cache hit ratio is, and consequently more perceived access waiting time can be saved. However, practically in a surfing process a Web user usually tends to jump from the current Web page to a new Web page that was not accessed before, seldom going back to the Web pages he has previously visited. In this case, most cached Web pages will not be used again during this user's surfing on the Internet. Therefore, using caching solely might not easily achieve a satisfactory performance in decreasing Web access latency.

Not like caching, prefetching techniques usually prefetch Web pages to a user before this user makes request to them. With a high-accuracy prefetching, most Web pages requesed by the Web user can be downloaded in advance. Furthermore, although some of these prefetched Web pages might be currently unwanted, they may be requested by the user in his subsequent access, and these

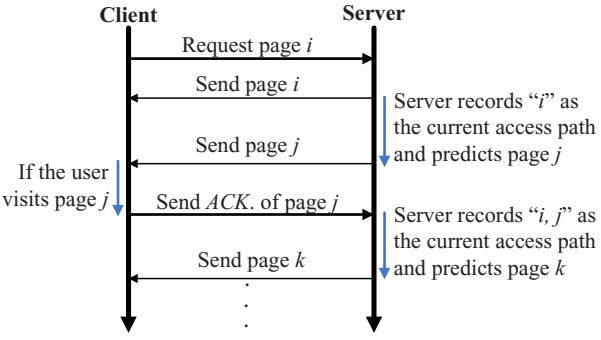

Fig. 1. Data exchange between client side and server side in a practical predictive caching system. ACK is the acknowledgement package.

prematurely prefetched Web pages can be stored in the user's cache to enhance the cache hit ratio. By doing so, caching and prefetching can be combined together as predictive caching to reduce more Web access latency, where cache is more like a buffer for prefetched Web pages. In this paper, we incorporate CRF-based prefetching and caching togegher in this way as CRF-based predictive caching to reduce the access latency of Web users.

Moreover, because Web page prediction needs to utilize the historical access information of previous users stored on Web servers, prefetching is usually made by the server side. In order to predict a possible Web page correctly for a Web user, the Web server needs to keep track of this user's current access path. However, in a predictive caching system, if the requested Web page can be found in the client's cache, the client will not make a request to the Web server for this Web page, in which case the server can not be aware of the user's current behaviour and thus can not make a correct prediction for the user. Therefore, in practice it is necessary for the client to send an acknowledgement package to the server informing the server about the user's access to a cached Web page. This allows the server to maintain this user's current access path exactly. Since the acknowledgement package only contains the ID information of a Web page, its size is very small and the caused network traffic of it can nearly be neglected. An example process of this is illustrated in Figure 1.

4 Experimental Evaluations

In this section we conducted experiments to evaluate the effect of predictive caching based on CRFs and Markov Chain models in reducing the Web access latency. In our experiments we implemented the first, second and third order Markov Chains (referred as 1^{st}-MC, 2^{nd}-MC and 3^{rd}-MC respectively in the future) for Web page prefetching. For the implementation of CRF training and decoding, we use the CRF++ toolkit [13]. We use three different feature templates of CRF++ to generate the feature functions that will be used in the

CRF training in our experiments. In the first template (referred to as CRF1), we define the current and previous one observation and their combination as the unigram features; the second and the third template (CRF2 and CRF3) are defined similarly.

Since the cache hit ratio will directly influence the perceived waiting time that can be saved, we first examine how the predictive caching based on CRFs and Markov Chains can improve the cache hit ratio on two datasets: the publicly accessible *msnbc* dataset [11] and the larger *CSSE* dataset [12].

4.1 Experiments on the *msnbc* Dataset

The first experiment about cache hit ratio is carried out on the *msnbc* dataset [11], which is obtained from the Web logs of *www.msnbc.com*. In this dataset, all the user visits are recorded in session format at the level of Web page categories such as *health*, *sports* and so on. There are 17 different page categories of this kind which can also be treated as 17 unique Web pages. After preprocessing we randomly selected 35,000 distinct sessions with length more than 8 and less than 100 from this dataset and divided them into two subsets: 30,000 sessions for training and 5,000 for testing.

In this experiment the training dataset is used to train the CRF and Markov Chain prediction models and the calculation of the average cache hit ratio is based on the testing dataset. All the user sessions in the testing dataset are assumed to be from different users, each of whom has an empty cache or buffer of a fixed size n (for simplicity we assume all Web pages are of the same size 1). Then for every pageview of each user session in the testing dataset a procedure is conducted: First, if this pageview is found existing in its user's cache, then there is a cache hit; otherwise, a cache miss occurs and this pageview is put into its user's cache. Second, the prediction models obtained from the training stage are employed to predict the most possible next Web page for this pageview and prefetch this page to its user's cache. Because prefetching only happens after a Web user requests the first Web page, there is always a miss for the first Web page of every user session. Every time when the user's cache is full, the "LRU" cache replacement strategy is adopted to remove the least recently used Web page from the cache. This procedure continues from the first pageview to the last pageview of a session, then the cache hit ratio for this session (or user) can be calculated. In this way, we can obtain the cache hit ratio of all sessions (or users) in the testing dataset and calculate the average value of them.

The average cache hit ratio of the predictive caching based on CRFs and Markov Chains at different cache sizes on the *msnbc* dataset are shown in Figure 2, in which the result of the pure caching method without prefetching (referred as LRU) is also shown as a baseline. From this figure we can see that while the cache size increases, the cache hit ratios of all the methods with or without prefetching increase correspondingly. The reason of this is straightforward: with a bigger cache size more Web pages can be cached, thus more Web pages a user requests can be found in the cache. When compare the cache hit ratios of these different methods at a same cache size, we can find that all prefetching methods perform much better than

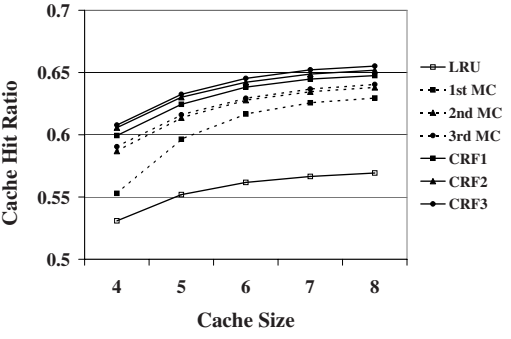

Fig. 2. Cache hit ratios at different cache sizes on the *msnbc* dataset

LRU (the method without prefetching), this is because at every visit the prefetching method will bring in the possible "next" pages that might be requested by the user immediately, which in turn improve the cache hit ratio. Furthermore, it is noticeable that CRF-based predictive caching can outperform Markov Chain-based predictive caching significantly, this is because CRFs are more accurate models than Markov Chains in predicting Web user behaviours.

4.2 Experiments on the *CSSE* Dataset

The same experiment is carried out on the *CSSE* dataset [12] from the Web log of the Department of Computers Science and Software Engineering (*CSSE*), the University of Melbourne, which contains 3,829 unique Web pages. After preprocessing, we randomly select 2,723 user sessions as the training dataset and 544 user sessions as the testing dataset.

Because there are many labels (3,829 unique pages) in this dataset, the training of a multi-label CRF here is infeasible due to the high complexity of CRF training. Therefore, we use grouped ECOC-CRFs for the CRF training in this dataset. We made use of the Search Coding [7] method to design a $3,829 \times 24$ ECOC code matrix whose minimum Hamming distance d is 8, and then grouped every 8 columns of this code matrix into one group, which means we have 3 groups and need to train 3 sub-CRFs.

In this experiment we also show the cache hit ratio of an *Optimal* scheme as a comparison. The *Optimal* scheme uses an ideal prefetching method whose prediction accuracy is 100% correct to every page except the first one of each user session (we assume prefetching only happens after a Web user requests the first Web page, and the first Web page of a user session is always unpredictable). Moreover, the *Optimal* scheme uses an optimal cache replacement strategy to replace Web pages. The optimal cache replacement is a theoretical page replacement algorithm, which works as follows: when a page needs to be swapped in, the cache system discards the page whose next use will occur farthest in the future. For example, a page that is not going to be used for the next 10 visits will be discarded over a page that will be used within the next 3 visits. Although the

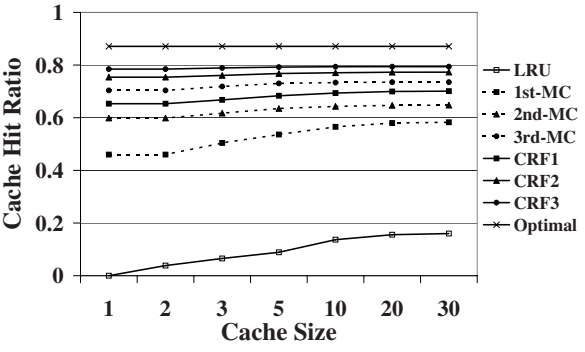

Fig. 3. Cache hit ratios at different cache sizes on the *CSSE* dataset

optimal cache replacement strategy is the best cache replacement strategy, it is not realizable in practice since it is impossible to predict how far in the future a page will be needed. However, because in our experiment from each user session we can know exactly what page sequence the user will request subsequently at each visit in advance, we can simulate the optimal replacement strategy. The *Optimal* scheme can yield a higher cache hit ratio than any other methods. The closer the cache hit ratio a predictive caching method with "LRU" cache replacement strategy can obtain to that of the *Optimal* scheme, the better the method.

The cache hit ratios of different methods at different cache sizes on the *CSSE* dataset can be found in Figure 3. In this figure we can see that the *Optimal* scheme has the highest cache hit ratio, while the other predictive caching methods with relatively higher prediction accuracy (such as CRF2 and CRF3) can obtain a closer cache hit ratio to it, which indicates these methods can reduce waiting time more for Web users. We can also notice that *LRU* (the method without prefetching) yields the lowest cache hit ratio (lower than 20% when at the cache size of 30), which shows the effect of prefetching in improving the cache hit ratio. Moreover, from this figure we can see that the cache hit ratios of all methods except *Optimal* increase while cache size enlarges, which again indicates that increasing cache size can help to increase cache hit ratio. The reason that the cache hit ratio of *Optimal* remains constant while cache size enlarges is due to its 100% correct prediction accuracy: for a user every requested Web page (except the first Web page) has already been prefetched into his cache, no matter what the cache size is. Finally, it is obvious in this plot that the increase rate of the cache hit ratio of the predictive caching methods with low prediction accuracy (such as 1^{st}-MC and 2^{nd}-MC) is higher than the predictive caching methods with high prediction accuracy (such as CRF2 and CRF3). Therefore, caching is more important when the predictive caching method's prediction accuracy is low.

In Figure 3 we showed the effect of predictive caching on increasing the cache hit ratio of Web pages requested by Web users, in Figure 4 we will examine the impact of predictive caching on the hit ratio of predicted Web pages (referred as

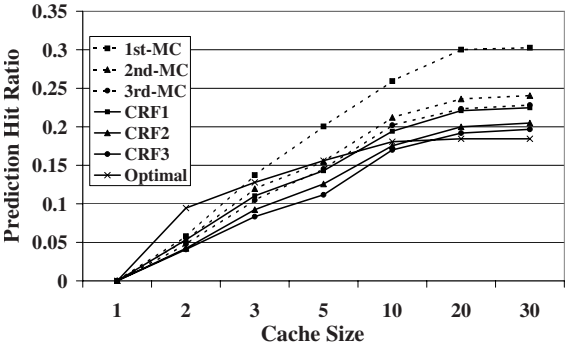

Fig. 4. Prediction hit ratios at different cache sizes on the *CSSE* dataset

prediction hit ratio, if the predicted Web page can be found in the cache, then there is a hit). The higher the prediction hit ratio is, more predicted Web pages do not need to be transmitted and thus can save more bandwidth. In Figure 4 we can see that while the cache size increases the prediction hit ratio of all predictive caching methods increase significantly. It is noticeable that the general trend in this plot shows that the prediction hit ratio of low-accuracy predictive caching methods (such as 1^{st}-MC) is higher than that of high-accuracy predictive caching methods (such as CRF3). This is because generally a Web user will jump from a Web page to a new page he does not visit before, and the low-accuracy predictive caching methods have less chance to predict the user's real intention correctly and tend to predict more randomly than the high-accuracy predictive caching methods.

We also evaluated the transmission cost of each method in Figure 5. Every time when the Web server transmits a page (whether it is requested by the user explicitly or prefetched by the prefetching system) to a user, a transmission cost will occur. Transmission cost can be used to indicate the extra network traffic incurred by prefetching. The lower a predictive caching method's caused transmission cost, the better this method. From Figure 5 we can see that the transmission cost of all methods decrease as the cache size increases due to more Web pages can be saved with a larger cache. The transmission cost of predictive caching methods (except *Optimal*) are much higher than that of the method without prefetching (*LRU*), the reason is that every time these predictive caching methods need to prefetch extra pages into the cache, and if the exact page the user accesses is not in the cache after prefetching, the user's browser still needs to connect to the Web server to download this page. Because of the 100% prediction accuracy and the optimal cache replacement strategy, the transmission cost of *Optimal* is lower than *LRU*. By comparing the predictive caching methods, we can notice that the methods with higher prediction accuracy can produce lower transmission cost, this is because higher prediction accuracy results in higher cache hit ratio and thus more Web pages a user requests can be found in his cache. Among all the predictive caching methods, CRF3 produces the closest transmission cost to the *Optimal* scheme, denoting the merits of CRF-based predictive caching.

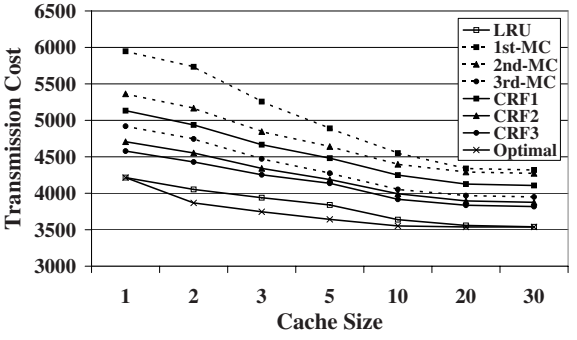

Fig. 5. Transmission cost at different cache sizes on the *CSSE* dataset

4.3 Simulations on Access Latency Reduced by CRF-Based Predictive Caching

In order to evaluate how well CRF-based predictive caching can reduce a Web user's perceived access latency, we run two more simulations regarding the average waiting time Web users spent on each Web page on the *CSSE* dataset.

In these simulations, each Web page of all user sessions in the testing dataset will be assigned a random transmission time (the time needed to transmit this Web page from the Web server to the user) and a random reading time (the time the user spent on reading this page). The calculation of the waiting time on every Web page of a session can be described by an example as follows: If a user session in the testing dataset is *"A B"*, which means the user's exact browsing path is from Web page A to page B, then the user's waiting time on page A is page A's transmission time because page A is the first page the user requests, his browser needs to download page A from the very beginning. Then there are two cases that need to be considered: (1) Page B is prefetched correctly as the next requested page by a prefetching method, and (2) the prefetching method fails to predict the user's behaviour correctly and prefetchs pages other than B for the user. In the first case, if page B's transmission time is less than the reading time the user spent on page A, then there is zero waiting time for page B since while the user is reading page A, page B has already been prefetched to his computer; Otherwise, if page B's transmission time is longer than page A's reading time, then the user's perceived waiting time on page B is calculated as page B's transmission time minus page A's reading time. In the second case, because page B is not correctly prefetched while the user is reading page A, his browser needs to download page B when he requests it, thus the user's perceived waiting time on page B is page B's transmission time. The calculation of the waiting time on each Web page of this example is shown in Figure 6. In addition, the influence of transmitting a prefetched Web page on the transmission time of the Web pages that a user explicitly requests is also considered in these simulations. For example, if a requested Web page and a prefetched Web page are transmitting simultaneously, the transmission time of the requested Web page will double because the transmission of the prefetched

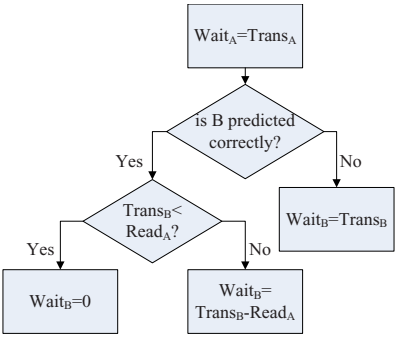

Fig. 6. Calculation of the waiting time on each Web page of the example session *"A B"*. $Wait_A$, $Trans_A$ and $Read_A$ mean the waiting time, transmission time, and reading time of Web page A respectively.

Web page takes up half of the user's bandwidth (assuming the user's bandwidth is constant). After the waiting time on every Web page is calculated, we sum them and divide this value by the total number of Web pages in the testing dataset to get the average waiting time on each Web page.

We generated a random transmission time that follows the Poisson distribution and a random reading time that obeys the Exponential distribution for each Web page of every session in the testing dataset. In the first simulation, we set the *mean transmission time* of all Web pages to 20 seconds and the *mean reading time* of all Web pages to 60 seconds to represent the case where the mean transmission time of Web pages is shorter than the mean reading time. In the second simulation, we set the *mean transmission time* to 60 seconds and the *mean reading time* to 20 seconds to represent the case where the mean transmission time of Web pages is longer than the mean reading time. Then we calculated the average waiting time of different predictive caching methods based on CRFs and Markov Chains in these two simulations, where the results of *LRU* and *Optimal* are depicted as well. We run each simulation 10 times and calculate the average value as the final result. The results of these two simulations are shown in Figure 7 and Figure 8 respectively.

From the results of these two simulations we find that with a bigger cache size, the average waiting time of all methods decrease drastically (especially in Simulation 2), thus caching is very useful in reducing a Web user's waiting time. Of all the methods, the *Optimal* scheme yields the shortest average waiting time. When compared at a certain cache size, the average waiting time of the predictive caching methods with high prediction accuracy (such as CRF3) is much shorter than the methods with low prediction accuracy (such as 1^{st}-MC), and the higher prediction accuracy a predictive caching method has, the closer the performance it can achieve to the *Optimal* scheme. Moreover, in Figure 8 we also observe that when the cache size is smaller than 5, the average waiting time of the predictive caching method based on the 1^{st}-order Markov Chain is longer than the non-prefetching method *LRU*, this is due to the low prediction accuracy

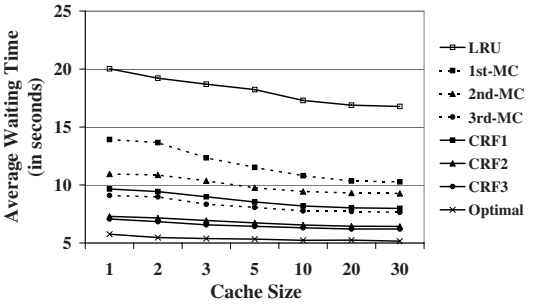

Fig. 7. Average waiting time perceived by Web users in the first simulation (*mean transmission time*=20 seconds and *mean reading time*=60 seconds) on the *CSSE* dataset

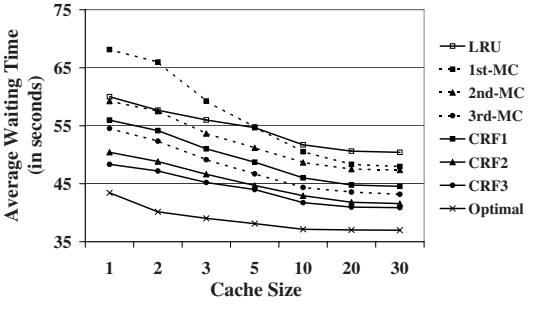

Fig. 8. Average waiting time perceived by Web users in the second simulation (*mean transmission time*=60 seconds and *mean reading time*=20 seconds) on the *CSSE* dataset

of the 1^{st}-order Markov Chain and in Simulation 2 the mean transmission time of Web pages is much longer than the mean reading time. However, with a cache size bigger than 5 the 1^{st}-MC can perform better than *LRU*.

Finally, in Figure 9 we depict the results concerning the percentage of perceived access latency reduction that can be achieved in these two simulations and the corresponding extra transmission cost for each predictive caching method when compared with *LRU* at the cache size of 30. From Figure 9 we can see that the percentage of reduced waiting time is proportional to the prediction accuracy of predictive caching methods, while the extra transmission cost presents an opposite trend. Predictive caching methods with high accuracy prediction can reduce substantial waiting time while only incur slight extra transmission cost. For example, CRF3 has the highest prediction accuracy and thus has the best performance here: by using CRF3 the percentage of perceived waiting time reduction can achieve 63.0% and 18.9% in Simulation 1 and Simulation 2 respectively, while the extra transmission cost is only about 7.3%, which is less than all other Markov Chain-based predictive caching methods. Therefore, CRF-based predictive caching can reduce Web access latency more efficiently than the one based on Markov Chains.

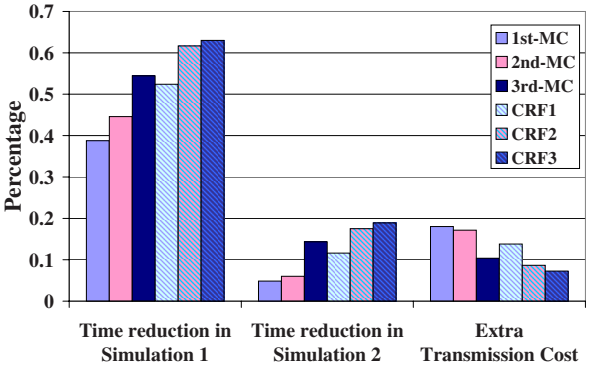

Fig. 9. Reduction of perceived waiting time VS. Extra transmission cost when compared with *LRU* at the cache size of 30 on the *CSSE* dataset

5 Conclusion

Low-bandwidth Web users usually suffer a lot from waiting for the Web pages they request being tranferred to them. In this paper, we propose the use of Web predictive caching for Web access latency reduction by combining Web prefetching and caching together. Since the performance of predictive caching relies crucially on the Web page prediction algorithm used in prefetching, in this paper we utilize a powerful prediction algorithm based on Conditional Random Fields in prefetching to save more waiting time for Web users. We examine the effect of predictive caching in improving the cache hit ratio and decreasing the transmisstion cost, and show the advantages of CRF-based predictive caching over the one based on Markov Chains in reducing a Web user's access latency. Our simulation results show that by using CRF-based Web predictive caching (CRF3), the average access latency of Web users can be dramatically decreased by up to 63% with a slight extra transmission cost of 7.3% when compared with *LRU*.

References

1. Lafferty, J., McCallum, A., Pereir, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: Proceedings of the 18th International Conference on Machine Learning, pp. 282–289 (2001)
2. Markov, A.: Extension of the Limit Theorems of Probability Theory to a Sum of Variables Connected in a Chain. In: Dynamic Probabilistic Systems, vol. 1, Markov Chains (1971)
3. Wallach, H.: Efficient Training of Conditional Random Fields. Master thesis, Division of Informatics, University of Edinburgh (2002)
4. Viterbi, A.J.: Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. IEEE Trans. Info. Theory, 260–269 (1967)
5. Nocedal, J.: Updating Quasi-Newton Matrices with Limited Storage. In: Mathematics of Computation, pp. 773–782 (1980)

6. Dietterich, T., Bakiri, G.: Solving Multiclass Learning Problems via Error-Correcting Output Codes. J. Arti. Inte. Rese (1995)
7. Jiang, Y.H., Zhao, Q.L., Yang, X.J.: A Search Coding Method and Its Application in Supervised Classification. J. Software (2005) (in Chinese)
8. Guo, Y.Z., Ramamohanarao, K., Park, L.: Web Page Prediction Based on Conditional Random Fields. In: Proceedings of the 18^{th} European Conference on Artificial Intelligence (ECAI 2008), pp. 251–255 (2008)
9. Guo, Y.Z., Ramamohanarao, K., Park, L.: Error Correcting Output Coding-Based Conditional Random Fields for Web Page Prediction. In: Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2008), pp. 743–746 (2008)
10. Guo, Y.Z., Ramamohanarao, K., Park, L.: Grouped ECOC Conditional Random Fields for Prediction of Web User Behavior. In: Proceedings of the 13^{th} Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 757–763 (2009)
11. UCI KDD Archive, `http://kdd.ics.uci.edu/databases/msnbc/msnbc.html`
12. Department of Computer Science and Software Engineering, University of Melbourne, `http://csse.unimelb.edu.au`
13. CRF++, Yet another CRF toolkit, `http://crfpp.sourceforge.net`