World Wide Web (2016) 19:921–932
DOI 10.1007/s11280-015-0367-8

Web prefetching through efficient prediction by partial matching

Arpad Gellert[1] · **Adrian Florea**[1]

Received: 15 December 2014 / Revised: 10 July 2015 /
Accepted: 28 July 2015 / Published online: 9 August 2015

Abstract In this work we propose a prediction by partial matching technique to anticipate and prefetch web pages and files accessed via browsers. The goal is to reduce the delays necessary to load the web pages and files visited by the users. Since the number of visited web pages can be high, tree-based and table-based implementations can be inefficient from the representation point of view. Therefore, we present an efficient way to implement the prediction by partial matching as simple searches in the observation sequence. Thus, we can use high number of states in long web page access histories and higher order Markov chains at low complexity. The time-evaluations show that the proposed PPM implementation is significantly more efficient than previous implementations. We have enhanced the predictor with a confidence mechanism, implemented as saturating counters, which classifies dynamically web pages as predictable or unpredictable. Predictions are generated selectively only from web pages classified as predictable, improving thus the accuracy. The experiments show that the prediction by partial matching of order 4 with a history of 500 web pages is the optimal.

Keywords Web page prediction · Prefetching · Markov chains · Prediction by partial matching · Browser extension

1 Introduction

Nowadays the access to the Internet becomes more prevalent worldwide and often there are requests for higher bandwidths. Clients are frequently confronted with delays in accessing web pages, especially those ones that are limited by low-bandwidth modems or wireless routers. Many latency-tolerant techniques have been developed during the last years, the most important being caching and prefetching. Caching exploits the temporal locality principle by keeping

✉ Arpad Gellert
arpad.gellert@ulbsibiu.ro

Adrian Florea
adrian.florea@ulbsibiu.ro

[1] Computer Science and Electrical Engineering Department, Lucian Blaga University of Sibiu, Sibiu, Romania

the accessed pages and files in a cache structure, whereas prefetching anticipates the next accesses and loads the corresponding web pages or files into the cache. If the user accesses a web page or a file which is available in the cache, the browser can load it without any delays. Thus, when the users have long browsing sessions, prediction-based prefetching can be very effective by minimizing the access latencies.

In this paper we analyze web page prefetching through prediction by partial matching (PPM) enhanced with a dynamic confidence mechanism. The PPM was first introduced by Cleary and Witten in [4] for data compression and represents an important context-based prediction method. It contains a set of simple Markov predictors. It uses the highest order Markov predictor which can provide prediction. The predicted value is the value that followed the context with the highest frequency. The number of states used in such models tends to rise exponentially as the order of the model increases [6]. Therefore, we implemented the PPM algorithm as simple searches in the observation sequence instead of using high complexity tree-, graph- or table-based modelling. Our simple representation allows superior order Markov chains with high number of states and long histories, at low complexity. Thus, the proposed PPM implementation is significantly more efficient than previous implementations. We have also enhanced the predictor with a confidence mechanism, which classifies dynamically web pages as predictable or unpredictable. Predictions are generated selectively only from web pages classified as predictable, improving thus the accuracy. The confidence mechanism consists in dynamically adapted saturating counters attached to each web page. We evaluate the proposed predictor in terms of prediction accuracy on the BU benchmark set, generated by the "Ocean Group Research" from Boston University [5]. The goal is to find the most appropriate prediction technique for anticipating and prefetching web pages and to integrate it as an extension into browsers.

The organization of the rest of this paper is as follows. In Section 2, we are reviewing the related work in web page prefetching. Section 3 is describing our proposed web page predictors. In Section 4 we are presenting the experimental results. Conclusions and further work directions are included in Section 5.

2 Related work

In our previous work [11] we compared Markov chains, Hidden Markov Models and graph algorithms as web page prediction methods. We applied multi-page prediction by prefetching all the pages that appeared in the history after the considered context. The best prediction accuracy has been obtained with a hybrid predictor consisting in a HMM and a graph-based predictor. The proposed hybrid predictor prefetched the web pages anticipated by both component predictors. In contrast, in this work we use the prediction by partial matching algorithm which combines different order Markov chains exploiting thus the advantage of each one. In order to reduce the additional network traffic, in this work we prefetch only one predicted web page from each confident state.

In [14] the authors proposed a continuous-time Markov model for web page prediction. They used Kolmogorov's backward equations to compute the transition probabilities and to predict which web page will be visited by a certain user and when. Their method can also estimate the total number of visits to a web page done by all the people at a certain interval. Link prediction based on Markov chains was presented in [25]. In [17] the author applied the Markov model together with the k-nearest neighbor classifier algorithm to enhance the

performance of traditional Markov chains in predicting web pages. He obtained lower consumed memory, quite similar build time and evaluation time and higher accuracy. In [18, 19] clustering is used to group homogeneous user sessions. Low order Markov chains are built on these clustered sessions and when Markov models cannot make clear predictions the association rules are used. In [20] the authors presented a survey of web page ranking for web personalization. They concluded that low order Markov models have higher accuracy and lower coverage, whereas the higher order models have a number of limitations associated with: higher state complexity, reduced coverage and sometimes even worse prediction accuracy.

In [24] Pitkow and Pirolli show that, compared to various Markov models, longest repeating subsequence models can significantly reduce complexity while keeping the ability to make accurate predictions. In [9] the authors proposed a hybrid web page prediction method which combines support vector machines, association rule mining and Markov chains in order to enhance the efficiency. Their experimental results showed that the proposed hybrid predictor outperformed the individual predictors. In [26] the authors presented an *n*-gram based model to utilize path profiles of users from very large web log to predict future requests. They showed that the proposed method achieves a reasonable balance between precision and applicability.

In [16] the authors used HMMs for semantic-based web page prefetching. In [12] the authors presented a web page prediction method based on conditional random fields, used for labeling and segmenting sequential data. Conditional random fields have the ability to model the dependencies among observations and they can also incorporate various features from observation sequences to increase the prediction accuracy. They concluded that conditional random fields outperformed Markov chains and HMMs but for a great number of labels their training may become very expensive and even intractable.

In [6] the authors used different PPM techniques for web page prefetching. They observed that as the order of the model increases, the number of states used for the model also increases dramatically. They reduced the complexity of the model by eliminating many of its states that are expected to have low prediction accuracy. But the behaviour of a certain user can change in time and a state with low prediction accuracy can become a state with high accuracy and vice versa. In contrast, we select dynamically the confident states through the saturating counters attached to each web page. The great advantage of the saturating counters is that they can adapt fast to any changes in the user's behaviour. We reduced the complexity of PPM modelling by keeping all the states. In [31] the authors reduced the high state complexity of Markov chains by clustering web pages with similar transition behaviours together to get a compressed transition matrix. However, their compression is based on similarity metrics which accept a low level of error and thus can cause some loss of information. In [23] the PPM is modelled as a tree whose nodes are the web pages. The model is updated on new user requests. The prediction engine selects which page to prefetch based on the occurrence count associated to each node. The fraction between the occurrence counts of a certain node and its parent is used to decide to activate or not the prefetch. In contrast, in our approach we decide to prefetch or not based on the confidence of the current web page and we use saturating confidence counters for faster adaptation to user behaviour changes. The tree is inefficient in the case of high Markov chain order and high number of web pages due to the resulting high state complexity. In [1] the authors use a noncompact suffix tree to implement an online PPM which keeps the most recent requests, being able thus to capture changing patterns and fit the memory. In our approach we apply the PPM on a limited sliding window of web requests as in [1] but our model representation is different. In contrast with [6, 31, 23, 1], we implemented the PPM as simple searches for the contexts in the sequence of visited web pages and thus the complexity grows only linearly with the order of the model.

Graphs were also used for prediction through some specific algorithms. In [15] Huang explored link prediction based on graph topology. In [13] Hasan addressed link prediction in social networks by using supervised learning and classification algorithms such as decision tree, multi-layer perceptron, support vector machines. His results were reported on BIOBASE and DBLP, two datasets that have information about different research publications in biology and computer science, respectively. Another approach in link prediction of social networks was proposed in [22]. The author's solution aims at predicting links based on weighted proximity measures of social networks relying on the structural properties (topology) of a given network.

In [3] the authors proposed a page rank algorithm to predict the next page that will be visited by a web surfer. For the first set of sessions they applied the page rank algorithm which provides the ranks for web pages. For each web page their method determines to which pages the user can navigate and, using the page ranks, it computes the probability of visiting them by dividing each rank to the sum of ranks, and the number of links to the total number of links, respectively.

In [2] Canali et al. proposed adaptive algorithms that combine predictive and social information and dynamically adjust their parameters according to the continuously changing workload characteristics. Their experimental results showed that such adaptive algorithms can achieve performance close to theoretical ideal algorithms. In [28] Wan et al. proposed an approach based on a vector space model, called random indexing, for the discovery of the intrinsic characteristics of web user activities. The underlying factors are then used for clustering individual user navigational patterns. The clustering results are used to predict and prefetch web requests for grouped users.

In [27] Temgire et al. presented a review on web prefetching techniques. The variety of prefetching algorithms hinder to a certain extent the performance evaluation of new techniques and their correct comparison with existing methods. Therefore, in [7] the authors proposed a free environment for implementation and efficient evaluation of prefetching techniques. Their framework combines real and simulated behaviour of web users and servers. A proxy-based framework for prefetching web pages has been proposed in [29]. Some of the algorithms proposed in the literature consider a training period before making predictions. The training period can improve or even decrease performance, since, if it is too long, it can involve a high amount of resources. In [8] the authors analyzed how this training affects the prediction accuracy. Instead of pre-training we applied run-time training.

User behaviour prediction has been applied also in online advertisement industry [21] and financial service industry [30].

3 Web page prediction

Our goal is to integrate the most efficient PPM configuration as an extension into browsers, as in [11]. The browser extension, presented in Figure 1, collects and pre-processes the links accessed by the user: each link is codified with a unique number, ports and relative parts are eliminated from links, if there are two consecutive accesses of the same link only one is considered, links having the extension *.gif* and *.xbm*, which are usually small images, are also eliminated. The browser extension keeps a certain history of the accessed links. When the current link is accessed, the next link is anticipated using the PPM algorithm on the basis of the history of the previously visited links. The predicted web page or file is prefetched in the cache in order to be available if the user accesses it.

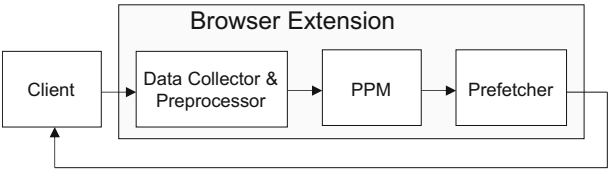

Figure 1 The structure of the application

In order to improve the prediction accuracy, the predictor is enhanced with a confidence mechanism as in [11], which consists in saturating counters, attached to all the links kept in the history. The associated saturating counters are incremented on correct prediction and decremented on misprediction. A prediction is generated only if the confidence counter of the current link is in a predictable state. By using such a confidence mechanism, the number of predictions is lower but the prediction accuracy is significantly higher. For a low traffic level, the proposed predictors are providing only one predicted web page, instead of a multi-page prediction presented in [11].

3.1 Markov predictors

In a first order Markov chain with N states, the current state depends only on the previous state:

$$P[q_t|q_{t-1}, \ldots, q_1] = P[q_t|q_{t-1}] \tag{1}$$

where q_t is the state at time t. In a Markov chain of order R, the current state depends on R previous states [11]:

$$P[q_t|q_{t-1}, \ldots, q_1] = P[q_t|q_{t-1}, \ldots, q_{t-R}] \tag{2}$$

In our application the states are represented by the web pages. The Markov chains can be used to predict the next web page by searching for the current context within the history of visited web pages. The web page that followed the context with the highest frequency is the predicted one. Figure 2 presents an example of prediction with a first order Markov chain: the context is the link 1 and the prediction is 5 since it occurred the most frequently after 1.

In a Markov chain of order R, the context consists in the last R web pages. The Markov predictor used for web page prefetching is presented in the following pseudocode:

```
MARKOV (SEQ, R)
 for k := 0 to R-1 do
   C[k] := SEQ[H-R + k]
 for i := R to H-1 do
   IS_CONTEXT := TRUE
   for k := 0 to R-1 do
     if SEQ[i-R + k] ! = C[k] then
       IS_CONTEXT := false
       break
```

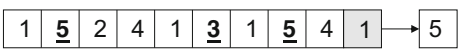

Figure 2 Prediction with a first order Markov chain

```
    if(IS_CONTEXT)
      P[SEQ[i]] := P[SEQ[i]] + 1
  PRED := 0
  MAX := P[0]
  for k := 1 to N-1 do
    if P[k] > MAX then
       MAX := P[k]
       PRED := k
  if MAX > 0 then return PRED
  return -1
```

where R is the order of the Markov chain, SEQ is the observation sequence, C is the context (the last R web pages from SEQ), H is the length of SEQ, P keeps the probability distribution for N distinct web pages, $PRED$ is the predicted web page, and MAX is the number of occurrences of the predicted web page after the context C. If the context C is not found in SEQ, expressed by returning −1, we do not predict any web page. We have also presented the Java implementation of such a Markov predictor in [10].

Tree-, graph-, or transition table-based implementations are also possible, but such methods are inefficient for a high number of web pages and superior order Markov models. The temporal complexity of our Markov model implementation is $\Theta(H \times R)$. The memory request is even lower: it needs to keep the web page sequence of size T in SEQ, the context of size R in C, the probability distribution of size N in P and other few variables ($IS\_CONTEXT$, $PRED$, MAX), which is remarkable especially for systems with memory constraints.

3.2 Prediction by partial matching

The PPM of order R first tries to predict with the Markov model of order R based on the web page sequence SEQ. If the Markov model of order R cannot predict, then the Markov model of order R-1 is used, and so on, the last trial being the Markov model of order 1. In other words, if the Markov model of a certain order cannot provide prediction, it triggers the next lower order Markov chain. Figure 3 presents an example of prediction with a PPM of order 3: the Markov chain of order 3 cannot predict because it could not find the context of 3 links, thus the prediction 3 is generated by the Markov chain of order 2 because it followed once the context within the link sequence.

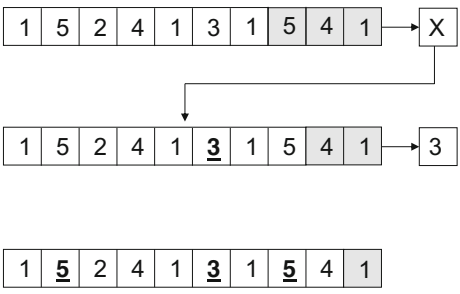

Figure 3 Prediction with PPM of order 3

We have not included the Markov model of order 0 because it is not using any contextual information. The prediction mechanism is presented in Figure 4. The PPM algorithm is given in the following pseudocode:

```
PPM (SEQ, R)
  for r := R downto 1 do
    PRED := MARKOV(SEQ, r)
    if PRED ! = −1 then return PRED
  return −1
```

Since, in the most unfavourable case, the MARKOV function is called R times, the complexity of the PPM algorithm is $\Theta(HxR^2)$. The memory request of the proposed PPM is similar with that of the above presented Markov model, which is remarkable, and significantly better than that of the other existing PPM implementations.

3.3 Graph-based prediction

The graphs are data structures used in many types of applications to model different processes. A graph consists in vertices which can be connected by edges. Figure 5 depicts a directed weighted graph whose vertices represent the accessed links. An edge between two vertices means at least one transition from a link to another in the direction shown by the arrow, whereas the weights are keeping the transition numbers.

The graph-based predictor anticipates the next link as being the vertex whose edge with the current vertex has the highest weight. In fact, it is a first order Markov predictor implemented using a directed graph. The algorithm is presented in the following pseudocode:

```
DGRAPH()
  A := C.GET_PAIRS()
  MAX := A[0]
  foreach Edge E from A
    if E > MAX then MAX := E
  if MAX > 0 then return E.GET_PAIR(C)
  return −1
```

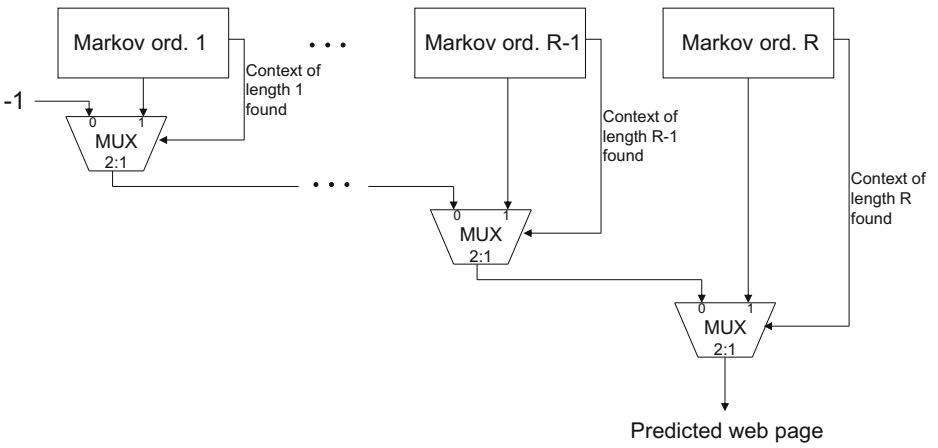

Figure 4 Prediction by partial matching

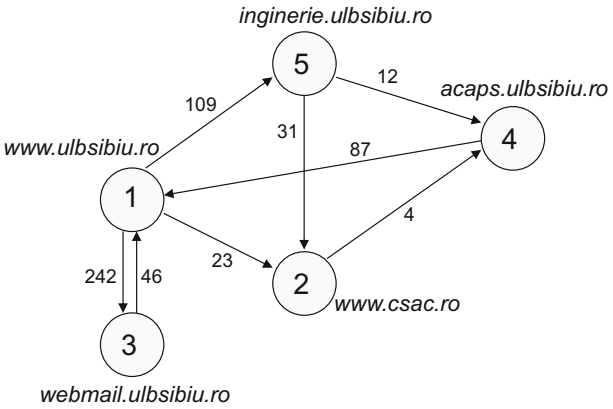

Figure 5 Modeling web page accesses using graphs

where *C* is the vertex corresponding to the current link, *GET_PAIRS* returns a list with all the adjacent vertices and *GET_PAIR* gives the pair vertex of a certain vertex from a given edge *E*. We assume that when the function is called, the directed graph is already constructed based on the sequence of visited web pages.

4 Experimental results

The first step of our research is to analyze the proposed algorithms from the prediction accuracy point of view, by varying their input parameters. Such a study can be better highlighted on a set of log files. Therefore, the above presented algorithms have been implemented in Java and evaluated on the BU benchmarks.

The BU dataset was generated by the "Ocean Group Research" from Boston University [5] and consists in log files collected during 7 months on 37 workstations, spanning the timeframe from 21 November 1994 to 8 May 1995. Each log file name contains a user ID number, the machine on which the session took place and the Unix timestamp when the session started. Each line in a log corresponds to a single URL requested by the user; it contains the machine name, the timestamp when the request was made, the URL, the size of the document (including the overhead of the protocol) and the object retrieval time in seconds. The average number of links in the BU benchmarks is 1905. After we have pre-processed the original log files, as we described in Section 3, we named them *con*X where X is the user ID.

First, we have evaluated a 3rd order PPM by varying the size of the web page sequence, also called history (H). We used 4-state confidence counters, identified in [11] as being optimal for web page prediction. The results are presented in Figure 6.

As Figure 6 presents, the highest average prediction accuracy has been obtained with a history length of 600. We can observe an increase in accuracy until H=600 and a fall after that, meaning that a rich history can lead to higher accuracy, but starting with a certain length it can behave as noise and the accuracy decreases. In fact, the accuracies obtained with history lengths of 500, 600 and 700 are very close – 70.14, 70.42 and 70.40 %, respectively –, therefore, we consider that the optimal history length is 500.

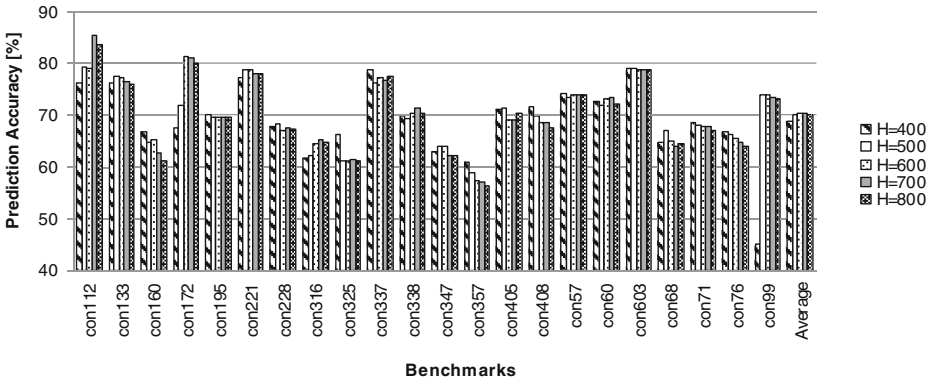

Figure 6 Prediction accuracy of 3rd order PPM (R=3) for different histories

We have continued our study by varying the order of the confidence-based PPM, considering the optimal history length of 500 web pages. The results are presented in Figure 7 which shows that the prediction accuracy increases with the order of the PPM, but there is marginal benefit of increasing the order beyond 4. On some benchmarks we can observe inflexion points followed by accuracy decrease. For those benchmarks, the Markov chains of higher orders than the inflexion point have more mispredictions at higher complexity. The average prediction accuracies show that the PPM of order 4 is the optimal.

We have compared the optimal PPM (H=500, R=4) with a PPM having the same configuration but without the 4-state confidence mechanism. As Figure 8 depicts, the 4-state confidence counters have a very high benefit. With this confidence mechanism we are able to selectively predict only from confident web pages, increasing thus the average prediction accuracy from 31.54 to 71.11 %. Figure 9 shows how the prediction rate (the number of predicted web pages divided to the total number of web pages) is influenced by the selectivity of the 4-state confidence mechanisms in the case of the optimal PPM (H=500, R=4).

It can be observed that the attached 4-state confidence counters reduce the number of predictions from 62 to 10 % but improve the accuracy from 31.54 to 71.11 %. In order to not increase too much the network traffic, we prefer to predict fewer times but accurately.

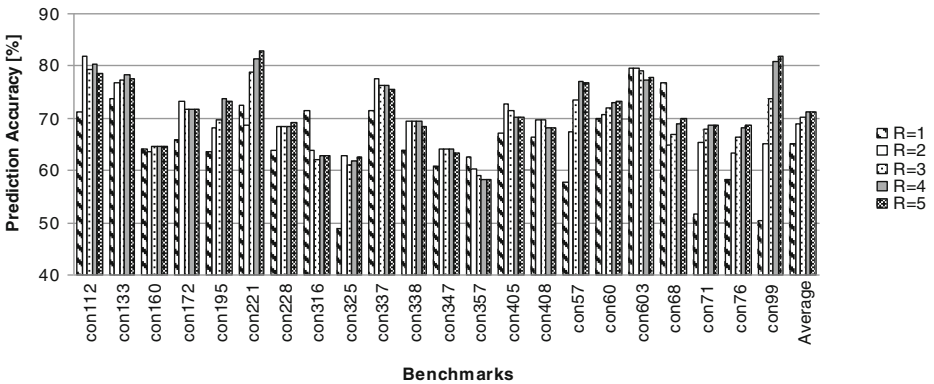

Figure 7 Prediction accuracy obtained with PPM of different orders and history of 500 (H=500)

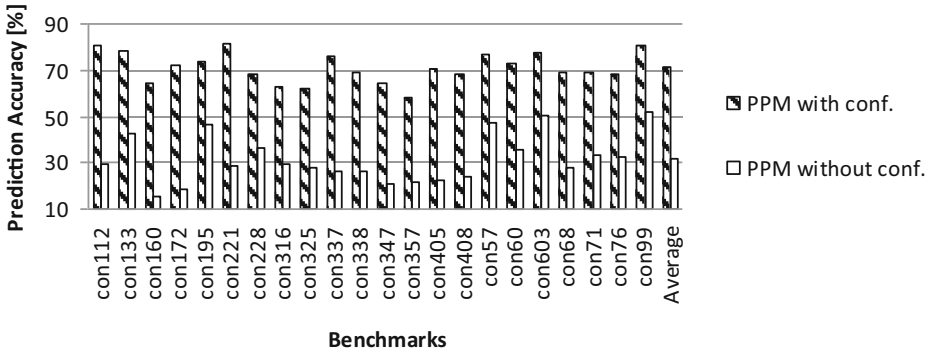

Figure 8 Comparing the optimal PPM (H=500, R=4) with and without confidence

Finally, Figure 10 depicts the efficiency of our PPM implementation opposite to a graph-based implementation, as simulation time. The implementations which are implying transition tables, trees or graphs, as in [6, 23, 25, 31], become very inefficient or even intractable for high number of distinct web pages and superior order models, because of the high state complexity. In Figure 10, *Graph 1* is a first order Markov predictor implemented using a directed weighted graph, whereas Markov 1 and PPM 4 are our first order Markov and 4th order PPM implementations. We reported the time necessary to process and eventually predict all the links from each benchmarks. As the evaluations show, our proposed Markov model implementation is more efficient than the graph-based one, and more important, our proposed PPM implementation has almost the same time-efficiency even for higher orders. The average execution time with the first order graph based Markov predictor is 29.77 s, while with our first order Markov predictor and our 4th order PPM it was only 10.18 and 11.27 s, respectively.

5 Conclusions and further work

In this study we proposed prediction based web prefetching methods in order to reduce the delays necessary to load the web pages and files visited by the users. We have presented an efficient way to implement the prediction by partial matching as simple searches in the

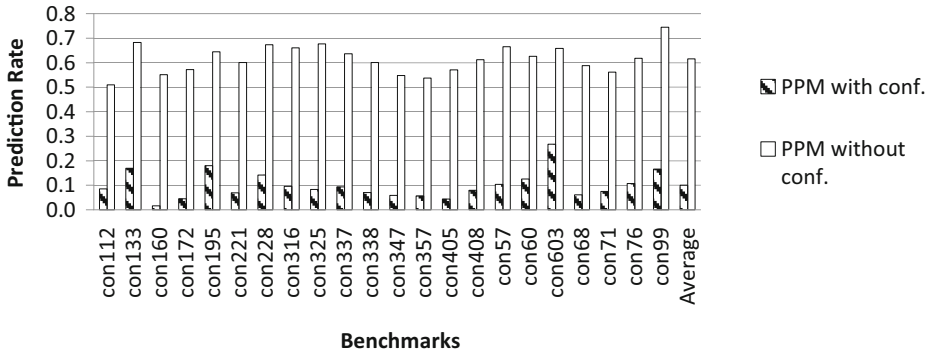

Figure 9 Comparing the prediction rates of the optimal PPM (H=500, R=4) with and without confidence

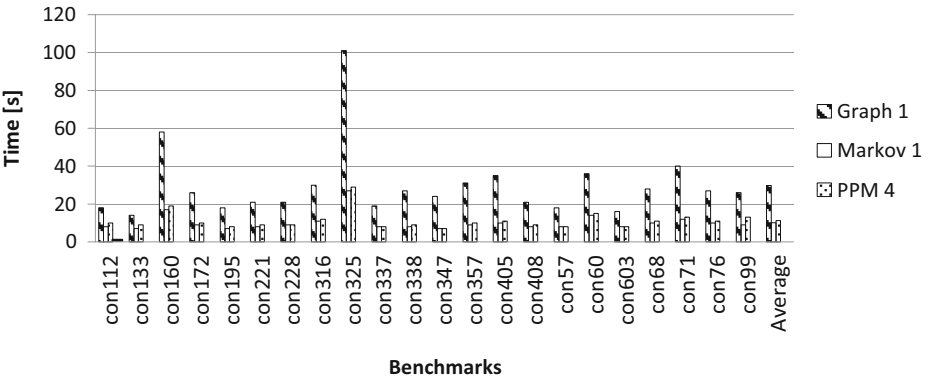

Figure 10 Simulation time

observation sequence. By avoiding tree-, table- and graph-based implementations, the memory necessity of the proposed PPM is similar with that of the Markov model. This low complexity is remarkable, especially for the superior order models with high number of states, and significantly better than in the other existing PPM implementations. Our time-evaluations show that the prediction latency of the proposed model is just slightly affected by the order of the model, opposite to the implementations using the above mentioned data structures with which the order of the model affects exponentially the complexity.

The confidence mechanism has a high benefit because it allows to predict selectively from high-confidence contexts, decreasing thus the prediction rate from 62 to 10 %, but increasing in the same time the accuracy from 31.54 to 71.11 %. The optimal prediction accuracy of 71.11 % was obtained with the PPM of order 4 using a history of 500 web pages and 4-state confidence counters.

A further work direction is the analysis of using neural networks for web page prediction as a first prediction level in a two-stage predictor. Another further work direction consists in developing and evaluating different hybrid predictors. The PPM is a hybrid predictor with prioritization-based static component selection and we intend to evaluate also dynamic selection in order to use the best predictor at a certain moment, for possible prediction accuracy improvements. Finding and exploiting similarity among users is a research challenge in itself.

References

1. Ban, Z., Gu, Z., Jin, Y.: An online PPM prediction model for web prefetching. The 9th ACM International Workshop on Web Information and Data Management, pp. 89–96. Lisboa (2007)
2. Canali, C., Colajanni, M., Lancellotti, R.: Adaptive algorithms for efficient content management in social network services. 10th International Conference on Computer and Information Technology, pp. 68–75. (2010)
3. Ciobanu, D., Dinuca, C.E.: Predicting the next page that will be visited by a web surfer using page rank algorithm. Int. J. Comput. Commun. **6**(1), 60–67 (2012)
4. Cleary, J., Witten, I.: Data compression using adaptive coding and partial string matching. IEEE Trans. Commun. **32**(4), 396–402 (1984)
5. Cunha, C.A., Bestavros, A., Crovella, M.E.: Characteristics of WWW client traces. Technical report TR-95-010. Boston University, Department of Computer Science (1995)

6. Deshpande, M., Karypis, G.: Selective Markov models for predicting web-page accesses. ACM Trans. Internet Technol. **4**(2), 163–184 (2004)
7. Domènech, J., Pont, A., Sahuquillo, J., Gil, J.A.: An experimental framework for testing web prefetching techniques. The 30th EUROMICRO Conference, pp. 214–221. (2004)
8. Domènech, J., Sahuquillo, J., Pont, A., Gil, J.A.: How current web generation affects prediction algorithms performance. Proceedings of SoftCOM International Conference on Software, Telecommunications and Computer Networks. Split, Croatia (2005)
9. Dubey, S., Mishra, N.: Web page prediction using hybrid model. Int J Comput Sci Eng **3**(5), 2170–2176 (2011)
10. Gellert, A., Florea, A.: Investigating a New design pattern for efficient implementation of prediction algorithms. J. Digit. Inf. Manag. **11**(5), 366–377 (2013)
11. Gellert, A., Florea, A.: Web page prediction enhanced with confidence mechanism. J Web Eng **13**(5–6), 507–524 (2014)
12. Guo, Y.Z., Ramamohanarao, K., Park, L.A.F.: Web page prediction based on conditional random fields. The 18th European Conference on Artificial Intelligence, pp. 251–255. (2008)
13. Hasan, M.A., Chaoji, V., Salem, S., Zaki, M.: Link prediction using supervised learning. Proceedings of SDM 06 Workshop on Link Analysis, Counterterrorism and Security. Bethesda (2006)
14. Huang, Q., Yang, Q., Huang, J.Z., Ng, M.K.: Mining of Web-Page Visiting Patterns with Continuous-Time Markov Models, pp. 549–558. Springer-Verlag, Berlin Heidelberg (2004)
15. Huang, Z.: Link prediction based on graph topology: the predictive value of generalized clustering coefficient. Proceedings of the Workshop on Link Analysis: Dynamics and Static of Large Networks. Philadelphia (2006)
16. Jin, X., Xu, H.: An approach to intelligent web pre-fetching based on hidden Markov model. Proceedings of the 42nd Conference on Decision and Control, vol. 3, pp. 2954–2958. Maui (2003)
17. Kaushal, P.: Hybrid Markov model for better prediction of web page. IJSRP. **2**(8), (2012)
18. Khalil, F., Li, J., Wang, H.: Integrating recommendation models for improved web page prediction accuracy. Proceedings of the 31st Australasian Conference on Computer Science, vol. 74, pp. 91–100. (2008)
19. Khalil, F., Li, J., Wang, H.: An integrated model for next page access prediction. IJKWI **1**(1/2), 48–80 (2009)
20. Khanchana, R., Punithavalli, M.: Web page prediction for web personalization: a review. GJCST **11**(7), 39–44 (2011)
21. Lee, J., Shi, Y., Wang, F., Lee, H., Kim, H.K.: Advertisement Clicking Prediction by Using Multiple Criteria Mathematical Programming. WWWJ (2015). doi:10.1007/s11280-015-0353-1
22. Murata, T., Moriyasu, S.: Link prediction of social networks based on weighted proximity measures. IEEE/WIC/ACM International Conference on Web Intelligence, pp. 85–88. (2007)
23. Palpanas, T., Mendelzon, A.: Web prefetching using partial match prediction. Proceedings of the 4th International Web Caching Workshop. San Diego (1999)
24. Pitkow, J., Pirolli, P.: Mining longest repeating subsequences to predict World Wide Web surfing. The 2nd USENIX Symposium on Internet Technologies & Systems, vol. 2, pp. 11–14. Boulder (1999)
25. Singhai, N., Nigam, R.K.: A novel technique to predict oftenly used web pages from usage patterns. IJETTCS **1**(4), 49–55 (2012)
26. Su, Z., Yang, Q., Zhang, H. J.: A prediction system for multimedia pre-fetching in internet. Proceedings of the eighth ACM international conference on Multimedia, pp. 3–11. New York (2000)
27. Temgire, S., Gupta, P.: Review on web prefetching techniques. IJTEEE **1**(4), 100–105 (2013)
28. Wan, M., Jönsson, A., Wang, C., Li, L., Yang, Y.: Web user clustering and web prefetching using random indexing with weight functions. Knowl. Inf. Syst. **33**(1), 89–115 (2012)
29. Wu, Y.-H., Chen, A.L.P.: Prediction of web page accesses by proxy server log. WWWJ **5**(1), 67–88 (2002). doi:10.1023/A:1015750423727
30. Zheng, Z., Wei, W., Liu, C., Cao, W., Cao, L., Bhatia, M.: An Effective Contrast Sequential Pattern Mining Approach to Taxpayer Behavior Analysis. WWWJ (2015). doi:10.1007/s11280-015-0350-4
31. Zhu, J., Hong, J., Hughes, J.G.: Using Markov Chains for Link Prediction in Adaptive Web Sites, pp. 60–73. Springer, Berlin Heidelberg (2002)