TWO PRUNING METHODS FOR ONLINE PPM WEB PAGE PREDICTION

Alborz Moghaddam and Ehsanollah Kabir
Department of Electrical and Computer Engineering, Tarbiat Modares University, Tehran, Iran
{a.moghaddam, kabir}@modares.ac.ir

Keywords: Prediction, Memory optimization, Pruning, PPM.

Abstract: The Web access prediction gets significant attention in recent years. Web prefetching and some personalization systems use prediction algorithms. Most current applications that predict the next web page have an offline part that does the data preparation task and an online part that provides personalized content to the users based on their current navigational activities. We use PPM for modelling user navigation history. In standard PPM, many states of the model are rarely useful for prediction and can be eliminated without affecting the performance of the model. In this paper we propose two pruning methods. Using these methods we present an online prediction model that fits in the memory with good prediction accuracy. A performance evaluation is presented using real web logs. This evaluation shows that our methods effectively decrease the memory complexity.

1 INTRODUCTION

Web mining has become an important research area in recent years. It can be used for different purposes such as: improving the web cache performance (Padmanabhan and Mogul, 1996; Palpanas, 2000), detecting the user interest and recommending related pages or goods for e-commerce web sites (Sarwar and Karypis, 2000), improving search engines results (Zhang and Dong, 2002), and personalizing web content as the users like (Anand , Kearney and Shapcott, 2007). Understanding the user navigation pattern and then predicting the next pages is the main problem.

Prediction by Partial Match, PPM, (Palpanas 2000; Deshpande and Karypis 2004) is a commonly used technique in prediction. Improvements on the efficiency of PPM were examined in (Deshpande and Karypis 2004) where three pruning criteria were proposed: a) support-pruning, b) confidence-pruning and c) error-pruning. With these pruning methods states of the Markov model are pruned in case they do not appear very frequently. In (Ban, Gu and Jin, 2007) the authors proposed an online PPM model. They compute each node's outgoing entropy and then normalize it. Their algorithm always chooses the nodes with low normalized entropy. In their model, prediction accuracy rate and longest match rule are also applied.

The content of many web sites are dynamic and new pages can be added to the site dynamically. So we need an online model to consider the changes of the web site and the user behaviour. The memory efficiency is an important factor for an online algorithm. Most models proposed for web page prediction are not online (Pitkow and Pirolli, 1999; Chen and Zhang, 2003) and online models such as (Ban, Gu and Jin, 2007) may soon become too big to fit in memory.

In this paper we use PPM algorithm to model user navigation behaviour and present two techniques for pruning the model. We do not build per-user predictive model. Individual models require more space and may be less accurate, because they see less data than a global model (Deshpande and Karypis, 2004). The execution time required for adding a new page to the model and predicting the next user action are significantly affected by the size of the related Markov model, so by pruning the model we enhance the model efficiency. The structure of the paper is as follows. In Section 2, we present some related works. The online PPM model creation and prediction are proposed in Section 3. Pruning methods are explained in Section 4. In Section 5 we present experimental results, and section 6 is conclusion.

2 RELATED WORKS

Various models have been proposed for modelling the user navigation behaviour and predicting the next requests of users. According to (Pierrakos, Paliouras, Papatheodorou and Spyropoulos, 2003), association rules, sequential pattern discovery, clustering, and classification are most popular methods for web usage mining. Association rules (Agrawal, Mannila, Srikant, Toivonen and Verkamo, 1996) were proposed to capture the co-occurrences of buying different items in a supermarket shopping. Association rules indicate groups that are related together. Methods that use association rules can be found in (Yang, Li and Wang, 2004) too.

The prediction scheme described in (Padmanabhan and Mogul, 1996) used a dependency graph, DG, to model user navigation behaviour. The DG prediction algorithm constructs a dependency graph that describes the pattern of user page requests. Every page visited by user is represented by a node in the dependency graph. There is an arc from node A to B if and only if at some point in time a client accessed to B within w accesses after A, where w is the lookahead window size. The weight of the arc is the ratio of the number of accesses to B within a window after A to the number of accesses to A itself. A DG is effectively a first-order Markov model. In this method the consecutiveness of requests are not applied.

Markov models contain precise information about users' navigation behaviour. They are most widely used in sequential pattern discovery for link prediction. Lower order Markov models are not very accurate in predicting the user's browsing behaviour, since these models do not look far into the past to correctly discriminate the different observed patterns. Higher order Markov models give better predictive precision with reduced hit rate. All-kth-Order Markov model maintains Markov predictors of order i, for all $1 \leq i \leq k$. This model improves prediction coverage and accuracy but the number of states in this model grows exponentially when the order of model increases. Improvements on the efficiency of PPM are examined in (Deshpande and Karypis, 2004). Three pruning criteria are proposed: a) support-pruning, b) confidence-pruning and c) error-pruning. The subject of (Deshpande and Karypis, 2004) is mainly the efficiency. The resulting model, called selective Markov model has a low state complexity. But this model is not online and can not be incrementally updated.

The Longest Repeating Subsequence, LRS PPM (Pitkow and Pirolli, 1999) stores a subset of all passes that are frequently accessed. It uses longest repeated sequence to predict next request. In this model each path occurs more than some Threshold T, where T typically equals one. In (Chen and Zhang, 2003), popularity-based PPM is proposed. In this model, the tree is dynamically updated with a variable height in each set of branches where a popular URL can lead a set of long branches, and a less popular URL leads to a set of short ones.

The study in (Ban, Gu and Jin, 2007) presents an online method for predicting next user request. In this model the entropy of a node is an important factor in prediction. But in this model, the memory efficiency of algorithm is not considered.

The techniques that mentioned above, work well for web sites that do not have a complex structure and do not dynamically generate web pages. Complex structure of a web site led to large number of states in a Markov model and so it needs much runtime requirements such as memory and computation power.

3 ONLINE PPM PREDICTION MODEL

In this paper we apply our pruning methods on a prediction tree based on PPM. The PPM model has an upper bound for its context length. The context length is the sequence length that preceding the current symbol. A kth order PPM model keeps the contexts of length 0 to k. The predictor is represented by a tree. The number on each edge records the number of times the request sequence occurs in the path from the root node to end node of that edge. The PPM prediction tree for sequences ABCDE, ABCA, CACD, BCD and ADAB is displayed in Figure 1.

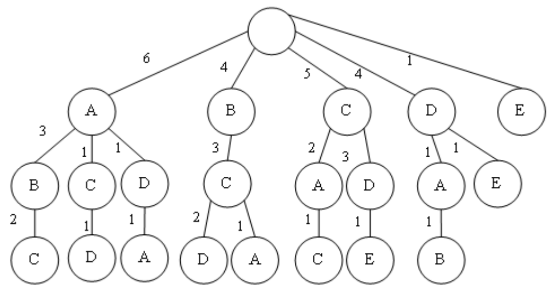

Figure 1: The PPM prediction tree for sequences ABCDE, ABCA, CACD, BCD and ADAB.

We create the PPM prediction tree, online. An online prediction method needs not rely on time-consuming pre-processing of the available historical data in order to build a prediction model. The pre-processing is done whenever we have a new request. The advantage of the PPM model is that it is not very complex .But the tree records every accessed sequence of URLs, so it needs too much space in the server. We apply two pruning methods on PPM model and decrease the model size and hence decrease the required computational power of our model. We apply our pruning methods after a specified number of page requests.

Figure 2 shows the learning process of our model. As you can see, when a new page is requested then:

A. Do an online filtering and ignore the image, media and script files.

B. Find the user active session corresponding to new request.

C. Lines 6 to 12 are for clearing our evaluation method.

D. Learn the user request list. Updates are performed in prediction tree according to PPM algorithm.

E. Prune the prediction tree after each 10000 page requests in this code.

```
1. PPMTree: Prediction tree
2. PRT = 10000 ; //Pruning Tree After
Each 10000 Page Request
3. requestCount = 0 ; //Count the
requests
4. while (there is Request){
1. {
1.  Request req = GetNextRequest();
2.  If IsFileter(req) continue;
3.  requestCount = requestCount + 1;
4.  userSession =
GetActiveSession(req);
5.  pSet =  NextPrediction(PPMTree,
userSession);
6.  if(pSet!= null)
7.  {
8.   if (req in pSet)// we could
predict the next requests
9.   {  positive++;} // we have a
correct prediction
10.     else
11.  { negative++;}// we have a wrong
prediction
12. }
13. LearnString(PPMTree,userSession);
//learn the string in PPM manner.
14. If ( requestCount == PRT )
15. {
16.      pruneTree(PPMTree);
17.      requestCount = 0;
18. }
19. }//End while
```

Figure 2: Learning a new user sequence in prediction tree.

`LearnString` Method, update the prediction tree according to PPM algorithm.

4 OUR PRUNING METHODS

We propose two methods for pruning the prediction tree and decreasing the nodes count and memory for constructing the prediction model. By these methods we remove the low information paths from the model, keeping only frequently accessed paths. These methods do not noticeably affect prediction accuracy, but they significantly reduce the storage requirement.

4.1 Child Count Based Pruning, CCBP

The CCBP is described below:

Traverse the tree using Breadth First Search, BFS algorithm and for the nodes that having more child nodes than a specified threshold T_c, remove the child nodes having lower weight until the child nodes count equals to threshold T_c.

T_c must be less than prediction set size. As you have seen in Section 4, the prediction tree is pruned after a specified number of page requests, PRT. When a new request is added to the tree as a child node, it has chances to overtake one of its siblings, until page requests count does not equal PRT. Experimental results show that this method decreases the nodes counts of prediction tree more that 80 percent with no effective decreasing in prediction accuracy. In Figure 3 the prediction tree after pruning by CCBP method is displayed. We set T_c = 2. If two branches have equal weight then we prune one of them randomly like branch `A-D-A' that is pruned.

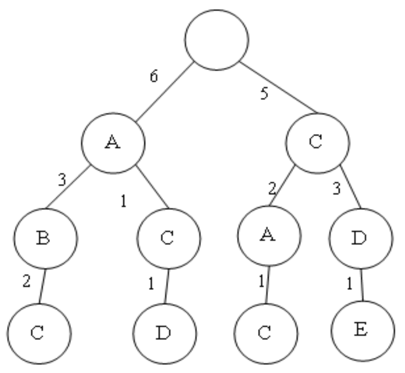

Figure 3: The prediction tree of figure 1, after pruning by CCBP method.

4.2 Path Probability Based Pruning, PPBP

Pruning nodes which have lower probability than a specified Threshold, is a common pruning technique (Chen and Zhang 2003). The access probability of a node is the ratio between the number of accesses to it and the number of accesses to its parent node. We propose a more comprehensive probability pruning method which considers probability of overall path from the root node to a specified node. A path S_i of length n is represented by $S[0..n]=\{x_0,x_2,\ldots,x_n\}$, where x_i is a node, n is length of S and x_0 is root node. We define $p(x_i)$ as the probability of node xi. Path probability, pp, of node x_i is the multiplication of probabilities of all nodes in the path from root node to x_i. For example the path probability of path, ABC in Figure 1, is (6/22) * (3/6)*(2/3) = 1/11. pp of root node is one. We show how $pp(x_i)$ is calculated below.

```
w(xi) = weight of node xi that is same
as weight of its descendent edge.
p(xi) = w(xi)/ w(parentOf(xi))
```

$$pp(xi) = \begin{cases} 1 & i = 0 \\ \prod_{i=1}^{n} p(xi) & i > 0 \end{cases}$$

Figure 4: Calculating the path probability.

The nodes that have pp lower than a specified threshold T_p, are pruned. It can be easily done using Depth First Search, DFS to traverse the tree. As in the CCBP method, the pruned nodes count depended on the T_p that can be set regarding to server computational power and available memory. Below we show prediction tree of Figure 1, after pruning by PPBP method, with Tp = 0.9.

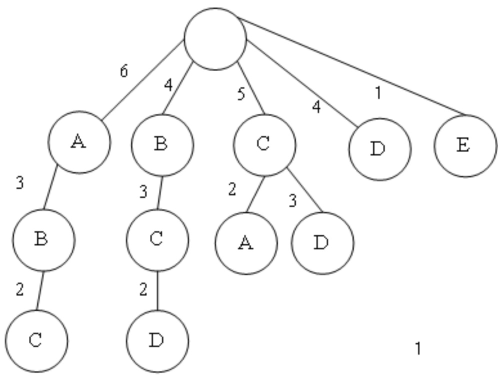

Figure 5: The prediction tree of Figure 1 after pruning by PPBP method.

5 EXPERIMENTAL RESULT

The web log we tested was NASA log from the NASA Kennedy Space Centre server in Florida available from the site http://ita.ee.lbl.gov/html /traces.html. It contains 1,569,898 requests and 72,198 IPs aggregated as 51,132 sessions involving 4,737 pages. For this log, if a user is idle for more than 30 minutes, we assume that the next request from the user starts a new session. The longest matching rule is used in all of tested models.

We tested the CCBP, PPBP, LRS and 5-Order PPM. For all algorithms we used a global model for prediction and proposed 10 pages as a prediction set. If the next request of the user is member of this set, our prediction is considered as a correct prediction. For pruning the trees we set the T_c to 15 and T_p to 0.00005. We used two performance metrics, hit-rate and precision to evaluate our methods. They are defined as follows.

Precision. The ratio of correct prediction divided by the number of requests that the model has prediction for them.

Hit Rate. The ratio of correct predictions divided by total number of requests.

All Experiments are performed on a Pentium 4, Core 2 Duo 2.33 GHz with a 1G main memory running Microsoft Windows XP 2002.

Figure 6 shows that CCBP and PPBP consume much less memory than PPM and LRS models. This Figure shows the prediction tree nodes count versus the number of page requests that are processed. The prediction tree nodes count can be controlled with T_p and T_c.

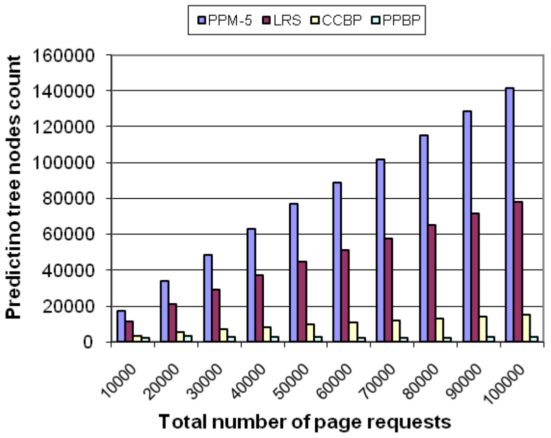

Figure 6: Prediction tree nodes count vs page request count

PPM-5 is more time consuming than other models, because PPM-5 model is larger than others.

Figure 7 shows the time spent for processing 100000 page requests. The time that spent for processing user request is dependent on model size. CCBP spends less time to process page requests because of its tiny model and simple nature. As you can see, using pruning methods led to a faster user modelling system.

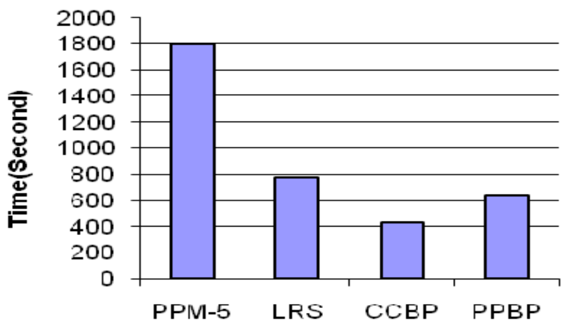

Figure 7: Time spent to create prediction tree after processing 100000 page requests.

Figure 8 compares the precision of four models. The precision of the PPM model is slightly higher than precision of other models. But as you can see in Figure 8, there is no considerable difference between these models.

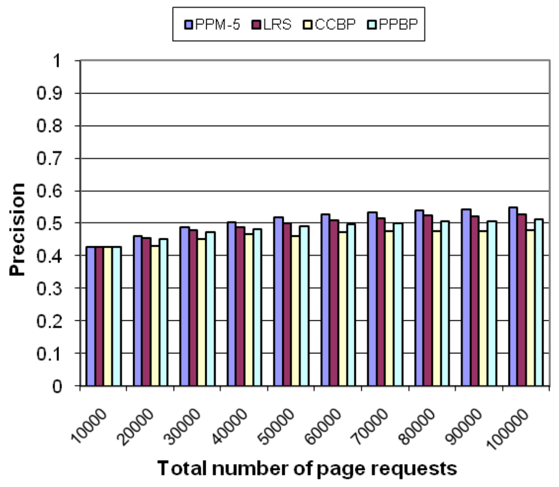

Figure 8: Prediction precision vs page requests count.

Figure 9 compares the hit rate of four models. The hit rate of PPM model is higher than the others. Because in PPM all sequences that their length are less than specified context length are maintained. But in other methods some user sequences are eliminated. The difference is not high and this is a trade off between the memory consumption and hit rate. If the memory and speed are critical, Pruning methods can be used. In such systems the decrease in required memory is more important than lower hit rate.

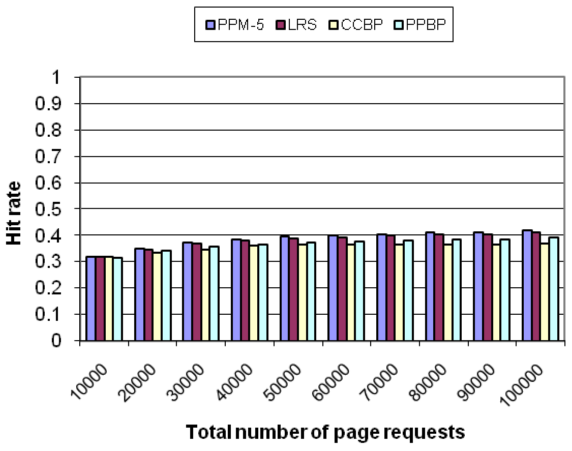

Figure 9: Prediction hit rate vs page requests count.

6 CONCLUSIONS AND FURTHER WORKS

Modelling and predicting user surfing paths involves a tradeoffs between model complexity and predictive accuracy. In this paper we proposed two pruning techniques that attempt to reduce model complexity while retaining predictive accuracy. In CCBP, the child nodes of nodes that their child nodes count are more than specified threshold are pruned. In PPBP the overall path probability of a node calculated and the nodes that have low path probability are pruned. We have tested our methods and the experimental results and show that our methods consume much less memory than others. Our models need no training or pre-processing of the historical data. Our tree pruning methods are suitable for dynamic web sites and can be considered for future research for temporal modelling of user session. We like to discuss how the best values of T_c and T_p can be found according to desired precision and prediction tree nodes count.

ACKNOWLEDGEMENTS

This work was supported in part by the Iran Telecommunication Research Centre, ITRC, under Contract T-500-20593, TMU-87-01-08.

REFERENCES

Agrawal, R., H. Mannila, et al. (1996). Fast discovery of association rules, American Association for Artificial Intelligence Menlo Park, CA, USA: 307-328.

Anand, S. S., P. Kearney, et al. (2007). Generating semantically enriched user profiles for Web personalization, ACM Press New York, NY, USA.

Ban, Z., Z. Gu, et al. (2007). An online PPM prediction model for web prefetching, ACM New York, NY, USA: 89-96.

Begleiter, R., R. El-Yaniv, et al. (2004). On Prediction Using Variable Order Markov Models. 22: 249-250.

Borges, J. and M. Levene (1999). Data Mining of User Navigation Patterns, Springer-Verlag London, UK: 92-111.

Borges, J. and M. Levene (2005). Generating dynamic higher-order Markov models in web usage mining, Springer. 3721: 34-45.

Chen, X. and X. Zhang (2003). A Popularity-Based Prediction Model for Web Prefetching, IEEE Computer Society.

Chim, H. and X. Deng (2007). A new suffix tree similarity measure for document clustering, ACM Press New York, NY, USA: 121-130.

Curewitz, K. M., P. Krishnan, et al. (1993). Practical prefetching via data compression, ACM Press New York, NY, USA. 22: 257-266.

Davison, B. D. (2004). Learning Web Request Patterns, Springer.

Deshpande, M. and G. Karypis (2004). Selective Markov models for predicting Web page accesses, ACM Press New York, NY, USA. 4: 163-184.

Hipp, J., U. Güntzer, et al. (2000). Algorithms for association rule mining—a general survey and comparison, ACM Press New York, NY, USA. 2: 58-64.

Katsaros, D. and Y. Manolopoulos (2005). A Suffix Tree Based Prediction Scheme for Pervasive Computing Environments, Springer: 267-277.

Nanopoulos, A., D. Katsaros, et al. (2002). Exploiting Web Log Mining for Web Cache Enhancement, Springer.

Padmanabhan, V. N. and J. C. Mogul (1996). Using predictive prefetching to improve World Wide Web latency. 26: 22-36.

Palpanas, T. (2000). Web Prefetching Using Partial Match Prediction, National Library of Canada= Bibliothèque nationale du Canada.

Pierrakos, D., G. Paliouras, et al. (2003). Web Usage Mining as a Tool for Personalization: A Survey, Springer. 13: 311-372.

Pitkow, J. and P. Pirolli (1999). Mining longest repeating subsequences to predict world wide web surfing, USENIX Association Berkeley, CA, USA: 13-13.

Sarwar, B., G. Karypis, et al. (2000). Analysis of recommendation algorithms for e-commerce, ACM Press New York, NY, USA: 158-167.

Yang, Q., T. Li, et al. (2004). Building Association-Rule Based Sequential Classifiers for Web-Document Prediction, Springer. 8: 253-273.

Zamir, O. and O. Etzioni (1998). Web document clustering: a feasibility demonstration, ACM Press New York, NY, USA: 46-54.

Zhang, D. and Y. Dong (2002). A novel Web usage mining approach for search engines, Elsevier. 39: 303-310.

ZhangYang, W. (2005). Mining sequential association-rule for improving Web document prediction: 146-151.