
Borges, J and Levene, M (2005). Generating dynamic higher-order Markov models in web usage mining. *Lecture Notes in Computer Science* 3721, 34-45.

Generating Dynamic Higher-Order Markov Models in Web Usage Mining

José Borges[1] and Mark Levene[2]

[1] School of Engineering, University of Porto, R. Dr. Roberto Frias, 4200 - Porto, Portugal
jlborges@fe.up.pt

[2] School of Computer Science and Information Systems, Birkbeck, University of London, Malet Street, London WC1E 7HX, U.K.
mlevene@dcs.bbk.ac.uk

Abstract. Markov models have been widely used for modelling users' web navigation behaviour. In previous work we have presented a dynamic clustering-based Markov model that accurately represents second-order transition probabilities given by a collection of navigation sessions. Herein, we propose a generalisation of the method that takes into account higher-order conditional probabilities. The method makes use of the state cloning concept together with a clustering technique to separate the navigation paths that reveal differences in the conditional probabilities. We report on experiments conducted with three real world data sets. The results show that some pages require a long history to understand the users choice of link, while others require only a short history. We also show that the number of additional states induced by the method can be controlled through a probability threshold parameter.

1 Introduction

Modelling user web navigation data is a challenging task that is continuing to gain importance as the size of the web and its user-base increase. Data characterising web navigation can be collected from server or client-based log files, enabling the reconstruction of user navigation sessions [15]. A session is usually defined as a sequence of pages viewed by a user within a given time window. The subarea that studies methods to extract patterns from navigation data has been called *web usage mining* and such methods have been applied in several contexts including personalisation, link prediction, e-commerce analysis, adaptive web site organisation and web page pre-fetching [10].

Several authors have proposed the use of Markov models to represent a collection of user web navigation sessions. Pitkow et al. [12] proposed a method to induce the collection of longest repeating sub-sequences, while Deshpande et al. [7] proposed a technique that builds $k^{th}-order$ Markov models and combines them to include the highest order model covering each state. On the other hand, Sarukkai [13] presented a study showing that Markov models have potential use

in link prediction applications, while Zhu et al. [16] inferred a Markov model from user navigation data to measure page co-citation and coupling similarity.

An alternative method of modeling navigation sessions are tree-based models. Schechter et al. [14] use a tree-based data structure that represents the collection of paths inferred from log data to predict the next page accessed, while Dongshan and Junyi [8] proposed a hybrid-order tree-like Markov model to predict web page access. In addition, Chen and Zhang [6] use a Prediction by Partial Match tree that restricts the roots to popular nodes.

In previous work we proposed to model a collection of user web navigation sessions as a Hypertext Probabilistic Grammar (HPG) [1, 2]. A HPG corresponds to a first-order Markov model, which makes use of the N-gram concept [5] to achieve increased accuracy by increasing the order of the Markov chain; for the full details on the HPG concept see [2]. In [2] an algorithm to extract the most frequent traversed paths from user data was proposed, and in [3] we have shown that the algorithm's complexity is, on average, linear time in the number of states of the grammar. In [4] we extended the HPG model with a dynamic clustering-based method that uses state cloning [9] to accurately represent second-order conditional probabilities; the method is presented in Section 2. In this work we generalise the method given in [4] to higher-order conditional probabilities.

Most current web mining systems use techniques such as clustering, association rule mining and sequential pattern mining to search for patterns in navigation records [10], and do not take into account the order in which pages were accessed. This limitation has been tackled by building a sequence of higher-order Markov models with a method that chooses the best model to use in each case [7]. However, we argue that a method to produce a single model representing the variable length history of pages has, so far, been missing.

The method we propose in Section 3 aims to fill that gap. By using the cloning operation we duplicate states corresponding to pages that require a longer history to understand the choice of link that users made. In this way the out-links from a given state reflect the n-order conditional probabilities of the in-paths to the state. In addition, the proposed model maintains the fundamental properties of the HPG model [1], while providing a suitable platform for utilising an algorithm for mining the navigation patterns that takes into account the order of page views.

In Section 2 we review the essential of the dynamic clustering method, in Section 3 we extend the method to model higher-order probabilities, and in Section 4 we present the experimental results. Finally, in Section 5 we give our concluding remarks.

2 Background

In previous work [1, 2] we proposed to model user navigation data as a Hypertext Probabilistic Grammar (HPG), which corresponds to a first-order Markov model. We now review the HPG model with the aid of an example.

Consider a web site with seven web pages, $\{A_1, A_2, \ldots, A_7\}$, and the collection of navigation sessions given on the left-side of Figure 1 (NOS represents the number of occurrences of each session). A navigation session gives rise to a sequence of pages viewed by a user within a given time window. To each web page there corresponds a state in the model. In addition, the start state, S, represents the first state of every navigation session, and the a final state, F, represents the last state of every navigation session. There is a transition corresponding to each pair of pages visited in sequence, a transition from S to the first state of a session, and a transition from the last state of a session to F. The model is incrementally built by processing the complete collection of navigation sessions.

Session	**NOS**
A_1, A_2, A_3	3
A_1, A_2, A_4	2
A_5, A_2, A_3	1
A_5, A_2, A_4	1
A_6, A_2, A_3	2
A_6, A_2, A_4	1
A_1, A_2, A_4, A_7	1
A_5, A_2, A_4, A_7	3
A_6, A_2, A_4, A_7	2

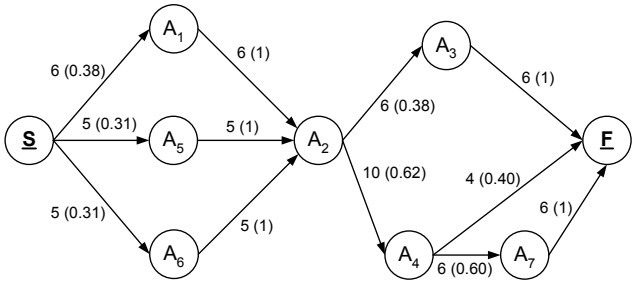

Fig. 1. A collection of user navigation sessions and the corresponding first-order model

A transition probability is estimated by the ratio of the number of times the transition was traversed and the number of times the anchor state was visited. The right-side of Figure 1 shows a representation of the first-order model corresponding to the input sessions. Next to a link, the first number gives the number of times the link was traversed and the number in parentheses gives its estimated probability.

In [4] we proposed a method to increase the HPG precision in order to accurately represent second-order probabilities. The method makes use of a cloning operation, where a state is duplicated if first-order probabilities diverge from the corresponding second-order probabilities. In addition, the method uses a clustering algorithm to identify the best way to distribute a state's in-links between a state and its clones. We now present the essential properties of the model proposed in [4], which we extend to higher-order probabilities in Section 3.

Given a model with states $\{S, A_1, ..., A_n, F\}$, we let w_i represent the number of times the page corresponding to A_i was visited, $w_{i,j}$ be the number of times the link from A_i to A_j was traversed, and $w_{i,j,k}$ be the number of times the sequence A_i, A_j, A_k was traversed. In addition, we let $p_{i,j} = w_{i,j}/w_i$ be the first-order transition probability from A_i to A_j, and $p_{i,k\,j} = w_{i,k,j}/w_{i,k}$, be the second-order transition probability. Also, the accuracy threshold, γ, sets the highest admissible difference between a first-order and a second-order probability; a model is said to be *accurate* if there is no link that violates the constraint set by γ.

In the example given in Figure 1, the user's past navigation behaviour implies that $p_{1,23} = p_{1,24} = 0.5$. Therefore, for $\gamma = 0.1$, the state A_2 is not accurate, since $|p_{1,2\,3} - p_{2,3}| > 0.1$, and needs to be cloned. To clone state A_2, we let each in-link define a vector of second-order probabilities; each of the vector's components corresponds to an out-link from state A_2. In the example, state A_2 has three in-links and two out-links, inducing three vectors of second-order probabilities: for $i = \{3, 4\}$ we have $P_{1,2i} = \{0.5, 0.5\}$, $P_{5,2i} = \{0.2, 0.8\}$ and $P_{6,2i} = \{0.4, 0.6\}$.

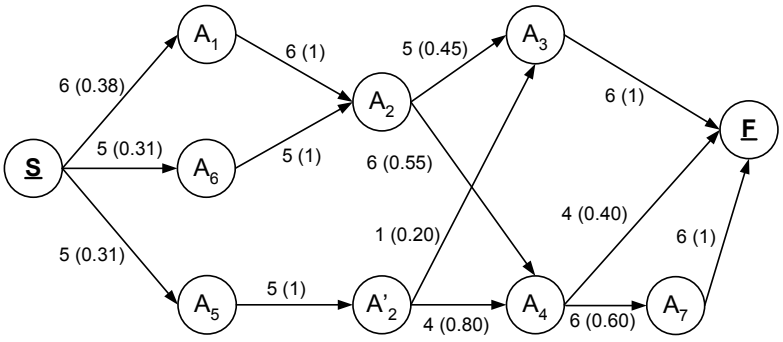

Fig. 2. The second-order HPG model obtained when applying the dynamic clustering method with $\gamma = 0.1$ to the first-order model given in Figure 1

The method applies a k-means clustering algorithm to the collection of second-order vectors, in order to identify groups of similar vectors with respect to γ. Figure 2 shows the result of applying the method to state A_2. Since $|p_{1,2\,i} - p_{6,2\,i}| < 0.1$, for $i = \{3, 4\}$, links from A_1 and A_6 are assigned to one clone, the link from A_5 is assigned to the other clone. The transition counts for the out-links are updated as follows: $w_{2,3} = w_{1,2,3} + w_{6,2,3}$, $w_{2,4} = w_{1,2,4} + w_{6,2,4}$, $w_{2',3} = w_{5,2,3}$, $w_{2',4} = w_{5,2,4}$. Note that, state A_4 is accurate, since all its in-links have an equivalent source state, and, moreover, every state having just one out-link is accurate by definition. Therefore, the model given in Figure 2 accurately represents every second-order transition probability.

3 A dynamic clustering method to model higher-order probabilities

We now extend the method presented in [4] to incorporate higher-order probabilities. In a second-order HPG model, the transition probabilities from a given state are considered to be accurate, if all in-links to it induce identical second-order probabilities. Similarly, in a third-order model every two-link path to a state must induce identical third-order probabilities. In general, to accurately model n-order probabilities each $(n-1)$-length path to a state must induce

identical n-order conditional probabilities. Estimates of the n-order conditional probabilities are obtained from the n-gram counts.

In the following, we let the length of a path be measured by the number of links it is composed of, and we call the length of the path from a state to the target state the *depth* of this state; $d = 0$ corresponds to the target state and $d = n - 1$ corresponds to the farthest state from the target when assessing the model for order n. We let $w_{1,\ldots,n}$ represent the n-gram counts, and $p_{i\ldots j,kt} = w_{i,\ldots,j,k,t}/w_{i,\ldots,j,k}$ represent the n-order conditional probability of going to state A_t given that the $(n-1)$-length path $A_i, \ldots, A_j, A_k$ was followed. Also, we let $\overrightarrow{l}$ represent a path and $p_{\overrightarrow{l},kt}$ the conditional probability of transition (A_k, A_t) given the path $\overrightarrow{l}$. Also, we let $\overrightarrow{l}_{[d]}$ be the state at depth d on $\overrightarrow{l}$ and $v_{\overrightarrow{l}}$ be the vector of n-order conditional probabilities given path $\overrightarrow{l}$. If a state y needs c_y clones, we let y_i, with $i = \{1, \ldots, c_y\}$, represent y and its $c_y - 1$ additional clones. Finally, we let $\overrightarrow{l}_c$ be the cluster to which path $\overrightarrow{l}$ was assigned.

For a state x, the n-order conditional probabilities are assessed in three steps:

(i) Apply a breath-first search procedure to induce the $(n-1)$-length in-paths to state x, estimate the corresponding n-order conditional probabilities and, for each path, $\overrightarrow{l}$, store the conditional probabilities in a vector $v_{\overrightarrow{l}}$ (the vector's dimension is given by the number of out-links from x). If the difference between a conditional probability and the corresponding transition probability is greater than γ, label the state as needing to be cloned.

(ii) If x needs cloning, apply the k-means algorithm to the probability vectors, $v_{\overrightarrow{l}}$. The number of clusters k is incremented until in the final solution, and in every cluster, the distance between each vector and its centroid is smaller than γ.

(iii) Identify states that need to be cloned to separate the paths to x. States included in paths to x are assessed in descending depth order from $d = n-1$ to $d = 0$. For depth d, we let a prefix of a path to x, whose last state is y, be named a y path-prefix to x. Thus, to separate paths to x, state y at depth, d, needs as many clones as the number of distinct path prefixes with the same length that are assigned to different clusters. The weights of the in and out-links of y and its clones are determined by the n-gram counts. After cloning y the in-paths to x need to be updated.

We now present an example of the method and a pseudo-code description. In particular, we evaluate the third-order probabilities for the model in Figure 2. The conditional probabilities induced by the paths to A_4 are: for $i = \{7, F\}$ we have $p_{12,4i} = \{0.33, 0.67\}$, $p_{62,4i} = \{0.67, 0.33\}$ and $p_{52,4i} = \{0.75, 0.25\}$. Thus, since these probabilities are not close to the corresponding second-order probabilities, A_4 is not third-order accurate for $\gamma = 0.1$. Table 1 gives the in-paths to A_4, the third-order conditional probabilities and the resulting clustering assignment. As result, state A_2 for $d = 1$ needs one clone, and for $d = 0$ state A_4 also needs one clone. Figure 3 gives the resulting third-order model.

In Figure 3, the path S, A_1, A_2, A_4 has probability estimate of $0.38 \cdot 1.00 \cdot 0.50 = 0.19$. It can be seen that in Figure 1, from a total of 16 sessions, 3

$d=2$	$d=1$	$d=0$	3rd order vectors		cluster
A_1	A_2	A_4	0.33	0.67	1
A_6	A_2	A_4	0.67	0.33	2
A_5	A'_2	A_4	0.75	0.25	2

Table 1. The paths to A_4, the third-order conditional probabilities and the resulting clustering assignment

begin with the 3-gram A_1, A_2, A_4 resulting in a probability estimate of 0.19. Also, according to the third-order model, path S, A_5, A_2, A_4, A_7 has probability $0.31 \cdot 1.00 \cdot 0.80 \cdot 0.71 = 0.18$. It can be seen that in the input data 3 sessions begin with A_5, A_2, A_4, A_7, resulting in a probability estimate of 0.19. In both cases the difference between the two estimates is below 0.1, which is the value specified for the accuracy probability threshold, γ.

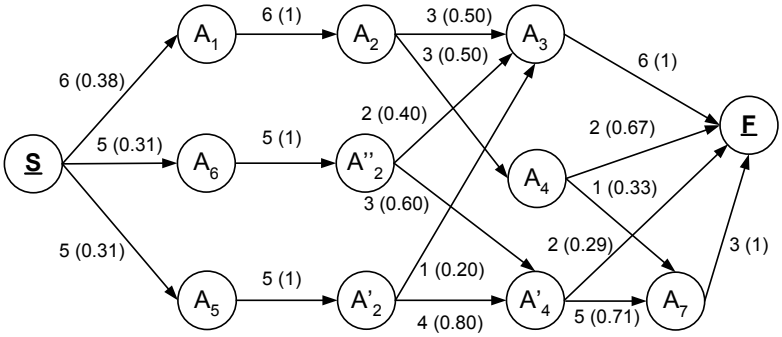

Fig. 3. The third-order model obtained when applying the dynamic clustering method with $\gamma = 0.1$ to the model given in Figure 2

Alternatively, the initial probability of a state can be estimated as $w_i / \sum_j w_j$ and every state has a link from S. In Figure 3 there is a total of 54 page views, and, for example, $p_{S,2} = 6/54 = 0.11$ and $p_{S,4'} = 7/54 = 0.13$. For the path A_2, A_4, A_7 the probability estimate is given by the sum of the probabilities of path A_2, A_4, A_7, path A'_2, A'_4, A_7 and path A''_2, A'_4, A_7, which is 0.12. In the input sessions, shown in Figure 1, we have a total of 54 3-gram counts (including 3-grams starting with S and ending with F) and the count of A_2, A_4, A_7 is 6, therefore, its estimate is $6/54 = 0.11$. For path A_1, A_2, A_3 the model gives 0.05, while the session analysis gives 0.05. Both cases are accurate with respect to $\gamma = 0.1$.

The pseudo-code description of the algorithm, which implements the method, is now given. We let n be the order with which to evaluate the model, $HPG_{(n-1)}$ be the previous order model, and $(n+1)$-grams be the n-gram counts of size $n+1$.

```
Algorithm (HPG_(n−1), n, γ, (n+1)-grams)
  begin:
    for each state x
      induce in-paths of length n − 1 to x
      for each in-path l⃗
        for each out-link i from x
          estimate p_{l⃗,xi} and store in v_{l⃗}
          if (|p_{l⃗,xi} − p_{x,i}| > γ) the state needs to be cloned
        end for
      end for
      if state needs to be cloned
        apply k-means to collection of vectors v_{l⃗}
        for depth d = (n − 1) to d = 0
          for each state y at depth d
            c_y = num. distinct path prefixes assigned to different clusters
            create c_y − 1 clones of state y
            for each in-path l⃗ to x
              if (l⃗_[d] = y and l⃗_c > 1) redirect link to corresponding clone
            end for
            for state y_i with i = {1, ..., c_y}
              for each in-link t to y_i
                for each out-link r from y_i
                  w_{t,y_i} = w_{t,y_i} + w_{t,y_i,r} , w_{y_i,r} = w_{y_i,r} + w_{t,y_i,r}
                end for
              end for
              remove out-links from y_i such that w_{y_i,r} = 0
            end for
            update ngram counts to take into account clones
          end for
          update in-paths with clone references
        end for
      end if
    end for
  end.
```

4 Experimental Evaluation

For the experimental evaluation we analysed three real world data sets. By using data from three different sources we aim to assess the characteristics of our model in a wide enough variety of scenarios. Our previous experience has shown that it is difficult to create random data sets that mirror the characteristics of real world data, and therefore looking at several data sets is necessary.

The first data set (CS) is from a university site, was made available by the authors of [15] and represents two weeks of usage data in 2002. The site was cookie based, page caching was prohibited and data was made available with the sessions identified. We split the data set into three subsets in order to enhance

analysis in a wider variety of scenarios. The second data set (MM) was obtained from the authors of [11] and corresponds to one month of usage from the Music Machines site (`machines.hyperreal.org`) in 1999 . The data was organised in sessions and caching was disabled during collection. We split the data set into four subsets, each corresponding to a week of usage. The third data set (LTM) represents forty days of usage from the London Transport Museum web site in 2003 (`www.ltmuseum.co.uk`). The data was obtained in a raw format. We filtered .gif and .jpg requests, and requests with an error status code. Sessions were defined as consecutive requests from a given IP address within a 30 minute time window and a maximum session length of 100 requests was set. We split this data set into four subsets, each corresponding to ten days of usage data.

Table 2 gives the summary characteristics for each data set; *ds* identifies the data set, *pg* gives the number of distinct pages visited, $\%1v$ and $\%\leq 2v$ indicate, respectively, the percentage of pages with just one visit and with two or less visits. Also, *aOL* gives the average number of out-links per state, *sOL* the standard deviation, *aIL* the average number of in-links per page and *sIL* the standard deviation. Finally, *ses* gives the number of sessions, *aSes* the average session length, *sSes* the standard deviation, and *req* the total number of requests. The variabilty on the number of states induced by the model for a given web site can be explained by the number of pages with less than one visit. Also, when the number of pages with few visits increases the average number of out-links and in-links decreases. The average session length is stable but the standard deviation shows that the MM data has a higher variability on the session length.

ds	*pg*	$\%1v$	$\%\leq 2v$	*aOL*	*sOL*	*aIL*	*sIL*	*ses*	*aSes*	*sSes*	*Req*
LTM_1	2998	0.62	0.68	4.5	9.6	4.4	11.6	9743	7.6	13.5	74441
LTM_2	1648	0.19	0.27	8.4	13.8	8.3	16.6	11070	7.4	13.2	82256
LTM_3	1610	0.27	0.37	7.8	12.8	7.7	15.0	9116	7.7	13.1	70558
LTM_4	1586	0.24	0.34	7.8	13.3	7.7	15.9	9965	7.8	13.4	78179
MM_1	8715	0.30	0.45	4.7	12.4	4.6	14.1	14734	6.4	37.8	94989
MM_2	5356	0.32	0.44	6.0	18.9	5.9	20.7	14770	6.1	14.7	90682
MM_3	5101	0.26	0.38	6.0	15.6	5.9	17.7	10924	6.7	35.2	73378
MM_4	6740	0.35	0.49	5.1	18.5	4.9	19.8	14080	6.3	23.8	88053
CS_1	3128	0.52	0.67	3.4	10.1	3.1	10.4	7000	4.8	6.5	33854
CS_2	3946	0.59	0.74	2.8	9.3	2.6	9.9	7000	5.0	8.4	34897
CS_3	5028	0.62	0.76	2.8	9.4	2.6	11.6	6950	5.5	12.8	38236

Table 2. Summary characteristics of the real data sets

The left-hand side of Figure 4, shows, for the three representative data sets, the variation of the model size with its order for $\gamma = 0$. (The data sets from each source reveal almost identical behaviour). For the MM_1 data set a large percentage of state cloning is performed for second and third-order probabilities which indicates that there is no significant difference between third-order probabilities and the corresponding higher-order probabilities, and that the MM site

only requires a short history when deciding which link to follow. The CS data set shows a slower increase in the model's size, and the model can be seen to reach close to full accuracy with respect to fourth-order probabilities. Finally, the LTM data set shows an increase in the number of states for up to the seventh-order probabilities, meaning that the choice of which link to follow is clearly influenced by a relatively long sequence of previously visited web pages.

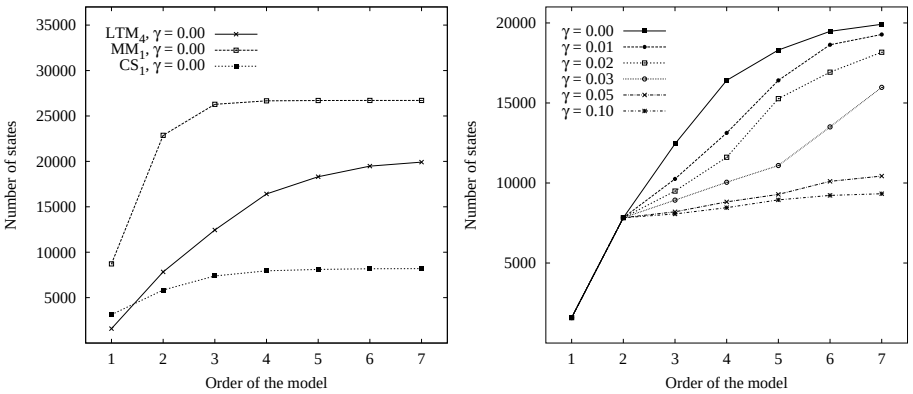

Fig. 4. The increase in model size with the model's order for $\gamma = 0$ and the increase in size for several values of the probability threshold, γ, with the LTM_4 data set

The right-hand side of Figure 4, shows the effect of γ, on the model size for the LTM_4 data set and the left-hand side of Figure 5, shows the effect for the CS_1 data set. In both cases it can be seen that by tuning the value of γ it is possible to control the increase on a models's number of states. For both data sets the difference in the number of states is not evident for second-order models. For third and higher-order models it is possible to reduce the number of states induced by the method by allowing some tolerance on representing the conditional probabilities (by setting $\gamma > 0$). Setting γ to a value greater than 0.1 results in almost no cloning for higher orders.

Figure 6 shows some statistics on the number of clones per state for the LTM_4 and CS_1 data sets, with $\gamma = 0.02$. The the average number of clones per state (avg) is higher for the LTM_4 data set than for the CS_1 data set, as expected by inspecting the left-side of Figure 4. The standard deviation (stdev) indicates a substantial variability in the number of clones per state, a fact that is supported by the maximum number of clones (max) and the indicated percentiles. For the LTM_4 data set 50% of the states were never cloned and 75% have at most six clones for the seventh order. In the CS_1 data set 75% of the states were never cloned and 90% of the states have at most seven clones for the seventh order. These results help to motivate our interest in the dynamic model, since while for some states the link choice of the corresponding page depends on the

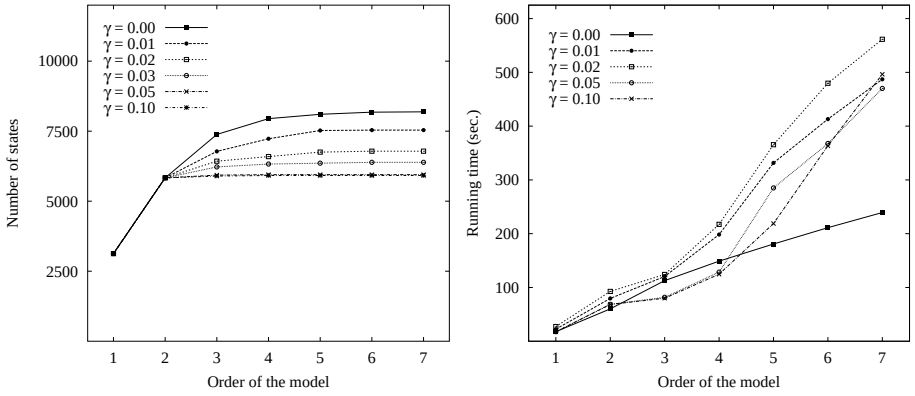

Fig. 5. The increase in model size with the model's order for several values of the probability threshold, γ, for the CS_1 data set and variation of the running time with the model's order for several values of the probability threshold for the LTM_4 data set

navigation history, for other states the link choice is completely independent of the navigation history.

	LTM_4 $\gamma = 0.02$					
	order					
	2	3	4	5	6	7
avg	3.94	4.99	6.32	8.63	9.67	10.46
stdev	10.86	15.29	24.85	40.88	47.20	50.69
max	205	307	683	989	1193	1265
75%	4.00	5.00	5.00	6.00	6.00	6.00
85%	10.00	11.25	13.00	14.00	16.25	18.00

	CS_1 $\gamma = 0.02$					
	order					
	2	3	4	5	6	7
avg	0.87	1.06	1.11	1.16	1.17	1.17
stdev	5.40	7.06	7.3	7.92	7.96	7.96
max	138	175	180	208	208	208
75%	0.00	0.00	0.00	0.00	0.00	0.00
95%	4.00	5.00	5.00	5.00	5.00	5.00

Fig. 6. Statistics on the number of clones per state with the model's order for the LTM_4 and CS_1 data set with $\gamma = 0.02$

The right-hand side of Figure 5, and the left-hand side of Figure 7, show our analysis of the running time of the algorithm for two representative data sets. We note that, while programming the method, we did not take particular care regarding the implementation efficiency. The method is close to linear time for $\gamma = 0$, since in such case no clustering is needed. For $\gamma > 0$ the k-means method is applied and we let k increase until a solution which meets the threshold criteria is obtained. For the reported experiments, we set k to vary according to the expression $k = ceiling(1.5k)$ in order to obtain a slow increment on its value in the first stages and a larger increase of the k value in the subsequent stages. Finally, the right-hand side of Figure 7, shows, for the LTM_1 data set, the increase in number of states with the model's order for three methods used to increase k, in the k-means method. The results show that for lower orders the number of states is not very sensitive to the method, and for orders higher

than five the faster methods over estimate the number of clusters but with the benefit of a faster running time (which is not shown in the plot).

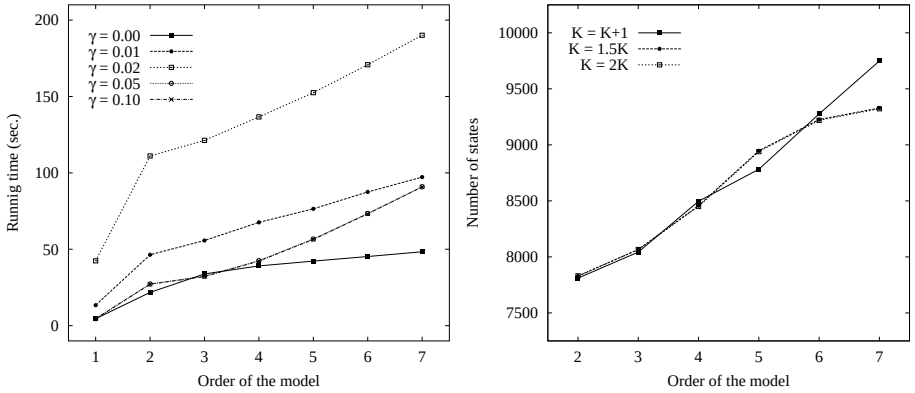

Fig. 7. The running time with the model's order for several values of γ for the CS_1 data set and the increase in model size with the model's order for three methods to set the number of clusters (k) for the LTM_1 data set

5 Concluding Remarks

We have proposed a generalisation of the HPG model by using a state cloning operation that is able to accurately model higher-order conditional probabilities. The resulting dynamic high-order Markov model is such that the probabilities of the out-links from a given state reflect the n-order conditional probabilities of the paths to the state. Thus, the model is able to capture a variable length history of pages, where different history lengths are needed to accurately model user navigation. In addition, the method makes use of a probability threshold together with a clustering technique that enables us to control the number of additional states induced by the method at the cost of some accuracy. Finally, the model maintains the fundamental properties of the HPG model, [1], providing a suitable platform for an algorithm that can mine navigation patterns, taking into account the order of page views.

We reported on experiments with three distinct real world data sets. From the results we can conclude that, for some web sites users navigate with only a short history of the pages previously visited (for example, the MM site) but in other sites users hold a longer history in their memory (for example, the LTM site). The results also suggest that, in a given site, different pages require different amounts of history in order to understand the possible options users have when deciding on which link to click on. This supports our interest in the proposed

dynamic method that models each state with the required history depth. The results indicate that the clustering method is interesting for large sites, where the number of states induced for high orderers, for $\gamma = 0$, becomes unmanageable.

In the short term we plan to conduct a study to analyse the semantics of the rules induced by different order probability models. We also plan to perform a statistical comparison of subsequent order probabilities aimed at determining if there is sufficient statistical evidence that the additional model complexity in moving to a higher order justifies the corresponding increase in the algorithm's complexity. It would also be interesting to be able to estimate the number of clusters necessary to achieve the required accuracy in order to speed up the method. Finally, a comparative study with tree-based models is also planned.

References

1. J. Borges. *A data mining model to capture user web navigation patterns.* PhD thesis, University College London, London University, 2000.
2. J. Borges and M. Levene. Data mining of user navigation patterns. In B. Masand and M. Spliliopoulou, editors, *Web Usage Analysis and User Profiling*, Lecture Notes in Artificial Intelligence (LNAI 1836), pages 92–111. Springer Verlag, 2000.
3. J. Borges and M. Levene. An average linear time algorithm for web usage mining. *Int. Jou. of Information Technology and Decision Making*, 3(2):307–319, June 2004.
4. J. Borges and M. Levene. A dynamic clustering-based markov model for web usage mining. cs.IR/0406032, 2004.
5. Eugene Charniak. *Statistical Language Learning.* The MIT Press, 1996.
6. X. Chen and X. Zhang. A popularity-based prediction model for web prefetching. *Computer*, 36(3):63–70, March 2003.
7. M. Deshpande and G. Karypis. Selective markov models for predicting web page accesses. *ACM Transactions on Internet Technology*, 4(2):163–184, 2004.
8. X. Dongshan and S. Junyi. A new markov model for web access prediction. *Computing in Science and Engineering*, 4(6):34–39, November/December 2002.
9. M. Levene and G. Loizou. Computing the entropy of user navigation in the web. *Int. Journal of Information Technology and Decision Making*, 2:459–476, 2003.
10. B. Mobasher. Web usage mining and personalization. In Munindar P. Singh, editor, *Practical Handbook of Internet Computing.* Chapman Hall & CRC Press, 2004.
11. Mike Perkowitz and Oren Etzioni. Towards adaptive web sites: conceptual framework and case study. *Artificial Intelligence*, 118(2000):245–275, 2000.
12. J. Pitkow and P. Pirolli. Mining longest repeating subsequences to predict world wide web surfing. In *Proc. of the 2nd Usenix Symposium on Internet Technologies and Systems*, pages 139–150, Colorado, USA, October 1999.
13. Ramesh R. Sarukkai. Link prediction and path analysis using markov chains. *Computer Networks*, 33(1-6):377–386, June 2000.
14. S. Schechter, M. Krishnan, and M. D. Smith. Using path profiles to predict http requests. *Computer Networks and ISDN Systems*, 30:457–467, 1998.
15. M. Spiliopoulou, B. Mobasher, B. Berendt, and M. Nakagawa. A framework for the evaluation of session reconstruction heuristics in web usage analysis. *IN-FORMS Journal on Computing*, (15):171–190, 2003.
16. J. Zhu, J. Hong, and J. G. Hughes. Using markov models for web site link prediction. In *Proc. of the 13th ACM Conf. on Hypertext and Hypermedia*, pages 169–170, June 2002.