Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 00-056

Selective Markov Models for Predicting Web-Page Accesses

Mukund Deshpande and George Karypis

October 30, 2000

Selective Markov Models for Predicting Web-Page Accesses*

Mukund Deshpande and George Karypis

University of Minnesota, Department of Computer Science/Army HPC Research Center
Minneapolis, MN 55455
Technical Report #00-056

{deshpand,karypis}@cs.umn.edu

Last updated on October 30, 2000 at 10:52am

Abstract

The problem of predicting a user's behavior on a web-site has gained importance due to the rapid growth of the world-wide-web and the need to personalize and influence a user's browsing experience. Markov models and their variations have been found well suited for addressing this problem. Of the different variations or Markov models it is generally found that higher-order Markov models display high predictive accuracies. However higher order models are also extremely complicated due to their large number of states that increases their space and runtime requirements. In this paper we present different techniques for intelligently selecting parts of different order Markov models so that the resulting model has a reduced state complexity and improved prediction accuracy. We have tested our models on various datasets and have found that their performance is consistently superior to that obtained by higher-order Markov models.

Keywords: world wide web, web mining, Markov models, predicting user behavior.

*This work was supported by NSF CCR-9972519, EIA-9986042, ACI-9982274, by Army Research Office contract DA/DAAG55-98-1-0441, by the DOE ASCI program, and by Army High Performance Computing Research Center contract number DAAH04-95-C-0008. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute. Related papers are available via WWW at URL: **http://www.cs.umn.edu/~karypis**

1 Introduction

In recent years, the problem of modeling and predicting a user's surfing behavior on a web-site has attracted a lot of research interest as it can be used to improve the web cache performance [SKS98, Bes95, PM96], recommend related pages [DH99, PPR96], improve search engines [BP98], understand and influence buying patterns [CPM+98], and personalize the browsing experience [PP99]. The significance of this problem is evident by the fact that at the SIGKDD 2000 Conference [KB00] the problem of predicting and understanding a user's browsing and purchasing behavior was the topic of the KDDCup 2000 Competition.

Markov models [Pap91] have been used for studying and understanding stochastic processes, and were shown to be well-suited for modeling and predicting a user's browsing behavior on a web-site. In general, the input for these problems is the sequence of web-pages that were accessed by a user and the goal is to build Markov models that can be used to model and predict the web-page that the user will most likely access next. Padbanabham and Mogul [PM96] use N-hop Markov models for improving pre-fetching strategies for web caches. Pitkow *et al*. [PP99] proposed a longest subsequence models as an alternative to the Markov model. Sarukkai [Sar00] use Markov models for predicting the next page accessed by the user. Cadez *et al*. [CHM+00] use Markov models for classifying browsing sessions into different categories.

In many applications, first-order Markov models are not very accurate in predicting the user's browsing behavior, since these models do not look far into the past to correctly discriminate the different observed patterns. As a result, higher-order models are often used. Unfortunately, these higher-order models have a number of limitations associated with high state-space complexity, reduced coverage, and sometimes even worse prediction accuracy. One simple method to overcome some of these problems is to train varying order Markov models and use all of them during the prediction phase, as is done in the All-K^{th}-Order Markov model proposed in [PP99]. However, this approach further exacerbates the problem of state-space complexity. An alternate approach proposed by Pitkow *et al*. [PP99] is to identify patterns of frequent accesses, which they call longest repeating subsequences, and then use this set of sequences for prediction. However, even though this approach is able to reduce the state-space complexity by up to an order of magnitude, it also reduces the prediction accuracy of the resulting model.

In this paper we present techniques for intelligently combining different order Markov models so that, the resulting model has a low state complexity, improved prediction accuracy, and retains the coverage of the All-K^{th}-Order Markov models. The key idea behind our work is that many of the states of the different order Markov models can be eliminated without affecting the performance of the overall scheme. In particular, we present three schemes for pruning the states of the All-K^{th}-Order Markov model, called (i) support pruning (ii) confidence pruning (iii) error pruning. Our experiments on a variety of data sets have shown that the proposed pruning schemes consistently outperform the All-K^{th}-Order Markov model and other single-order Markov models, both in term of state complexity as well as improved prediction accuracy. For many problems, our schemes prune up to 90% of the states from the All-K^{th}-Order Markov model, and improve the accuracy by up to 11%.

Even though our algorithms were developed in the context of web-usage data, we have successfully used these techniques for prediction in different applications, as well. For example, these models were used to predict the next command typed by the user in word processor based on his/her past sequence of commands or for predicting the alarm state of telephone switches based on its past states. These applications will be discussed in detail in Section 4.1.

The rest of this paper is organized as follows. Section 2 presents an overview of Markov models followed by a brief overview of the problem of predicting a user's browsing behavior. Section 3 presents a detailed

description of our selective Markov models. Section 4 provides a detailed experimental evaluation of our algorithms on a variety of data sets. Finally, Section 5 offers some concluding remarks.

2 Markov Models for Predicting User's Actions

As discussed in the introduction, techniques derived from Markov models have been extensively used for predicting the action a user will take next given the sequence of actions he or she has already performed. For this type of problems, Markov models are represented by three parameters $< A, S, T >$, where A is the set of all possible *actions* that can be performed by the user; S is the set of all possible states for which the Markov model is built; and T is a $|S| \times |A|$ *Transition Probability Matrix* (TPM), where each entry t_{ij} corresponds to the probability of performing the action j when the process is in state i.

The state-space of the Markov model depends on the number of previous actions used in predicting the next action. The simplest Markov model predicts the next action by only looking at the last action performed by the user. In this model, also known as the *first-order Markov model*, each action that can be performed by a user corresponds to a state in the model. A somewhat more complicated model computes the predictions by looking at the last two actions performed by the user. This is called the *second-order Markov model*, and its states correspond to all possible pairs of actions that can be performed in sequence. This approach is generalized to the K^{th}*-order Markov model*, which computes the predictions by looking at the last K actions performed by the user, leading to a state-space that contains all possible sequences of K actions.

For example, consider the problem of predicting the next page accessed by a user on a web site. The input data for building Markov models consists of *web-sessions*, where each session consists of the sequence of the pages accessed by the user during his/her visit to the site. In this problem, the actions for the Markov model correspond to the different pages in the web site, and the states correspond to all consecutive pages of length K that were observed in the different sessions. In the case of first-order models, the states will correspond to single pages, in the case of second-order models, the states will correspond to all pairs of consecutive pages, and so on.

Once the states of the Markov model have been identified, the transition probability matrix can then be computed. There are many ways in which the TPM can be built. The most commonly used approach is to use a *training* set of action-sequences and estimate each t_{ji} entry based on the frequency of the event that action a_i follows the state s_j. For example consider the *web-session* $WS_2(\{P_3, P_5, P_2, P_1, P_4\})$ shown in Figure 1. If we are using *first-order Markov model* then each state is made up of a single page, so the first page P_3 corresponds to the state s_3. Since page p_5 follows the state s_3 the entry t_{35} in the TPM will be updated. Similarly, the next state will be s_5 and the entry t_{52} will be updated in the TPM. In the case of higher-order model each state will be made up of more than one actions, so for a second-order model the first state for the *web-session* WS_2 consists of pages $\{P_3, P_5\}$ and since the page P_2 follows the state $\{P_3, P_5\}$ in the web-session the TPM entry corresponding to the state $\{P_3, P_5\}$ and page P_2 will be updated.

Once the transition probability matrix is built making prediction for web sessions is straight forward. For example, consider a user that has accessed pages P_1, P_5, P_4. If we want to predict the page that will be accessed by the user next, using a first-order model, we will first identify the state s_4 that is associated with page P_4 and look up the TPM to find the page p_i that has the highest probability and predict it. In the case of our example the prediction would be page P_5.

Web Sessions:
$WS_1 : \{P_3, P_2, P_1\}$
$WS_2 : \{P_3, P_5, P_2, P_1, P_4\}$
$WS_3 : \{P_4, P_5, P_2, P_1, P_5, P_4\}$
$WS_4 : \{P_3, P_4, P_5, P_2, P_1\}$
$WS_5 : \{P_1, P_4, P_2, P_5, P_4\}$

1st Order	P_1	P_2	P_3	P_4	P_5
$s_1 = \{P_1\}$	0	0	0	2	1
$s_2 = \{P_2\}$	4	0	0	0	1
$s_3 = \{P_3\}$	0	1	0	1	1
$s_4 = \{P_4\}$	0	1	0	0	2
$s_5 = \{P_5\}$	0	3	0	2	0

2nd Order	P_1	P_2	P_3	P_4	P_5	*2nd Order*	P_1	P_2	P_3	P_4	P_5
$\{P_1, P_4\}$	0	1	0	0	0	$\{P_3, P_5\}$	0	1	0	0	0
$\{P_1, P_5\}$	0	0	0	1	0	$\{P_4, P_2\}$	0	0	0	0	1
$\{P_2, P_1\}$	0	0	0	1	1	$\{P_4, P_5\}$	0	2	0	0	0
$\{P_2, P_5\}$	0	0	0	1	0	$\{P_5, P_2\}$	3	0	0	0	0
$\{P_3, P_2\}$	1	0	0	0	0	$\{P_3, P_4\}$	0	0	0	0	1

Figure 1: Sample web sessions with the corresponding 1st & 2nd order Transition Probability Matrices.

2.1 Limitations of Markov Models

In many applications, first-order Markov models are not successful in predicting the next action to be taken by the user. This is because these models do not look far into the past to correctly discriminate the different behavioral modes of the different users. As a result, in order to obtained good predictions higher-order models must be used (*e.g.*, second and third) . Unfortunately, these higher-order models have a number of limitations associated with (i) high state-space complexity, (ii) reduced coverage, and (iii) sometimes even worse prediction accuracy.

The number of states used in these models tend to rise exponentially as the order of the model increases. This is because, the states of higher-order models are nothing but different combinations of the actions observed in the dataset. This dramatic increase in the number of states can significantly limit the applicability of Markov models for applications in which fast predictions are critical for real-time performance or in applications in which the memory constraints are tight (*e.g.* embedded models for web-access caching [PM96]). Furthermore, there may be many examples in the test set that do not have corresponding states in higher-order Markov model; thus, reducing their coverage. In such scenarios higher-order models must make a default prediction that can lead to lower accuracies.

To better understand these shortcomings we conducted an experiment on one of our web datasets. We compared various higher-order Markov models starting from the first all the way to the fifth order Markov model. If a higher-order model was unable to make a prediction for a particular web-session, it was ignored from accuracy calculations (*i.e.*, the accuracy is calculated only on those examples on which the prediction was made). The results are plotted in Figure 2, and we can see that as the order of the model increases the accuracy increases accompanied by a decrease in the coverage. At the same time, as the order of the model increases, the number of states used for the model also increase dramatically.

One simple method to overcome the problem of low coverage is to train varying order Markov models and then combine them for prediction [PP99]. In this scheme, for each test instance, the highest-order Markov model that covers the instance is used for prediction. For example, if we build the first, second, and third-order Markov models, then given a test instance, we first try to make a prediction using the third-order model. If this model does not contain the corresponding state, then we try to make a prediction using the second-order model, and so on. This scheme is called the ***All-K^{th}-Order Markov model*** [PP99]. Note that even though the All-K^{th}-Order Markov model solves the problem of low coverage it exacerbates the

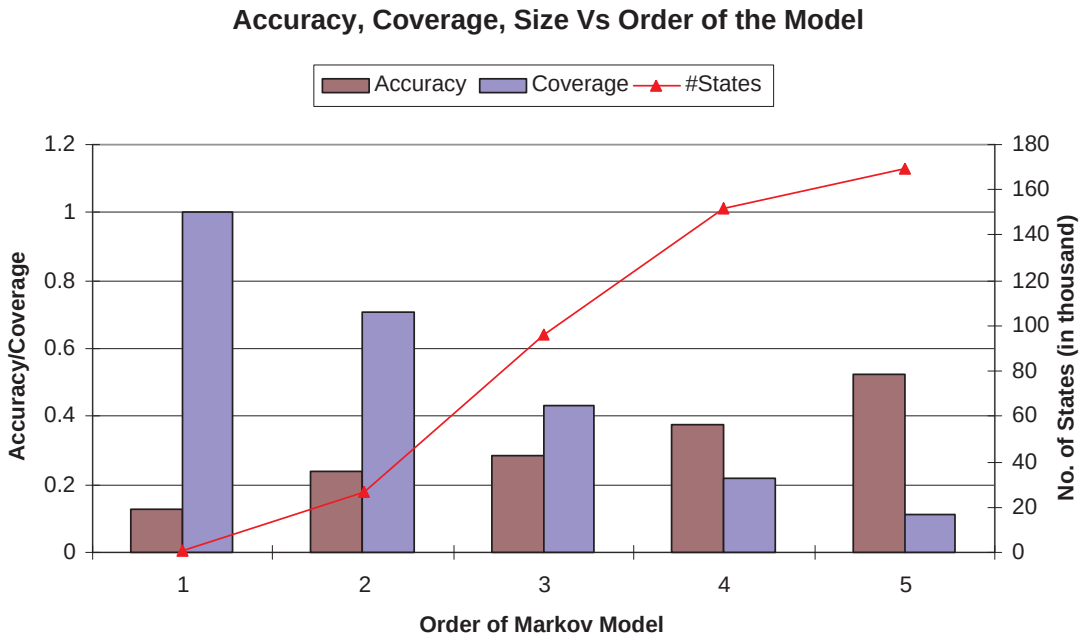

Figure 2: Plot comparing accuracy, coverage and model size with the order of Markov model.

problem of model size, as the states of all the different order models are part of the model.

Finally, as the number of states in higher-order Markov models increase the number of training-set instances used to compute the state-action transition probabilities for each of the states tends to decrease. A consequence of this reduction in the *support* of individual states, is that some of the estimated state-action transition probabilities are not accurate. As a result, the overall accuracy achieved by higher-order Markov models can sometimes be lower that achieved by the corresponding lower-order Markov models.

3 Selective Markov Models

As discussed in the previous section, All-K^{th}-Order Markov model holds the promise of achieving higher prediction accuracies and improved coverage than any single-order Markov model, at the expense of a dramatic increase in the state-space complexity. This led us to develop techniques for intelligently combining different order Markov models so that the resulting model has low state complexity, improved prediction accuracy, and retains the coverage of the All-K^{th}-Order Markov model.

Our schemes were motivated by the observation that given a sequence of actions for which we need to predict the next most probable action, there are multiple states in the All-K^{th}-Order Markov model that can be used to perform that prediction. In fact, there can be as many states as the number of the different order Markov models used to form the All-K^{th}-Order Markov model. Now, depending on the particular set of states involved, each of them can have different prediction accuracies. Based on this observation, we can then start from the All-K^{th}-Order Markov model and eliminate many of its states that are expected to have low prediction accuracy. This will allow us to reduce the overall state complexity without affecting the performance of the overall scheme.

The starting point for all of our algorithms is the All-K^{th}-Order Markov model obtained by building a

sequence of increasing order Markov models. However, instead of using this model for prediction, we use a number of techniques to eliminate certain states across the different order Markov models. The set of states that survive this step, then become the final model used for prediction. The goals of this *pruning* step is primarily to reduce the state complexity and secondarily improve the prediction accuracy of the resulting model. We will refer to these models as ***selective Markov models***.

The key step in our algorithm is the scheme used to determine to potential accuracy of a particular state. In the rest of this section we present three different schemes with an increasing level of complexity. The first scheme simply eliminates the states that have very low support. The second scheme uses statistical techniques to identify states for which the transition probabilities to the two most prominent actions are not statistically significant. Finally, the third scheme uses an error-based pruning approach to eliminate states with low prediction accuracy.

3.1 Support-Pruned Markov Model

The ***support-pruned Markov model*** (SPMM) is based on the observation that states that have low support in the training set tend to also have low prediction accuracies. Consequently, these low support states can be eliminated without affecting the overall accuracy as well as coverage of the resulting model. The amount of pruning in the SPMM scheme is controlled by the parameter ϕ referred to as the *frequency threshold*. In particular, SPMM eliminates all the states of the different order Markov models that are supported by fewer than ϕ training-set instances.

There are a number of observations to be made about the SPMM scheme. First, the same frequency threshold is used for all the models regardless of their order. Second, this pruning policy is more likely to prune higher-order states as higher order states have less support; thus dramatically reducing the state-space complexity of the resulting scheme. Third, the frequency threshold parameter ϕ, specifies the actual number of training-set instances that must be supported by each state and not the fraction of training-set instances as it is often done in the context of association rule discovery [AIS93]. This is done primarily for the following two reasons: (i) the trust-worthiness of the estimated transition probabilities of a particular state depend on the actual number of training-set instances and not on the relative number; (ii) the total number of training-set instances is in general exponential on the order of the Markov model, thus the same fractional pruning threshold will have a completely different meaning for the different order Markov models.

3.2 Confidence-Pruned Markov Model

One of the limitations of the SPMM scheme is that it does not capture all the parameters that influence the accuracy of the state. In particular the probability distribution of outgoing actions from a state is completely ignored. For example, consider a Markov state which has two outgoing actions/branches, such that one of them is substantially more probable than the other. Even if the overall support of this state is somewhat low, the predictions computed by this state will be quite reliable because of the clear difference in the outgoing probabilities. On the other hand, if the outgoing probabilities in the above example are very close to each other, then in order for that difference to be reliable, they must be based on a large number of training instances. Ideally, we would like the pruning scheme to not only consider the support of the state but also weigh the probability distribution of the outgoing actions before making its pruning decisions.

This observation led to us to develop the ***confidence-pruned Markov model*** (CPMM) scheme. CPMM uses statistical techniques to determine for each state, if the probability of the most frequently taken action is significantly different from the probabilities of the other actions that can be performed from this state. If the

probability differences are not significant, then this state is unlikely to give high accuracy and it is pruned. In contrast, if the probability differences are significant the state is retained.

The CPMM scheme determines if the most probable action is significantly different than the second most probable action by computing the $100(1-\alpha)$ percent confidence interval around the most probable action and checking if the probability of the second action falls within that interval. If this is true, then the state is pruned, otherwise it is retained. If $\hat{p}$ is the probability of the most probable action, then its $100(1-\alpha)$ percent confidence interval is given by

$$\hat{p} - z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \leq p \leq \hat{p} + z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}, \tag{1}$$

where $z_{\alpha/2}$ is the upper $\alpha/2$ percentage point of the standard normal distribution and n is the frequency of the Markov State [MR94].

The degree of pruning in CPMM is controlled by α (confidence coefficient). As the value of α decreases the size of the confidence interval increases, resulting in more pruning. Also note, that if a state has a large number of examples associated with it, then Equation 1 will compute a tighter confidence interval. As a result, even if the difference in the probabilities between the two most probable actions is relatively small, the state will most likely be retained. As our earlier discussion indicated this feature is desirable.

3.3 Error-Pruned Markov Model

In the previous schemes we used either the support of a state or the probability distribution of its outgoing branches to gauge the potential error associated with it. However, the error of each state can be also automatically estimated and used to decide whether or not to prune a particular state. A widely used approach to estimate the error associated with each state is to perform a *validation* step. During the validation step, the entire model is tested using part of the training set (*validation* set) that was not used during the model building phase. Since we know the actual actions performed by the sequences in the validation set, we can easily determine the error-rates and use them for pruning.

This led us to develop the ***error pruned Markov model*** (EPMM) scheme. Specifically, we have developed two different error-based pruning strategies that use a different definition as to what constitutes the error-rate of a Markov state. We will refer to these schemes as *overall error pruning* and *individual error pruning*.

The overall error pruning scheme works as follows. First, for each sequence in the validation set we use each one of the K single-order Markov models to make a prediction. For each prediction we record whether that prediction was correct or not. Once all the sequences in the validation set have been predicted, we use these prediction statistics to calculate the error-rate of each state. Next, for each state of the highest-order Markov model we identify the set of states in the lower-order models that are its proper subsets. For example, if the highest-order state corresponds to the action-sequence $\{a_5, a_3, a_6, a_7\}$, then the lower-order states that are identified are $\{a_3, a_6, a_7\}$ (third-order), $\{a_6, a_7\}$ (second-order) and $\{a_7\}$ (first-order). Now if the error-rate of the highest-order state is higher than any of its subset lower-order sates, it is pruned. The same procedure of identifying the subset states and comparing their error-rates is repeated for all the states in the lower-order Markov models as well, except the first-order Markov model. The states from the first-order Markov model are never pruned so as not to reduce the coverage of the resulting model.

In the second scheme we first iterate over all the higher-order states, and for each of them we find its subset states (as described in the previous scheme). Then, we identify all the examples in the validation set that can be predicted using the higher-order state (*i.e.*, the validation examples which have a sequence of

actions corresponding to the higher-order state). This set of examples are then predicted by the higher-order state and its subset states and the error-rates on these examples for each one of the states is computed. If the error-rate of the higher-order state is greater than any of its subset states, the higher-order Markov state is pruned. The same procedure is repeated for all the lower-order Markov models except the first-order Markov model.

Though both schemes follow a similar procedure of locating subset states and pruning the ones having high error-rates, they differ on how the error-rates for each state is computed. In the first scheme, every lower-order Markov state has a single error-rate value that is computed over the entire validation set. In the second scheme, each of the lower-order Markov states will have many error-rate values as it will be validated against a different set of examples for each one of its superset higher-order states.

4 Experimental Results

We experimentally evaluated the performance of the proposed Selective Markov models on a variety of data sets. In the rest of this section we briefly describe these data sets, our experimental methodology, and present the results obtained by our schemes.

4.1 Datasets

We have evaluated the performance of the proposed Selective Markov models schemes on four datasets, whose characteristics are shown in the Table 1.

Dataset	*# Sessions*	*Avg. Ses. Length*	*# of unique actions*
EC1	234,954	3.18967	874
EC2	144,367	7.93933	6,554
OWL	22,884	3.26735	160
TC1	142,983	2.67119	151

Table 1: Preliminary dataset statistics.

- **ECommerce Web Logs:** We used the web server logs from two large E-Commerce companies for our analysis. These logs were first cleaned using the WebSIFT system [CTS99] and then broken down into series of sessions, each session corresponds to the sequence of web pages accessed by the user during his/her visit to the site. Note that the session contains only the accesses to web pages and accesses to images are ignored. In our model each session corresponds to an example in the train/test set and the page accessed by the user is considered as an action. These two datasets will be referred as **EC1** and **EC2**. From Table 1 we can see that EC1 has a total of 234, 954 sessions, an average length of 3.18 pages/sessions and a total of 874 unique web-pages. Similarly, EC2 has 144, 367 sessions of average length 7.93 pages/sessions and about 6, 554 unique web-pages.

- **OWL Dataset:** This dataset contains the log of editing commands typed by different users in Microsoft Word over a period of 2 years [Lin00]. The goal of the model built on this dataset was to predict the next command typed by the user based on the user's past sequence of commands. Such predictive system could be used to make online recommendations to the user about commands which could be typed next. This dataset will be referred as **OWL**. In this dataset, each session corresponds to the sequence of commands typed by a user on a particular document in one sitting. The different commands typed by the users constitute as the actions of the Markov model.

- **Telephone Switch Dataset:** This dataset was obtained from a large telecommunications company which maintains nationwide telephone networks. The dataset contains the log of different alarms generated by a telephone switch over a period of one month. Each session in this dataset corresponds to the sequence of alarms given out by the switch that are related to the same problem. For this dataset the alarm generated by the switch is considered as an action. This dataset will be referred as **TC1**.

4.2 Experimental Design & Metrics

To make the evaluation of different schemes manageable, we limit ourselves to the problem of predicting just the last action of a test example/session. For evaluation purposes we divide the complete dataset into training set and test set. Depending on the model, the training set may further be divided into a validation set. During the testing step the model is given a *trimmed session* for prediction *i.e.*, the last part of the session is removed. The prediction made by the model is then compared with the removed part of the session to compute model's accuracy. In some cases, Markov model based schemes are unable to make a prediction for a session in the test set. This could be either because the length of test session is less than the order of the model or the model has not seen a similar session in the training step. In our evaluation scheme we will only consider sessions which are longer than the highest-order of the model and if a model is unable to predict for these long sessions, it is considered as a wrong prediction. In all of our experiments, for both the All-K^{th}-Order Markov model and selective Markov model schemes we combined first-, second-, and third-order Markov models, *i.e.*, $K = 3$.

The overall performance of the various Markov model-based algorithms were evaluated in terms of their accuracy and their model size. The *accuracy* of a model is defined as the number of correct predictions divided by the total number of predictions, and the *model size* is defined as the total number of states in the model. Accuracy measures the predictive ability of the model, whereas the model size gives us an insight into the memory and time requirements of the model.

4.3 Results for Support Pruned Markov Model

The goal of our first set of experiments was to study the effect of the frequency threshold (ϕ) parameter on the performance of the support-pruned Markov model. To achieve this we performed an experiment in which we used different values for ϕ ranging from 0 (no pruning) up to 24 in increments of two, and measured both the accuracy as well as the number of states in the resulting Markov model. These results are shown in Table 2 for the four different data sets in our experimental testbed.

To better visualize the results we plotted the accuracy obtained by the SPMM scheme for different values of ϕ over that obtained when $\phi = 0$ (*i.e.*, the original All-K^{th}-Order Markov model). These plots are shown in Figure 3. A number of interesting observations can be made by looking at Figure 3. First, for all four data sets we can see that in the beginning the increase in the frequency threshold is accompanied by an increase in the accuracy for the SPMM. In particular, accuracy improvements for the EC1 and OWL datasets are quite substantial. Second, as we continue to further increase the frequency threshold the accuracy achieved on the different datasets starts to flatten out or even decrease. For *EC1* and *EC2* datasets the overall accuracy remains about the same, whereas there is sharp drop for both *OWL* and the *TC1* datasets. The decrease in the overall accuracy for increasing values of the ϕ is because some useful Markov states are being pruned from the model; thus, affecting its overall accuracy. Third, each one of the four datasets achieve their maximum accuracy levels at different values of the frequency threshold, indicating that the *optimal* value of ϕ is dataset dependent. Fourth, the overall accuracy tends to change smoothly with ϕ; thus, making it possible to use a

Freq.	*EC1*		*EC2*		*OWL*		*TC1*	
Thr.	*Accuracy*	*# states*	*Accuracy*	*# states*	*Accuracy*	*# states*	*Accuracy*	*# states*
0	0.274301	124125	0.480756	295079	0.427862	6892	0.765419	4406
2	0.283366	27302	0.487485	49348	0.435307	1811	0.766087	1631
4	0.289381	16197	0.489525	28138	0.442396	1110	0.766422	1140
6	0.291637	11748	0.491019	20082	0.445587	820	0.766422	866
8	0.293527	9323	0.491687	15798	0.443283	676	0.763413	738
10	0.294442	7735	0.492306	13042	0.443814	570	0.761909	661
12	0.295112	6625	0.492415	11189	0.443283	490	0.762577	591
14	0.2956	5830	0.492634	9799	0.442928	435	0.762494	530
16	0.295539	5253	0.492646	8649	0.439206	395	0.762828	478
18	0.295702	4716	0.492525	7804	0.438143	364	0.763413	448
20	0.295905	4349	0.492586	7086	0.436725	341	0.762159	412
22	0.295844	4020	0.492622	6499	0.434066	316	0.76241	385
24	0.29623	3733	0.492658	5994	0.432648	303	0.762159	364

Table 2: The accuracy and the model size of SPMM for different values of frequency threshold(ϕ).

validation set approach to estimate its optimal value for each dataset.

The effect of the frequency threshold is very pronounced on the number of states in the Markov model. The size of the model drops drastically as the frequency threshold increases. For example in the case of the *EC2* dataset the number of states in the SPMM scheme with highest accuracy is almost 2% of the number of states of the All-K^{th}-Order Markov model. The dramatic reduction in the number of states and the accompanied improvement in accuracy are due to the fact that there are many low-support states that tend to be noise and outliers in the training set and their lower-order states (that in general have higher support) are more reliable in making predictions.

4.4 Results for Confidence Pruned Markov Model

To study the effect of the confidence coefficient (α) used by the CPMM scheme on the accuracy and state space complexity of the resulting model, we performed a sequence of experiments in which we varied α from 0.45 to 0.01 to obtain 55% to 99% confidence intervals. The accuracy values and the model size for different values of the percentage confidence interval are shown in Table 3.

% Conf.	*EC1*		*EC2*		*OWL*		*TC1*	
Inter.	*Accuracy*	*# states*	*Accuracy*	*# states*	*Accuracy*	*# states*	*Accuracy*	*# states*
55	0.295966	16992	0.492221	32093	0.441687	1468	0.770433	1464
60	0.297104	15736	0.492221	29984	0.441156	1408	0.770266	1415
65	0.297815	13970	0.49244	26863	0.441687	1325	0.771686	1351
70	0.2981	12417	0.492743	23788	0.442219	1199	0.772773	1272
75	0.297978	12118	0.492549	23267	0.443105	1181	0.770349	1260
80	0.298161	9656	0.492586	18358	0.439738	975	0.770851	1118
85	0.297754	9101	0.492865	17425	0.439383	942	0.769931	1088
90	0.297206	8164	0.492913	15803	0.437256	889	0.770182	1037
95	0.297002	7202	0.49312	14121	0.436016	802	0.768427	969
99	0.296271	5891	0.492962	11492	0.428749	690	0.764249	836

Table 3: The accuracy & model size of CPMM for different values of confidence interval.

To better visualize the effect of the confidence interval's length, we plotted the accuracy improvements achieved by CPMM over the accuracy of the All-K^{th}-Order Markov model for the different confidence intervals. These results are shown in Figure 4. As we can see from this figure, the length of the confidence interval has a similar effect on the accuracy as that of the frequency threshold parameter used in SPMM.

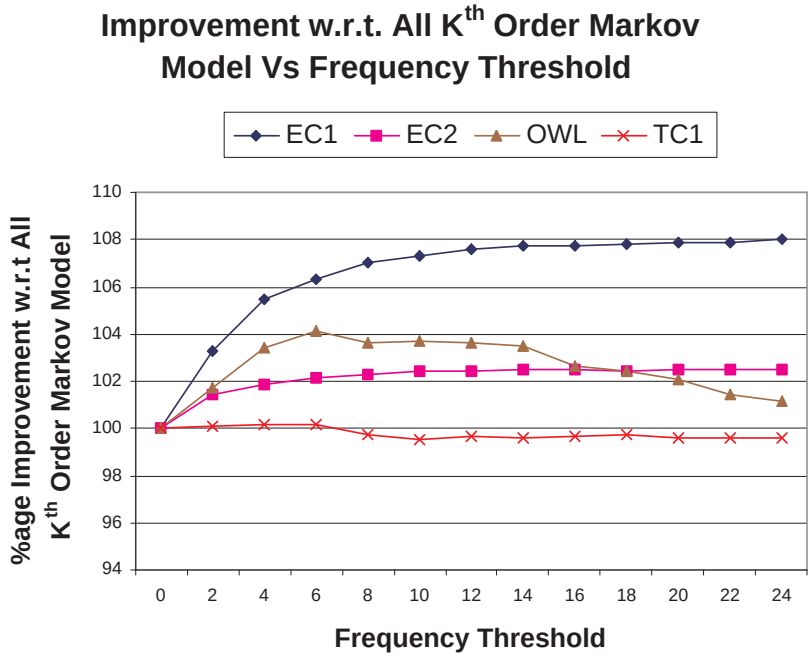

Figure 3: The accuracy obtained by SPMM for different values of ϕ relative to that obtained by All-K^{th}-Order Markov model.

Initially, the accuracy improves as we increase the length of the confidence interval; however, after a certain point, the accuracy starts to decrease. The reason for this performance degradation, is due to the fact that CPMM prunes a larger fraction of the states as the length of the confidence interval increases. Consequently, some reasonably high accuracy states are getting pruned as well.

Looking at the effect of the confidence interval's length on the model size we can see that as the length of the confidence interval increases, the number of states decreases. Note that the model size reduction achieved by CPMM is somewhat lower than that achieved by SPMM. We believe that this is because the CPMM scheme tends to retain some of the states that would have been pruned by SPMM, because their most probable outgoing actions are sufficiently more frequent than the rest.

4.5 Results for Error Pruned Markov Model

In this section we compare the performance of the overall and individual error pruning schemes discussed in Section 3.3. The accuracy and model size achieved by these schemes on the four test datasets are shown Table 4. The rows labeled "O. EPPM" and "I. EPPM" correspond to the overall and individual error-pruned schemes, respectively.

Model	*EC1*		*EC2*		*OWL*		*TC1*	
	Accuracy	*# states*	*Accuracy*	*# states*	*Accuracy*	*# states*	*Accuracy*	*# states*
O. EPPM	0.31255	10218	0.502399	24207	0.469337	1332	0.779208	868
I. EPPM	0.304461	21619	0.499666	30373	0.468451	1795	0.790908	1162

Table 4: The accuracy & model size of error pruned Markov models for different datasets.

From the results in this table we can see that both error-based pruning schemes produce very similar results in terms of model accuracy, with the overall error-pruned scheme performing slightly better. However,

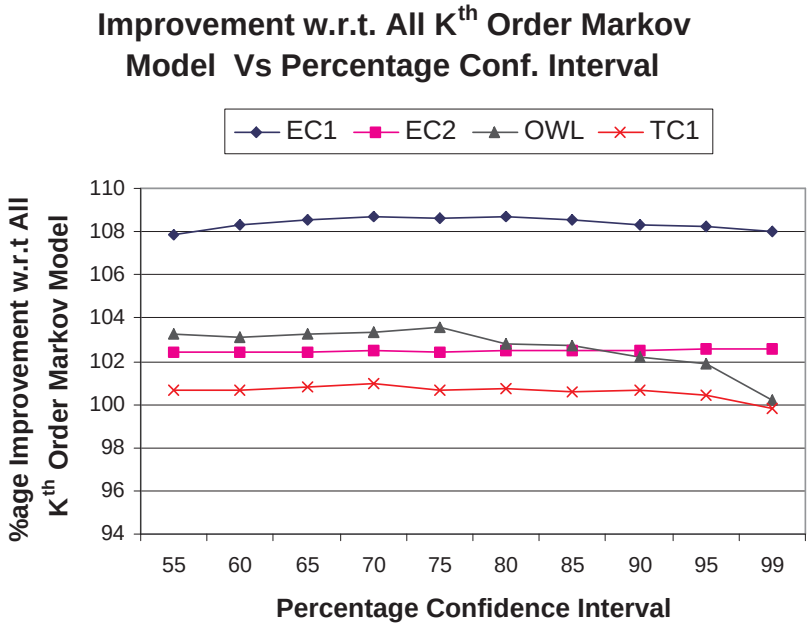

Figure 4: The accuracy obtained by CPMM for different confidence intervals relative to that obtained by All-K^{th}-Order Markov model.

if we compare their corresponding model sizes we can see that the individual error-pruned scheme leads to models that have 33% to 110% more states than the corresponding models of the overall error-pruned scheme. These results suggest that the overall error-pruned scheme is more aggressive in pruning states. We believe this may be due to the fact that in the overall error-pruned scheme, a lower-order state with low error-rate can potentially prune most of its superset higher-order states. However, in the case of the individual error-pruned scheme, since each state has a different error-rate that depends on the particular superset higher-order state, the pruning will be more selective. However, this is only one possible explanation, and we are currently studying this phenomenon to better understand it.

4.6 Overall Comparison of Different Schemes

Our last set of experiments compares the performance achieved by our different selective Markov model schemes against that achieved by the first-, second-, third-, and All K-order Markov models. These results are shown in Table 5. Note that the results for SPMM and CPMM correspond to the results obtained using the optimal frequency threshold and the optimal confidence coefficient, respectively, for each one of the four different datasets.

From the results in this table we can see that our algorithms, in general, achieve better accuracies than those achieved by the other algorithms. The error-pruned schemes have the highest accuracies, whereas the support-pruned schemes lead to models whose state-space complexity is within a factor of five of that achieved by the first-order Markov model (that has substantially worse accuracy). Comparing the overall error-pruned schemes against the All-K^{th}-Order Markov model we see that the improvement is 11.00% for EC1, 4.50% for EC2, 9.69% for OWL and 1.8% for the TC1 dataset. Furthermore, if we compare the size of

Model	*EC1*		*EC2*		*OWL*		*TC1*	
	Accuracy	*# states*	*Accuracy*	*# states*	*Accuracy*	*# states*	*Accuracy*	*# states*
First	0.260625	874	0.482292	6554	0.37378	160	0.555642	151
Second	0.285909	26866	0.473025	92040	0.424517	1902	0.555642	1441
Third	0.246662	95660	0.448759	197745	0.396665	4960	0.769909	4044
All K^{th}	0.274301	124125	0.480756	295079	0.427862	6892	0.765419	4406
SPMM	0.29623	3733	0.492658	5994	0.445587	820	0.766422	866
CPMM	0.298161	9656	0.49312	14121	0.443105	1181	0.772773	1272
O. EPPM	0.31255	10218	0.502399	24207	0.469337	1332	0.779208	868
I. EPPM	0.304461	21619	0.499666	30373	0.468451	1795	0.790908	1162

Table 5: The accuracy and model size of different Markov models on the four datasets.

their corresponding models, we find that the number of states for the error-pruned schemes are significantly lower than that of the All-K^{th}-Order Markov model. In particular for the two web-log datasets that contain the largest number of unique actions and sessions, the state space complexity of the error-pruned scheme is smaller by an order of magnitude.

To better compare the different schemes across the different datasets we performed statistical significance tests on the accuracy values of the different models. These results are shown in Table 6. The details of these tests are explained in Appendix A. Each cell in the Table 6 contains three numbers, corresponding to the number of datasets that the scheme along the row does statistically better, equal, or worse than the scheme along the column, respectively. As we can see from these results, the selective Markov model schemes perform statistically better than the traditional techniques on most datasets. Furthermore, there is not statistical difference between the accuracies obtained by the SPMM and CPMM schemes, and that the overall error-pruned scheme outperforms the individual error-pruned scheme in only one dataset.

Model	First	Second	Third	All K^{th}	Support	Conf	O.Error	I.Error
First	–	1, 1, 2	2, 0, 2	0, 0, 4	0, 0, 4	0, 0, 4	0, 0, 4	0, 0, 4
Second	2, 1, 1	–	3, 0, 1	1, 0, 3	0, 0, 4	0, 0, 4	0, 0, 4	0, 0, 4
Third	2, 0, 2	1, 0, 3	–	0, 1, 3	0, 1, 3	0, 1, 3	0, 0, 4	0, 0, 4
All K^{th}	4, 0, 0	3, 0, 1	3, 1, 0	–	0, 3, 1	0, 3, 1	0, 0, 4	0, 0, 4
Support	4, 0, 0	4, 0, 0	3, 1, 1	1, 3, 0	–	0, 4, 0	0, 0, 4	0, 0, 4
Conf	4, 0, 0	4, 0, 0	3, 1, 1	1, 3, 0	0, 4, 0	–	0, 0, 4	0, 0, 4
O.Error	4, 0, 0	4, 0, 0	4, 0, 0	4, 0, 0	4, 0, 0	4, 0, 0	–	1, 3, 0
I.Error	4, 0, 0	4, 0, 0	4, 0, 0	4, 0, 0	4, 0, 0	4, 0, 0	0, 3, 1	–

Table 6: Statistical significance comparisons between the different schemes. Each cell shows the number of datasets the scheme of the row does better, equal, or worse than the scheme on the column, respectively.

5 Conclusions

In this paper we presented a class of Markov model-based prediction algorithms that are obtained by selectively eliminating a large fraction of the states of the All-K^{th}-Order Markov model. Our experiments on a variety of datasets have shown that the resulting Markov models have a very low state-space complexity and at the same time achieve substantially better accuracies than those obtained by the traditional algorithms.

References

[AIS93] Rakesh Agrawal, Thomas Imielinski, and Arun Swami. Mining associations between sets of items in massive databases. In *Proc. of the ACM SIGMOD Int'l Conference on Management of*

Data, 1993.

[Bes95] A Bestravos. Using speculation to reduce server load and service time on www. In *Proceedings of 4th ACM International Conference of Information and Knowledge Management*, 1995.

[BP98] Sergey Brin and L. Page. The anatomy of large-scale hypertextual web search engine. In *Proceedings of the Seventh International World Wide Web Conference*, 1998.

[CHM+00] Igor Cadez, David Heckerman, Christopher Meek, Padhraic Smyth, and Steven White. Visualization of navigational patterns on web site using model based clustering. Technical Report MSR-TR-0018, Microsoft Research, Microsoft Corporation, 2000.

[CPM+98] Ed Chi, James Pitkow, Jock Mackinlay, Peter Pirolli, Rich Gossweiler, and Stuart Card. Visualizing the evolution of web ecologies. In *CHI 98*, 1998.

[CTS99] Robert Cooley, Pang-Ning Tan, and Jaideep Srivastava. Websift: The web site information filter system. In *Proceedings of the Web Usage Analysis and User Profiling Workshop*, 1999.

[DH99] J Dean and M. R. Henzinger. Finding related pages in world wide web. In *Proceedings of the Eighth International World Wide Conference*, 1999.

[KB00] Ronny Kohavi and Carla Brodley. Knowledge discovery and data mining cup part of SIGKDD 2000. In *http://www.ecn.purdue.edu/KDDCUP/*, 2000.

[Lin00] Frank Linton. Owl: A recommender system for organization-wide learning. *Journal of International Forum of Educational Technology & Society*, 2000.

[MR94] Douglas Mongomery and George Runger. *Applied Statistics and Probability for Engineers*. John Wiley & Sons Inc., 1994.

[Pap91] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw Hill, 1991.

[PM96] V.N. Padmanabham and J.C. Mogul. Using predictive prefetching to improve world wide web latency. *Computer Communication Review*, 1996.

[PP99] James Pitkow and Peter Pirolli. Mining longest repeating subsequence to predict wolrd wide web surfing. In *Second USENIX Symposium on Internet Technologies and Systems*, Boulder, C0, 1999.

[PPR96] Peter Pirolli, James Pitkow, and Ramana Rao. Silk from a sow's ear: Extracting usable structures from the web. In *CHI-96*, 1996.

[Sar00] Ramesh R. Sarukkai. Link prediction and path analysis using markov chains. In *Ninth International World Wide Web Conference*, 2000.

[SKS98] S. Schechter, M. Krishnan, and M. D. Smith. Using path profiles to predict http requests. In *Seventh International World Wide Web Conference*, 1998.

[YL99] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 1999.

A Significance Test for Comparing Accuracies

To get a better understanding of the differences in accuracies we use *p-test* for comparing the accuracy values between two schemes [YL99].

If we have two schemes A and B, which produce accuracies p_a and p_b respectively and the size of the test dataset is given as n. Then we can use the standard normal distribution for the statistic Z,

$$Z = \frac{p_a - p_b}{\sqrt{2p(1-p)/n}}$$

where,

$$p = \frac{p_a + p_b}{2}$$

Please note that the same test set was used for evaluating both the schemes.