Prediction of User's Web-Browsing Behavior: Application of Markov Model

Mamoun A. Awad and Issa Khalil

***Abstract*—Web prediction is a classification problem in which we attempt to predict the next set of Web pages that a user may visit based on the knowledge of the previously visited pages. Predicting user's behavior while serving the Internet can be applied effectively in various critical applications. Such application has traditional tradeoffs between modeling complexity and prediction accuracy. In this paper, we analyze and study Markov model and all-Kth Markov model in Web prediction. We propose a new modified Markov model to alleviate the issue of scalability in the number of paths. In addition, we present a new two-tier prediction framework that creates an example classifier *EC*, based on the training examples and the generated classifiers. We show that such framework can improve the prediction time without compromising prediction accuracy. We have used standard benchmark data sets to analyze, compare, and demonstrate the effectiveness of our techniques using variations of Markov models and association rule mining. Our experiments show the effectiveness of our modified Markov model in reducing the number of paths without compromising accuracy. Additionally, the results support our analysis conclusions that accuracy improves with higher orders of all-Kth model.**

***Index Terms*—All-Kth Markov, association rule mining (ARM), Markov model, N-gram, two-tier architecture.**

I. Introduction

WEB PREDICTION is a classification problem in which we attempt to predict the next set of Web pages that a user may visit based on the knowledge of the previously visited pages. Such knowledge of user's history of navigation within a period of time is referred to as a *session*. These sessions, which provide the source of data for training, are extracted from the logs of the Web servers, and they contain sequences of pages that users have visited along with the visit date and duration.

The Web prediction problem (WPP) can be generalized and applied in many essential industrial applications such as search engines, caching systems, recommendation systems, and wireless applications. Therefore, it is crucial to look for scalable and practical solutions that improve both training and prediction processes. Improving the prediction process can reduce the user's access times while browsing, and it can ease network traffic by avoiding visiting unnecessary pages.

Manuscript received November 23, 2010; revised May 10, 2011, September 3, 2011, and December 5, 2011; accepted January 11, 2012. Date of publication March 2, 2012; date of current version July 13, 2012. This paper was recommended by Associate Editor V. Murino.

The authors are with the Faculty of Information Technology, United Arab Emirates University, Al Ain, United Arab Emirates (e-mail: mamoun.awad@uaeu.ac.ae; ikhalil@uaeu.ac.ae).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2012.2187441

When a prediction model for a certain Web site is available, the search engine can utilize it to cache the next set of pages that the users might visit [3], [9], [12]. Such caching mitigates the latency problem of viewing Web documents particularly during Internet traffic congestion periods. Another widespread application of Web prediction is "personalization," in which users are categorized based on their interests and tastes [4]–[7]. The interests and tastes of users are captured according to the previously visited categorized Web pages. Furthermore, in the wireless domain, mobile Web prediction is used to reduce the number of clicks needed in wireless devices such as PDAs and smart phones, which helps to mitigate problems related to the display size limitations [23], [24].

In Web prediction, we face challenges in both preprocessing and prediction. Preprocessing challenges include handling large amount of data that cannot fit in the computer memory, choosing optimum sliding window size, identifying sessions, and seeking/extracting domain knowledge. Prediction challenges include long training/prediction time, low prediction accuracy, and memory limitation.

In our previous work, [19], [20], we have computed a new prediction model based on fusing several prediction models, namely, support vector machines (SVMs), artificial neural networks (ANNs), and Markov model. This model fusion, along with domain knowledge exploitation, has enabled us to considerably improve the prediction accuracy. Our previously proposed framework has shown clear improvement in prediction accuracy; however, it suffers from expensive prediction and training overhead. Additionally, some models, such as association rule mining (ARM) and SVM, do not scale well with large data sets. Furthermore, some models, such as SVM and ANN, do not handle the multiclass problem efficiently because of the large number of labels/classes involved in the WPP.

In this paper, we extend our previous work [19], [20] to successfully improve prediction accuracy using simpler probabilistic models such as Markov model and ARM. In addition, we aim to reduce the prediction time particularly when there are many prediction models to consult. Our contributions in this paper can be summarized as follows.

1) We propose a new two-tier prediction framework to improve prediction time. Such framework can accommodate various prediction models. The framework is based on creating a special classifier, dubbed *example classifier* (*EC*), during training to map each example to one classifier from a pool of classifiers produced during training. In prediction, an example goes through *EC* first to select a classifier, and after that, the selected classifier is used in

prediction. This desired feature enables us to analyze the performance of any single or combination of models and utilize the most effective ones.

2) We present an analysis study for Markov model and all-Kth model in the WPP utilizing different N-grams. Specifically, we show how accuracy is affected when using different N-grams.
3) We propose a new modified Markov model that handles the excess memory requirements in case of large data sets by reducing the number of paths during the training and testing phases.
4) We conduct extensive experiments on three benchmark data sets to study different aspects of the WPP using Markov model, modified Markov model, ARM, and all-Kth Markov model. Our analysis and results show that higher order Markov model produces better prediction accuracy. In addition, the results indicate a dramatic improvement in prediction time for our novel proposed two-tier framework. Moreover, the results demonstrate the positive effect of our proposed modified Markov model in reducing the size of the prediction models without compromising the prediction accuracy. Finally, we present experiments to study the effect of sparsity of pages, training partitioning, and ranking on the prediction accuracy.

The organization of this paper is as follows. In Section II, we present the related work. In Section III, we introduce basic background on different prediction models used in this paper. In Section IV, we present our proposed modified Markov model. In Section V, we present our new two-tier prediction framework. In Section VI, we exhibit the experiments and explain the results. In Section VII, we conclude this paper and outline some future research directions.

II. Related Work

Prediction models for addressing the WPP can be broadly classified into two different categories, namely, point-based and path-based prediction models. Path-based prediction is based on user's previous and historic path data, while point-based prediction is based on currently observed actions. Accuracy of point-based models is low due to the relatively small amount of information that could be extracted from each session to build the prediction model [25], [26].

Researchers have used various prediction models including k-nearest neighbor (kNN) [13], ANNs [15], [20], fuzzy inference [14], [15], SVMs [19], [20], Bayesian model [18], Markov model [2], [19], and others.

Recommendation systems are one of the early applications of Web prediction. Joachims *et al.* [13] propose the WebWatcher which is a path-based recommender model based on kNN and reinforcement learning. The combination of previous tours of similar users and reinforcement learning is used in recommendations. Nasraoui *et al.* [14]–[16] propose a Web recommendation system using fuzzy inferences. Clustering is applied to group profiles using hierarchical unsupervised niche clustering. Context-sensitive URL associations are inferred using a fuzzy-approximate-reasoning-based engine. Mobasher *et al.* [5] use the ARM technique in WPP and propose the frequent item set graph to match an active user session with frequent item sets and predict the next page that user is likely to visit. However, ARM suffers from well-known limitations including scalability and efficiency [5], [11].

In the context of adaptive learning, Anderson *et al.* [24] use dynamic links that provide shorter path to reach the final destination. Perkowitz and Etzioni [23] utilize adaptive Web sites based on the browsing activities. Su *et al.* [10] have proposed the N-gram prediction model and applied the all-N-gram prediction model in which several N-grams are built and used in prediction. Levene and Loizou [1] compute the information gain from the navigation trail to construct a Markov chain model to analyze the user navigation pattern through the Web. Pitkow and Pirolli [2] propose longest repeating subsequences to focus on significant surfing patterns to reduce the model complexity. Fu *et al.* [22] propose the use of N-*gram* model in mobile Web navigation. They mainly show that, the lower the order of the N-*gram*, the better the prediction accuracy and performance. Hassan *et al.* [18] use Bayesian model to focus on certain patterns such as short and long sessions, page categories, range of page views, and rank of page categories.

In our previous work [19], [20], we have successfully combined several effective prediction models along with domain knowledge exploitation to improve the prediction accuracy. However, the module endures expensive training and prediction overheads because of the large number of labels/classes involved in the WPP.

Our work is related to recommendation systems in [5] and [13]–[15] and follows N-gram and all-N-gram [2], [6], [10], [18], [23] path-based prediction models. However, it is different in three aspects. First, we synergize different prediction models to improve accuracy using simpler probabilistic models, namely, ARM and Markov. Second, we propose a two-tier framework to reduce prediction time in case of consulting many classifiers to resolve prediction. Our framework can accommodate different prediction models; hence, we can analyze the performance of any single or combination of models, and later, we utilize the most effective ones. Third, we try to reduce complexity of models by proposing a modified Markov model in which we reduce complexity by treating sessions as set of pages rather than sequence of pages.

III. Background

In this section, we present the necessary background of the well-known prediction models that we utilize in our work. We first present the N-gram representation of sessions. Next, we briefly present Markov model. Then, we introduce and analyze the all-Kth model. After that, we present the ARM model. Finally, we explain the concept of ranking in Web prediction.

A. N-Gram Representation of Sessions

In Web prediction, the best known representation of the training session is the N-*gram*. N-gram depicts sequences of page clicks by a population of users surfing a Web site. Each component of the N-gram takes specific page ID value that identifies a Web page. For example, the N-gram $\langle X_{10}, X_{21}, X_4, X_{12}\rangle$ describes the fact that the user has visited pages in the following order: page 10, page 21, page 4, and, finally, page 12. Many models further process these N-gram sessions by applying a sliding window to make training examples have the same length [2], [5].

B. Markov Model

The basic concept of Markov model is to predict the next action depending on the result of previous actions. In Web prediction, the next action corresponds to predicting the next page to be visited. The previous actions correspond to the previous pages that have already been visited. In Web prediction, the Kth-order Markov model is the probability that a user will visit the kth page provided that she has visited the ordered $k-1$ pages [10], [20]. For example, in the second-order Markov model, prediction of the next Web page is computed based only on the two Web pages previously visited.

The main advantages of Markov model are its efficiency and performance in terms of model building and prediction time. It can be easily shown that building the kth order of Markov model is linear with the size of the training set [10]. The key idea is to use an efficient data structure such as hash tables to build and keep track of each pattern along its probability. Prediction is performed in constant time because the running time of accessing an entry in a hash table is constant. Note that a specific order of Markov model cannot predict for a session that was not observed in the training set since such session will have zero probability.

C. All-Kth Markov Model

In all-Kth Markov model [2], [5], we generate all orders of Markov models and utilize them collectively in prediction. Table I presents the steps of prediction using all-kth model. Note that the function $predict(x, m_k)$ is assumed to predict the next page visited of session x using the kth-order Markov model m_k. If the m_k fails, the m_{k-1} is considered using a new session x' of length $k-1$ where x' is computed by stripping the first page ID in x. This process repeats until prediction is obtained or prediction fails. For example, given a user session $x = \langle P1, P5, P6\rangle$, prediction of all-$K$th model is performed by consulting third-order Markov model. If the prediction using third-order Markov model fails, then the second-order Markov model is consulted on the session $x' = x - P1 = \langle P5, P6\rangle$. This process repeats until reaching the first-order Markov model.

Therefore, unlike the basic Markov model, the all-Kth-order Markov model achieves better prediction [10], and it only fails when all orders of the basic Markov models fail to predict.

The running time of building all-Kth-order Markov model is linear because building each order of Markov model takes linear time with the training set size. Therefore, in order to build all-Kth-order model, we need $O(kN) = O(N)$ where k is the number of orders and N is the size of the training set. Prediction time is still constant because the worst case scenario in resolving a prediction is to consult all the kth orders of Markov model, i.e, $O(kC) = O(1)$.

TABLE I
ALL-KTH-ORDER MARKOV MODEL

```
Algorithm: All Kth Prediction
Input: user session, x, of length K
Output: Next page to be visited, p
    1. p ← predict (x, m_k).
    2. If p is not 0 then return p
    3. x ← strip first page ID from x
    4. K ← K-1
    5. if (K = 0) return 'failure'
    6. Goto step 1
    7. Stop
```

For the sake of analysis, we define the *predictability (PR)* of the kth model m_k as the ability of the kth model to predict regardless of the prediction correctness as in the following:

$$PR(m_i) = \frac{\sum_{j-1}^{N} predict\ (x_j, m_i)}{N},$$

$$\text{where } predict\ (x, m) = \begin{cases} 1 & \text{if } p(x) > 0 \\ 0 & \text{if } p(x) = 0 \end{cases} \quad (1)$$

where x is a testing example, N is the size of the testing set, and $p(x)$ is the probability distribution of session x in the training set. Intuitively, the PR of a model k is the ratio of the number of examples that k can predict to the total number of examples in the testing set. Note that the inability to predict results mostly from the lack of observed experiences.

The *confidence/accuracy* of the kth model C_k is defined as the ratio of the correctly predicted sessions to the total number of predictable sessions M [see (2)]

$$C_i = \frac{\sum_{j=1}^{M} test(x_j, m_i)}{\sum_{k=1}^{N} predict(x_k, m)}, \text{where}$$

$$test(x, m) = \begin{cases} 1 & \text{if } predict(x, m) \text{ is correct} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Based on all-Kth prediction algorithm, as shown in Table I, the probability that the kth model is unable to predict is $1 - PR(m_k)$, and the probability that $(k-1)$th model is unable to predict is $(1 - PR(m_k))(1 - PR(m_{k-1}))$.

Equations (3) and (4) are the PR and inability to predict of all-Kth model, respectively. Fig. 1 shows the prediction process of the all-Kth Markov model. Given a set of testing sessions, the kth model will be conferred with first. At this stage, the PR of the model k is $PR(m_k)$, i.e., $1 - PR(m_k)$ examples will be redirected to the $(k-1)$th model for consultation. Among those predictable examples, in the kth model, there will be probability of C_k of correct predictions and $1 - C_k$ of incorrect predictions. For the $1 - PR(m_k)$ examples redirected from the kth model to the $(k-1)$th model, there will be $PR(m_{k-1}) \times (1 - PR(m_k))$ probability that m_{k-1} examples are predictable and $(1 - PR(m_{k-1}) \times (1 - PR(m_k))$ examples are unpredictable. The probability of correct prediction for

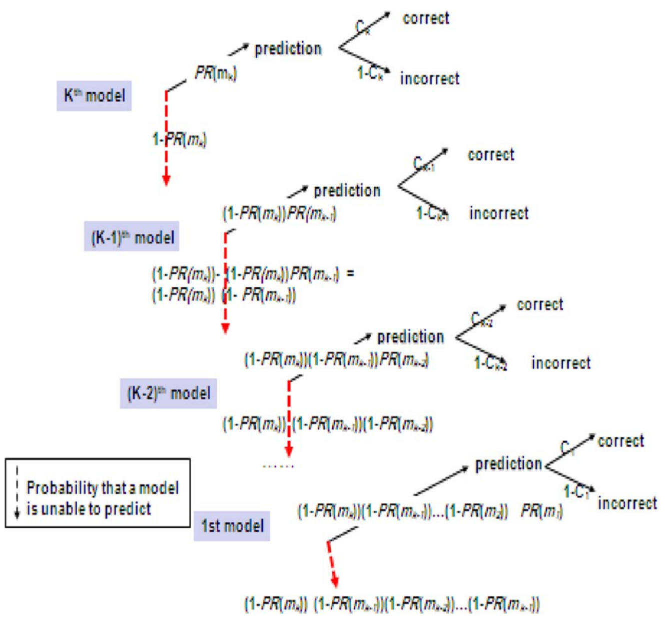
K^th model
PR(m_k)
prediction
C_k
correct
1-C_k
incorrect
1-PR(m_k)
(K-1)^th model
(1-PR(m_k))PR(m_k-1)
C_k-1
1-C_k-1
(1-PR(m_k))- (1-PR(m_k))PR(m_k-1) =
(1-PR(m_k)) (1- PR(m_k-1))
(K-2)^th model
(1-PR(m_k))(1-PR(m_k-1))PR(m_k-2)
C_k-2
1-C_k-2
(1-PR(m_k)) (1-PR(m_k-1))(1-PR(m_k-2))
......
Probability that a model is unable to predict
1st model
(1-PR(m_k))(1-PR(m_k-1))...(1-PR(m_2)) PR(m_1)
C_1
1-C_1
(1-PR(m_k)) (1-PR(m_k-1))(1-PR(m_k-2))...(1-PR(m_k-1))

Fig. 1. All-Kth-order prediction.

the $(k-1)$th model is $C_{k-1} \times PR(m_{k-1}) \times (1 - PR(m_k))$. Similarly, we can obtain the probability of correct prediction of the all-Kth Markov model by summing the probability of the correct predictions through all the models as in (5). Finally, by replacing C_i with $1 - C_i$, we can obtain the probability of incorrect prediction of the all-kth model as in (6)

$$PR_{All-Kth} = 1 - \prod_{i=1}^{K}(1 - PR(m_i)) \tag{3}$$

$$p_{All-Kth}(no\ prediction) = \prod_{i=1}^{K}(1 - PR(m_i)) \tag{4}$$

$$p_{All-Kth}(correct\ prediction) = \sum_{i=1}^{K}\left(PR(m_i)C_i \times \prod_{j=i+1}^{K}(1 - PR(m_j))\right) \tag{5}$$

$$p_{All-Kth}(incorrect\ prediction) = \sum_{i=1}^{K}\left(PR(m_i)(1 - C_i) \times \prod_{j=i+1}^{K}(1 - PR(m_j))\right). \tag{6}$$

For example, given an all-Kth model that has three-order Markov model with PR 76%, 65%, and 55% for the first, second, and third orders, respectively, then the probability of no prediction is $(1 - 0.76)(1 - 0.65)(1 - 0.55) = 0.0378$, and PR is $1 - 0.0378 = 0.9622$.

Note that (3) and (5) show that PR and correct accuracy improve with K, where K is the highest order of Markov model used. However, as we will see in the experiment section, such improvement decays with increasing K because higher order models require long sessions for training. Given the fact that the distribution of long sessions in the training set is low; therefore, the prediction model is weak. Practically speaking, one usually chooses a prospective accuracy level and as small number of kth models as possible because of the memory limitation. This means that we want to load reasonable (smaller) number of models to be used in prediction and, at the same time, we need to preserve a pragmatic accuracy level.

D. Association Rule Mining (ARM)

ARM is a data mining technique that has been applied successfully to discover related transactions. In ARM, relationships among item sets are discovered based on their co-occurrence in the transactions. Specifically, ARM focuses on associations among frequent item sets. For example, in a supermarket store, ARM helps uncover items purchased together which can be utilized for shelving and ordering processes. In the following, we briefly present how we apply ARM in WPP. For more details and background about ARM, see [6] and [8].

In WPP, prediction is conducted according to the association rules that satisfy certain support and confidence as follows. For each rule, $R = X \rightarrow Y$, of the implication, X is the user session and Y denotes the target destination page. Prediction is resolved as follows:

$$prediction(X \rightarrow Y) = \arg\max_{Y} \frac{\text{supp}(X \cup Y)}{\text{supp}(X)}, X \cap Y = \phi. \tag{7}$$

Note that the cardinality of Y can be greater than one, i.e., prediction can resolve to more than one page. Moreover, setting the minimum support plays an important role in deciding a prediction. In order to mitigate the problem of no support for $X \cup Y$, we can compute *prediction* $(X' \rightarrow Y)$, where X' is the item set of the original session after trimming the first page in the session. This process is very similar to the all-Kth Markov model. However, unlike in the all-Kth Markov model, in ARM, we do not generate several models for each separate N-gram. In the following sections, we will refer to this process as all-Kth ARM model.

Several efficient algorithms have been proposed to generate item sets and to uncover association rules such as AIS algorithm [6], [8]. However, ARM endures efficiency and scalability problems. The scalability issue originates from generating item sets which requires exponential time with the number of item sets. In this paper, we use ARM as a Web prediction solution, we study the relationship of varying the number of items and the support on the accuracy, and we compare its accuracy with other models.

E. Ranking in Prediction Resolution

Once prediction models are built, we test these models against new examples. However, in many cases, a prediction model might give several outcomes, each with different support/probability. Note that resolving the prediction to only

Sessions	Target	Frequency
P1, P2	P3	5
P2, P1	P4	1
P5, P2	P1	6
P1, P5	P4	3

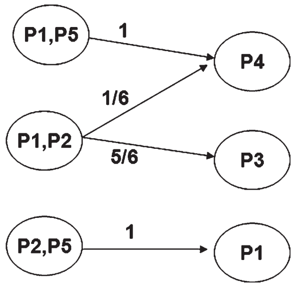

Fig. 2 Toy example: A collection of (left) user sessions and (right) probabilities of sessions.

the most probable target page would make the prediction accuracy very low because the accuracy can be computed as follows:

$$accuracy = \sum_{p \in pages} dist(p) \times pred(p) \tag{8}$$

where $dist(p)$ is the target distribution of page p and $pred(p)$ is the target prediction accuracy of page p. For example, given a Web site of two pages with the following target distribution and accuracy: $dist(p_1) = 0.4, dist(p_2) = 0.6, pred(p_1) = 0.65$, and $pred(p_2) = 0.35$, by (8), the accuracy of this Web site is 47%. Note that there are two factors that influence accuracy negatively in (8): 1) the low distribution of pages and 2) the higher interleaving targets of sessions, i.e., same session has more than one target. Unfortunately, in case of Web sites of large number of pages, these two factors affect accuracy negatively. Therefore, the prediction accuracy in Web navigation is generally low.

To mitigate this problem, we can use ranking in which we choose a bag of target pages instead of one. Specifically, we consider the first N most probable pages produced by the prediction model. Recall that a prediction model such as Markov or ARM gives several outcomes/recommendations for a specific session. For example, in Fig. 2, the 2-gram Markov model prediction for $\langle P1, P2\rangle$ is $P4$ and $P3$ with support 1/6 and 5/6, respectively. During testing, if the target page is among the top N pages predicted by the model, then we consider such prediction correct because these predicted pages will be displayed to the user and she will utilize them. In this paper, we refer to using the top N pages in prediction as using *rank* N.

IV. Modified Markov Model

In this section, we propose another variation of Markov model by reducing the number of paths in the model so that it can fit in the memory and predict faster. Recall that, in Markov model, we consider lists in building the model, for example, user sessions $S1 = \langle P1, P2\rangle$ and $S2 = \langle P2, P1\rangle$ are two different sessions; hence, each session can have different prediction probability. On the other hand, in ARM, $S1$ and $S2$ are the same item set.

The basic idea in the modified Markov model is to consider a set of pages in building the prediction model to reduce its size. For example, we consider all the sessions $\langle P1, P2, P3\rangle$, $\langle P1, P3, P2\rangle$, $\langle P2, P1, P3\rangle$, $\langle P2, P3, P1\rangle$, $\langle P3, P1, P2\rangle$, and $\langle P3, P2, P1\rangle$ as one set $P1, P2, P3$. Our motivation is that a task on the Web can be done using different paths regardless of the ordering that the users choose. In addition, we reduce the size of prediction model by discarding sessions that have repeated pages. These sessions might result when the user accidentally clicks on a link and hits the back button.

The Kth order of *modified Markov* model computes the probability that a user will visit the kth page given that she has visited the $k-1$ pages in any order as in

$$\Pr(P_k|\{P_{k-1}, \ldots, P_{k-n}\}) = \Pr(P_k|P_T). \tag{9}$$

Fig. 2 shows a toy example. Note that the last page of the session is assumed to be the final destination and it is separated from the sessions. For example, the prediction of $P1, P2$ is $P3$ and $P4$ with probabilities 5/6 and 1/6, respectively; hence, prediction is resolved to $P3$.

TABLE II
Two-Tier Framework Training Process

```
Input: M is the set of prediction models of size N; T is a set of training examples
Output: a set of trained classifiers and an example Classifier EC.
1. For each classifier model m in M train m on T
2. For each training example e in T and a classifier model m in M Do
    2.1 If m predicts the target of e correctly then map e to m and record the confidence of m in prediction.
3. For each example e in T, if e is mapped to more than one model then filter the labels so that only one label is kept.
4. Train EC on the training set T', where T' is a training set that has all examples in T and each example is mapped to one model in M.
```

V. Two-Tier Prediction Framework

In this section, we present a novel framework for Web navigation prediction. The fundamental idea is to generate different prediction models either by using different classification techniques or by using different training samples. A special prediction model, namely, *EC*, will be generated and later consulted to assign examples to the most appropriate classifier. Note that each predictor, in the generated bag of prediction models, captures strengths and weaknesses of that model depending on many factors such as the set of training examples, the structure, the flexibility, and the noise resiliency of the prediction technique.

Table II presents our proposed two-tier framework. In step 1, we train the set of classifiers M on the training set T. Note that a subset of T is sufficient for training when T is large. In step 2, the labeling process is applied in which each example e of the training set is labeled with one of the M classifiers that successfully predicts the outcome of e. Filtering the examples that have more than one mapped label is done in step 3. In step 4, the *EC* is trained on the filtered mapped training examples.

Fig. 3 shows the different input/output of each stage of the framework. At first, all classifiers are trained on the training set T. The output of the training process is N-trained classifiers. During the mapping stage, each training example e in T is mapped to one or more classifiers that succeed to predict its target. For example, in Fig. 3, $t1$ is mapped to classifier $C2$, while $t3$ is mapped to the set of classifiers $C1, C2$. The mapped

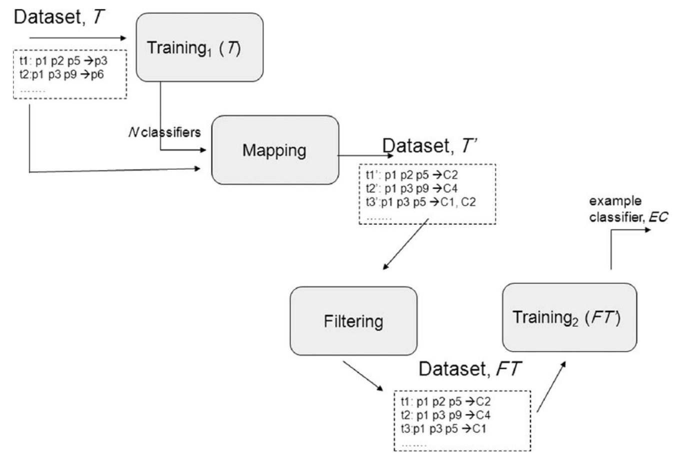

Fig. 3. Different phases in the two-tier framework training process.

training set T' undergoes a filtering process in which each example is mapped to only one classifier according to the confidence/strength of the classifiers. For example, after filtering stage, $t3$ in T' is mapped to $C1$ rather than $C1, C2$ because $C2$ predicts $t3$ correctly with higher probability, for instance. In case the models have equivalent prediction confidences, one model is selected randomly. Finally, the filtered data set FT is used to train the *EC* as the final output.

In this paper, we generate N orders of Markov models, namely, first, second, …, Nth-order Markov models, by applying sliding windows on the training set T. These prediction models represent a repository that can be used in prediction. Next, we map each training example in T to one or more orders of Markov models. For example, in Fig. 3, training example $t3 : (P1, P3, P5)$ is mapped to two classifier IDs, namely, $C1$ (first-order Markov model) and $C2$ (second-order Markov model). After that, filtering/pruning process is conducted in which each example is mapped to only one classifier. In our experiments, we choose the classifier that predicts correctly with the highest probability. For example, in Fig. 3, $t3$ is finally mapped to $C1$ because it has higher probability than $C2$ of predicting $t3$ correctly. Finally, we generate the *EC* based on training the filtered data set using one of the prediction techniques (in this paper, we use SVM). Recollect that EC can use any traditional learning technique, such as ANNs or decision trees, and it is special because it is trained based on training examples mapped to classifier IDs.

In prediction, a testing example x submits to two stages of prediction. First, x is fed to *EC* as input to predict the suitable classifier for x C_x. Next, x goes through the predicted classifier C_x to determine the final outcome (see Fig. 4). Note that, in other multiclass prediction models, such as bagging and Ada-boosting, all prediction models are consulted, and a confident scheme is used to pick the outcome of the strongest classifier. This is a time-consuming process, particularly when prediction is required online. In our framework, only one classifier is consulted which results in less prediction time and less ambiguity during prediction. It is also important to choose the prediction technique for *EC* carefully; specifically, it should be multiclass classifier (not binary classifier) and nonprobabilistic model. The reason for requiring a multiclass

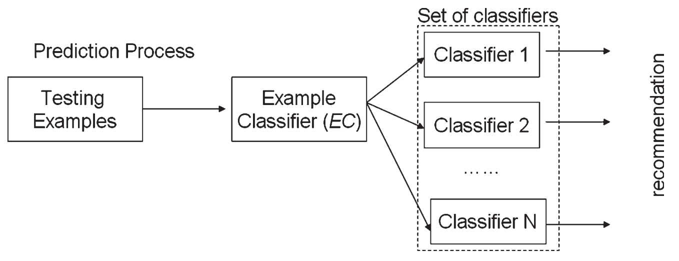

Fig. 4. Prediction process in the two-tier model.

TABLE III
SUMMARY OF NASA AND UOFS DATA SETS

	NASA	UOFS	UAEU
Total log records	1,891,714	2,408,625	5,065,074
Total sessions	118,718	172,984	283,565
Avg. session length	6.4	5.5	4.77
Number of pages	1005	5423	5195
Dataset date (month/year)	7/1995	6-12/1995	3/2011

classifier is intuitive since we need a prediction model that can predict for many labels. On the other hand, the reason behind not being a probabilistic technique is twofold. First, we need to uncover hidden associations/relationships between examples and classifiers. Such relationships cannot be found using probability and counting occurrences of examples. Second, for new experiences/examples that do not exist in the training set, the probabilistic models fail to predict because the probability of such new example is zero. So, models such as ANNs, SVM, and decision tree might be good candidates for *EC*.

VI. Evaluation

In this section, we present experimental results for Web prediction using the following prediction models: Markov, ARM, all-Kth Markov, two-tier framework, and the modified Markov model. In our experiments, we preprocess the sessions by applying a sliding window to make the sessions conform to a specific N-gram [2], [5].

We employ ranking in calculating the prediction accuracy. Recollect that the prediction can be resolved to several outcomes depending on the probabilities (for Markov model) and the minimum support (for ARM). In our experiments, we use different rankings to monitor accuracy. Recall that, in R-rank prediction, the prediction model predicts correctly if the target page is in the top R predicted pages (see Section III-E).

A. Data Sets and Preprocessing

We considered three data sets, namely, the NASA data set, the University of Saskatchewan's (UOFS) data set, and the United Arab Emirates University (UAEU) data set [21]. Table III shows a brief statistics of each data set.

In addition to many other items, the preprocessing of a data set includes the following: grouping of sessions, identifying the beginning and the end of each session, assigning a unique session ID for each session, and filtering irrelevant records. In our experiments, we follow the cleaning steps and the session identification techniques introduced in [17].

TABLE IV
INPUT PARAMETERS AND THEIR DEFAULT VALUES

Parameter	Value	Parameter	Value
Rank	1	*Max Sliding Window*	8
Training set %	67%	*Testing set %*	33%
Min Support	100	*No. of runs*	30

B. Experimental Setup

We implemented both the Markov and the ARM prediction models. In ARM, we generate the rules using the *a priori* algorithm proposed in [8]. To measure the accuracy, we follow the *generalization accuracy* procedure by partitioning each data set randomly into a training set (two-thirds of the original set) and a testing set (one-third of the original set). The generalization accuracy is a standard procedure which is widely used to measure prediction models' accuracy against new examples that might not have been observed during training. We have run each experiment 30 times, applying random partitioning/sampling on the data set, for each different criterion. In other words, each point in any of the presented curves is the average value of 30 runs of the same parameters yet different partitioning/sampling. Through the result section, unless otherwise mentioned, we use two-thirds of the data set for training, and we use rank 1. Table IV presents the default setup of our experiments.

C. Prediction Setup

Given a testing session (t) of length L, we conduct prediction using the $(L-1)$-gram Markov model and obtain the prediction to evaluate the accuracy of the model. Recall that the last page of t is the final outcome that we will evaluate the correctness of the mode against; hence, we use $(L-1)$-gram. In case t is longer than the highest N-gram used in the experiment, we apply a sliding window of size L on t. For example, suppose $t = p1, p2, p3, p4, p5$, if we use the third-order Markov model, then we break t into $p1, p2, p3, p4$ and $p2, p3, p4, p5$.

D. Results

In this section, we present and discuss the results of our experiments. Then, we collectively draw conclusions and provide recommendations based on our findings throughout these experiments.

Fig. 5 shows a comparison of the prediction accuracy of various N-grams of Markov model with different ranks using UOFS and UAEU data sets. We observe that the prediction accuracy decreases as N increases. Recall that the number of training sessions obtained while processing the data set for the N-gram model is more than the number of training sessions for the $(N+1)$-gram model. Therefore, the number of experiences which the N-gram model encounters in the data set is more than that which the $(N+1)$-gram model encounters. Hence, accuracy decreases with higher N-grams. The figure also shows that the N-gram accuracy increases with higher rank in all N-grams.

In the following set of experiments, we study the impact of the sliding window length on the accuracy of the various

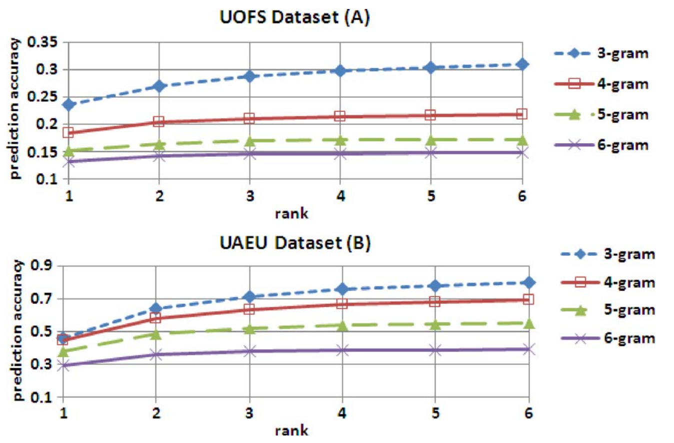

Fig. 5 Accuracy versus rank for each N-gram of Markov model using (above) UOFS data set and (below) UAEU data set.

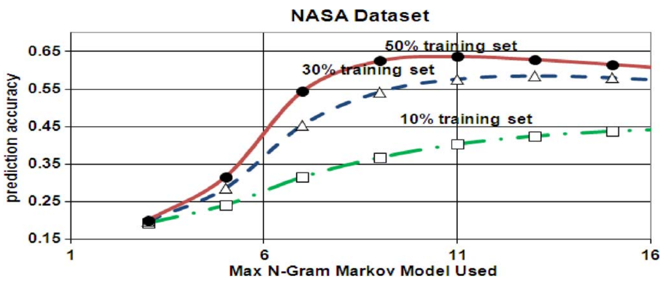

Fig. 6. Comparing prediction accuracy versus sliding window size using NASA data set and all-Kth Markov model.

prediction models. Recall that, in the all-Kth Markov model and the all-Kth ARM model, we need to choose a maximum sliding window w in order to create all the N-gram models where $N \leq w$. The value of w is critical to decide how fast the training and the prediction times will be since it decides the number of models that we need to build and consult. So, in this experiment, we try to answer the question: What is the highest K that we should consider in the all-Kth Markov model in order to achieve the highest possible accuracy? For this purpose, we plot the change in prediction accuracy against the sliding window size by performing the following steps.

1) Fix a set of parameters such as the training set percentage, rank, sliding window w, etc.
2) Calculate the accuracy of all-Kth model of sliding window w.
3) Plot the relationship between accuracy and the sliding window size.
4) Repeat steps 2) and 3) using different values for w.

Fig. 6 shows the effect of changing the maximum N-gram model used on the prediction accuracy of the NASA data set. Each curve in the figure depicts the change of prediction accuracy for the all-Kth Markov model when K increases. The figure shows that the prediction accuracy improves when higher Markov orders are used. This conforms with the results presented in Fig. 5 because the all-$(K+1)$th Markov model has the advantage of having an extra $(K+1)$-gram model which will contribute to the accumulated prediction accuracy. Furthermore, we observe from Fig. 6 that, in each curve, the accuracy reaches the peak value when the value of the sliding

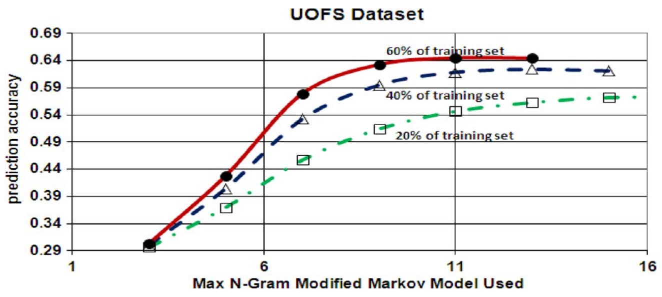

Fig. 7. Comparing prediction accuracy against sliding window size using UOFS data set and all-Kth modified Markov model.

TABLE V
SIZES OF GENERATED MARKOV MODELS BASED ON NUMBER OF PATHS (UOFS AND UAEU DATA SETS)

Size/model	Markov	Modified Markov
UOFS Dataset		
3-gram	31.9K	5.5K
4-gram	22.4K	3.1K
5-gram	7.6K	1.2K
UAEU Dataset		
3-gram	12K	203
4-gram	14.8.4K	64
5-gram	14.2K	23

window size lies between 10 and 12. When the size of the sliding window goes beyond this value, the accuracy either stabilizes or starts dropping. For example, when the size of the training set equals 30%, the prediction accuracies are 19%, 45%, 57%, and 58% for k-grams of sizes 3, 7, 11, and 13, respectively. Beyond the 13-gram, the accuracy starts dropping with values of 58%, 55%, and 52% for k-grams of sizes 17, 21, and 23, respectively. This behavior repeats for the other two curves that use 10% and 50% training sets shown in Fig. 6. To the best of our knowledge, the study presented in Figs. 6 and 7 is the first in its kind in the WPP.

The empirical results that we obtain in this set of experiments comply with (5), which states that, in all-Kth Markov model, the higher the order of Markov model, the better the accuracy. However, when we use very high orders of Markov model (large sliding window size), the curve deviates from (5). This is because larger sliding window size means less number of training sessions which causes the prediction accuracy to drop. Fig. 6 also shows that the prediction accuracy improves with larger training sets due to the availability of more experiences which results in more effective prediction model. For example, in Fig. 6, when maximum number of N-gram used is 11, accuracies are 40%, 57%, and 64% for training sets of sizes 10%, 30%, and 50%, respectively. We have encountered similar results for UAUE and UOFS data sets. Note that, in Fig. 7, we have used our proposed modified Markov model and obtained similar conclusions.

Table V compares the Markov model and the modified Markov model in terms of model size and prediction accuracy for various N-grams. The model size is measured based on the number of paths in the Markov graph. In Table V, the size of

TABLE VI
COMPARISON BETWEEN ACCURACY OF MARKOV AND MODIFIED MARKOV MODELS FOR NASA AND UAEU DATA SETS

	UAEU dataset		**NASA dataset**	
	Markov	Modified Markov	Markov	Modified Markov
1-Gram	0.199	0.578	0.161	0.264
2-Gram	0.166	0.283	0.305	0.303
3-Gram	0.121	0.125	0.261	0.196
4-Gram	0.137	0.012	0.116	0.089
5-Gram	0.112	0	0.046	0.043
6-Gram	0.261	0	0.029	0.030
7-Gram	N/A	N/A	0.079	0.071

TABLE VII
COMPARISON BETWEEN ALL-KTH MARKOV MODEL AND ALL-KTH MODIFIED MARKOV MODEL FOR THE DATA SETS UOFS AND UAEU

Model/Accuracy	All-3 Accuracy	All-7 Accuracy	All-11 Accuracy
UOFS dataset			
Markov (40%)	0.305	0.567	0.638
Modified Markov (40%)	0.302	0.532	0.618
Markov (60%)	0.308	0.620	0.664
Modified Markov (60%)	0.303	0.579	0.640
UAEU dataset			
Markov (40%)	0.517	0.603	0.634
Modified Markov (40%)	0.525	0.608	0.663
Markov (60%)	0.519	0.609	0.640
Modified Markov (60%)	0.528	0.610	0.665

the modified Markov model is reduced dramatically with larger N-grams.

Table VI presents a comparison of the prediction accuracy between Markov model and the modified Markov model for various N-grams. For example, in NASA data set, both 2-gram Markov and 2-gram modified Markov models achieve 30% accuracy, while 3-gram Markov model outperforms 3-gram modified Markov model slightly. In the same figure using UAEU data set, the prediction accuracy for the modified Markov model is higher than that in the Markov model for small N-grams, while in larger N-grams, Markov model accuracy is very close to the accuracy of the modified Markov. This is due to the small sizes of sessions in the UAEU data set.

Table VII shows that, even though the size of the modified Markov model is smaller than that of Markov model, the all-Kth modified Markov model has achieved very close results to the all-Kth Markov model. This clearly reflects the effectiveness of the modified Markov model. In Table VII, we use the all-Kth models, where each column represents the number of orders of Markov models, namely, 3, 7, and 11. For example, using 60% of the UAEU data set as training set and with all-7 orders, the all-Kth modified Markov model has achieved 61% accuracy versus 60% accuracy achieved by all-Kth Markov model. Also, using UOFS data set with 60% as training set and with all-11, all-Kth modified Markov achieved very close prediction accuracy (64%) to all-Kth Markov model (66%). We

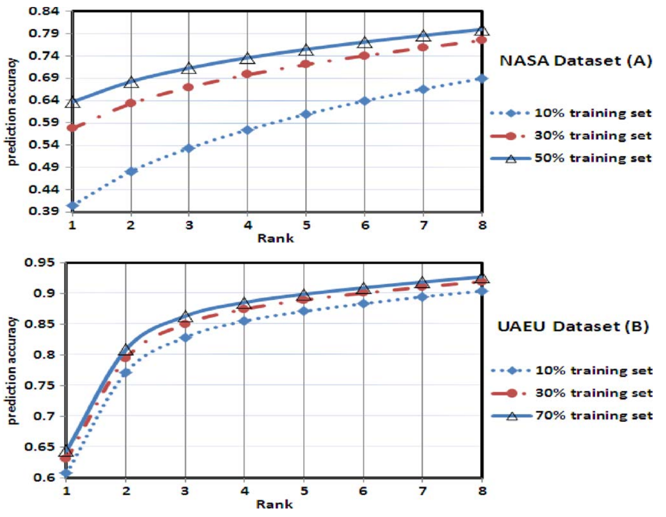

Fig. 8 Improvement of prediction accuracy with ranking using all-Kth Markov model.

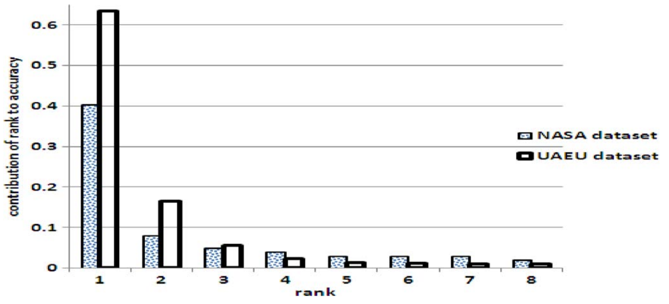

Fig. 9. Contribution of each ranking to the total prediction accuracy (all-Kth Markov model, 10% of NASA data set, and 50% of UAEU data set).

observed similar results when conducting the experiments on NASA data set with different sliding window sizes and different training partitioning sizes.

Fig. 8 shows the improvement in the prediction accuracy when increasing the rank. Recall that using rank R denotes considering the prediction correct as long as the page is resolved among the top R predictions by the model (see Section III-E). As expected, the accuracy jumped from 62% (for rank 1) to 80% (for rank 8) when considering 50% of the NASA data set as the training set. Fig. 8 (B) shows similar results using the UAEU data set. The effect of ranking in this study presents the critical role of ranking in developing Web prediction solutions.

Fig. 9 shows the contribution of each ranking on the prediction accuracy of the all-Kth Markov model using NASA and UAEU data sets with 10% and 50% training percentages, respectively. The x-axis of the figure represents the rank, and the y-axis represents the contribution of that rank in the total prediction accuracy achieved. Note that rank 1 contributes the highest percentage to the prediction accuracy among all the ranks, i.e., the higher the ranking, the less the contribution to the prediction accuracy. For example, in Fig. 9, the total accuracy for all ranks in NASA data set is 66% (see Fig. 8(a), lower curve) with 40% of the testing sessions being correctly predicted using rank 1 while 8% of the testing sessions are

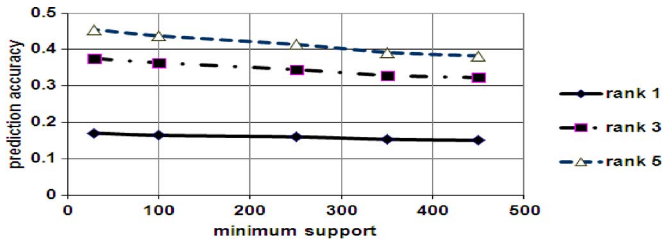

Fig. 10. Accuracy for ARM model applying different minimum support values (UOFS data set).

TABLE VIII
COMBINING ALL-KTH MARKOV AND ALL-KTH ARM MODELS $(rank = 7)$

	All-Kth Markov Accuracy	All-Kth ARM Accuracy	All-Kth ARM-alone Accuracy	**All-Kth ARM-Markov**
NASA	0.692	0.488	0.131	**0.823**
UOFS	0.748	0.507	0.089	**0.838**

correctly resolved using rank 2. Moreover, rank 8 contributes with less than 3% to the prediction accuracy.

Fig. 10 shows the effect of minimum support on prediction accuracy in ARM model. Each curve in the figure presents the prediction accuracy based on different rankings. Recall that, in ARM, increasing the minimum support reduces the number of item sets, which negatively affects prediction accuracy (see Section III-D). The results in Fig. 10 comply with such intuitive relationship between the prediction accuracy and the minimum support. For example, considering rank-3 curve in Fig. 10, the accuracy drops from 38% to 31% when the support increases from 50 to 450, respectively.

In Table VIII, we combine the all-Kth ARM model and the all-Kth Markov model to perceive its effect on accuracy. Note that fusing the two models has positive impact on the prediction accuracy. The first and second columns in the table represent the prediction accuracies of the all-Kth Markov and the all-Kth ARM models, respectively. The third column (all-Kth ARM alone) in Table VIII presents the percentage of testing examples correctly predicted by the all-Kth ARM model when the all-Kth Markov model has failed to predict. The accuracy of ARM–Markov is computed as follows:

$$Accuracy = Markov + ARM - Markov \cap ARM = Markov + ARM_{\text{alone}}.$$

This clearly shows that combining more than one model tends to synergize the prediction power of the two models.

In Fig. 11, we shed light on the effect of the number of browsed pages on the prediction accuracy. Specifically, we analyze the effect of the sparsity of pages in the data set. We run this experiment by, first, fixing the data set size, then randomly picking a set of P pages after removing from the sessions any page that is not in the random set. We repeated that using different P pages, different data set sizes, and different rankings. The figure shows that smaller number of pages implies high sparsity; hence, lack in the knowledge needed during prediction results in lower accuracy. On the other hand, when the number of pages increases, the amount of experiences

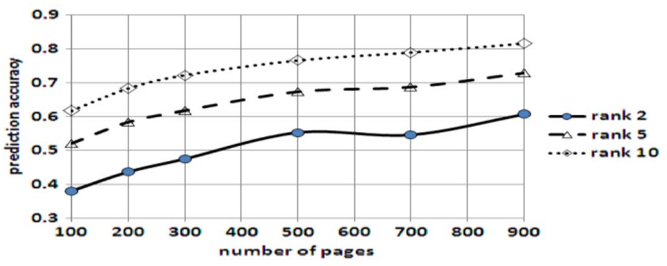

Fig. 11. Effect of the number of pages in the Web site on accuracy (all-Kth Markov model, NASA data set).

TABLE IX
COMPARING ACCURACY OF TWO-TIER FRAMEWORK VERSUS MARKOV MODEL (UOFS DATA SET)

Training %	10%	15%	20%	25%	30%	35%
2-Tier Arch. Accuracy	**0.290**	**0.302**	**0.304**	**0.309**	**0.311**	**0.314**
Markov Accuracy	0.092	0.102	0.105	0.109	0.113	0.115

grows, and according to that, accuracy augments. For example, considering rank-5 curve in Fig. 11, the prediction accuracies of 200-, 600-, and 900-page Web sites are 58%, 68%, and 73%, respectively. The presented study of page sparsity on prediction accuracy is the first, to our knowledge, in the area of Web prediction.

In the following set of experiments, we study our proposed two-tier prediction framework and compare it with Markov and all-Kth Markov models. The setup of this set of experiments is as follows. In the training process, we generate a set of N-gram Markov models, where $1 \leq N \leq 5$. Next, we generate the *EC* model necessary for the two-tier framework based on SVM model [30]. We choose SVM because it has strong theoretical foundations and high generalization accuracy [27], [28]. In our experiments, we use the LIBSVM package [30] for SVM implementation. Specifically, we use C-SVC and *RBF* kernel as values of the parameters svm_type and $kernel\_type$, respectively. For other parameters, we use the default values. Recall that *EC* should be a multiclass model; hence, we use the One-VS-one scheme in LIBSVM to generalize SVM to solve multiclass problems. We use the following confidence measure for mapping an example e to an N-gram model: $\arg_i \max predict(e, m_i)/R$, where $predict(e, m_i)$ is the probability distribution of predicting the outcome of example e, using i-gram Markov model, and R is the order/rank of the predicted page among all other predicted pages. In case two models have the same confidence, then we choose the higher N-gram model. In the prediction process of Markov model, given a session s of length l, the l-gram Markov model is used for prediction. On the other hand, for the two-tier prediction framework, the e-gram Markov model is used for prediction, where e is the N-gram model predicted by *EC* and $1 \leq e \leq 5$.

Table IX compares the accuracy prediction of two-tier architecture versus Markov model using different training set percentages. The table clearly shows the efficacy of our framework in improving the accuracy. For example, using 30% of the UOFS data set as training set, the prediction accuracy for two-tier framework is 31% versus 11% in Markov model.

TABLE X
COMPARING ACCURACY OF ALL-KTH MARKOV MODEL AND TWO-TIER FRAMEWORK (UOFS DATA SET)

Training %	10%	15%	20%	25%	30%	35%
2-Tier Arch. Accuracy	**0.575**	**0.597**	**0.609**	**0.627**	**0.628**	**0.636**
All-Kth Markov Accuracy	0.570	0.589	0.603	0.617	0.627	0.637

TABLE XI
COMPARISON BETWEEN PREDICTION TIME OF TWO-TIER FRAMEWORK AND THAT OF ALL-KTH MARKOV MODEL (UOFS DATA SET)

training %	10%	15%	20%	25%	30%	35%
2-Tier Arch. Prediction time	**1.366**	**1.332**	**1.311**	**1.309**	**1.285**	**1.276**
All-Kth Markov prediction time	3.686	3.585	3.518	3.454	3.401	3.354

This improvement in prediction accuracy is due to the *EC* classifier which provides an important knowledge for selecting the appropriate N-gram model to be used in prediction. We have repeated similar experiments on UAEU and NASA data sets, and we have attained similar results.

Table X presents a comparison between two-tier framework and all-Kth Markov in terms of prediction accuracy and using different training set percentages. In this experiment, the prediction of the two-tier framework is following the same steps of all-Kth model except that the e-gram model, where e is the Markov model predicted by *EC*, is first consulted rather than the highest order N-gram model. The table shows a very close accuracy prediction in both models. However, as Table XI shows, the prediction time in two-tier framework is reduced dramatically. For example, using 25% of the UOFS data set as training set, the prediction accuracy of two-tier framework is 63%, while it is 62% for all-Kth model. In terms of prediction time, the average prediction time for the two-tier framework is 1.3, while for the all-Kth model, it is 3.45. Note that the prediction time in Table XI is represented by the average number of N-gram models consulted per session. The prediction time reduction in the two-tier framework is due to the *EC* classifier which provides a shortcut to the most appropriate classifier to be consulted rather than starting unnecessarily from the highest order N-gram. The aforementioned study and comparisons of our novel two-tier framework show the major contributions of our work to the WPP. Our study proves that the two-tier framework can improve the prediction time dramatically without compromising prediction accuracy. This fact is invaluable particularly if we know that most applications of Web prediction are online and cannot compromise prediction time.

E. Summary and Discussion

In this section, we discuss and summarize the final findings and conclusions of our empirical results.

In terms of the training set size, the results confirm the fact that larger training set has positive impact on prediction accuracy. This can be clearly seen from the analysis of Figs. 6, 7, and other results that are not shown here, in which we find that increasing the size of the data set by 10% results in an average increase in prediction accuracy by 8%, 5.5%, and 4.5% for NASA, UOFS, and UAEU data sets, respectively. In addition, we have showed (Fig. 5) that the prediction accuracy decreases with the order of Markov model because the distribution of long sessions is low in the training set (see Table III). The empirical results show that the average decrease in prediction accuracy between N-gram and $(N+1)$-gram is 6.1% and 6.5% for the UOFS and UAEU data sets, respectively.

For the all-Kth Markov model, our empirical results in Figs. 6 and 7 confirm the theoretical analysis [(5)] which states that prediction accuracy increases with K. However, this amount of increase gets smaller with K; hence, prediction accuracy stabilizes beyond a point no matter how much K increases. Our investigation concludes that a balanced value of K should be considered in the setup because choosing higher K value for the all-Kth Markov model is not worth of the computational and storage overhead.

We have also found in Fig. 7 and Tables V–VII that the modified Markov model is comparable with Markov model in terms of prediction accuracy. This has critical impact in reducing the amount of memory used for loading the prediction models.

Regarding ranking, the empirical results indicate that prediction accuracy improves with higher rank values (see Fig. 8). However, we have observed that the improvement in prediction accuracy gets smaller with higher ranks. Therefore, a careful adjustment should be considered when selecting the suitable rank. Additionally, the results in Fig. 9 uncover the fact that the first rank contributes the most in prediction accuracy. This comes in line with user behavior on search engines in which the first two listings capture over half of the user's attention [29]. The effect of ranking in this study exposes the critical role of ranking in developing Web prediction solutions.

We have also studied page sparsity and its effect on prediction accuracy. The results in Fig. 11 indicate that page sparsity has negative impact on prediction accuracy because sparse sessions imply less number of pages used; hence, prediction accuracy declines because of lack of training experiences. The presented study of page sparsity on prediction accuracy is the first, to our knowledge, in the area of Web prediction.

Experimental results in Tables IX–XI show that the two-tier framework improves prediction time dramatically without compromising prediction accuracy. This fact is invaluable particularly if we know that most applications of Web prediction are online and cannot compromise prediction time.

Finally, the results in Figs. 10 and 11 and Table VIII show that combining more than one model improves prediction accuracy because such combination synergizes the prediction power of all the combined models.

VII. Conclusion and Future Work

In this paper, we have reviewed the current state-of-the-art solutions for the WPP. We analyzed the all-Kth Markov model and formulated its general accuracy and PR. Moreover, we proposed and presented the modified Markov model to reduce the complexity of original Markov model. The modified Markov model successfully reduces the size of the Markov model while achieving comparable prediction accuracy. Additionally, we proposed and presented a two-tier prediction framework in Web prediction. We showed that our two-tier framework contributed to preserving accuracy (although one classifier was consulted) and reducing prediction time.

We conducted extensive set of experiments using different prediction models, namely, Markov, ARM, all-Kth Markov, all-Kth ARM, and a combination of them. We performed our experiments using three different data sets, namely, NASA, UOFS, and UAEU, with various parameters such as rank, partition percentage, and the maximum number of N-grams.

Our comparative results show that large number of N-grams in the all-Kth model does not always produce better prediction accuracy. The results also show that smaller N-gram models perform better than higher N-gram models in terms of accuracy. This is because of the small number of experiences/sessions obtained during data processing of large N-grams. We have also applied ranking to improve the prediction accuracy and to enhance its applicability. Our results show that increasing the rank improves prediction accuracy. However, we have found that individual higher ranks have contributed less to the prediction accuracy. In addition, the results clearly show that better prediction is always achieved when combining all-Kth ARM and all-Kth Markov models. Finally, for the two-tier framework, our results show the efficacy of the *EC* to reduce the prediction time without compromising the prediction accuracy.

In the near future, we plan to extend this paper by conducting in-depth analysis and study of our proposed two-tier prediction framework. Additionally, we plan to explore other features in the session's logs by extracting statistical features from data sets to improve prediction accuracy. Moreover, we plan to use boosting and bagging in the same context and compare it with our hybrid approach. Finally, we will investigate the effect of applying two-tier framework in boosting/bagging.

References

[1] M. Levene and G. Loizou, "Computing the entropy of user navigation in the Web," *Int. J. Inf. Technol. Decision Making*, vol. 2, no. 3, pp. 459–476, 2003.

[2] J. Pitkow and P. Pirolli, "Mining longest repeating subsequences to predict World Wide Web surfing," in *Proc. 2nd USITS*, Boulder, CO, Oct. 1999.

[3] J. Griffioen and R. Appleton, "Reducing file system latency using a predictive approach," in *Proc. Summer USENIX Tech. Conf.*, Cambridge, MA, 1994.

[4] R. Agrawal, C. Aggarawal, and V. Prasad, "A tree projection algorithm for generation of frequent item sets," in *Proc. High Perform. Data Mining Workshop*, San Juan, Puerto Rico, 1999.

[5] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Effective personalization based on association rule discovery from Web usage data," in *Proc. ACM Workshop WIDM*, Atlanta, GA, Nov. 2001.

[6] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proc. ACM SIGMOD Conf. Manage. Data*, Washington, DC, May 1993.

[7] M. Baumgarten, A. G. Büchner, S. S. Anand, D. D. Mulvenna, and J. B. Hughes, *Navigation Pattern Discovery from Internet Data, User-Driven Navigation Pattern Discovery from Internet Data*. Heidelbert, Germany: Springer-Verlag, 2000, pp. 74–91.

[8] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. 20th Int. Conf. VLDB*, Santiago, Chile, 1994.

[9] A. Pandey, J. Srivastava, and S. Shekhar, "SIAM Workshop on Web Mining—A Web intelligent pre-fetcher for dynamic pages using

association rules—A summary of results," Univ. Minnesota, Minneapolis, MN, Tech. Rep. 01-004, 2001.

[10] Z. Su, Q. Yang, Y. Lu, and H. Zhang, "WhatNext: A prediction system for Web requests using n-gram sequence models," in *Proc. 1st Int. Conf. Web Inf. Syst. Eng. Conf.*, Hong Kong, Jun. 2000, pp. 200–207.

[11] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of recommender algorithms for e-commerce," in *Proc. 2nd ACM EC*, Minneapolis, MN, Oct. 2000.

[12] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," in *Proc. 7th Int. WWW Conf.*, Brisbane, Australia, 1998.

[13] T. Joachims, D. Freitag, and T. Mitchell, "WebWatcher: A tour guide for the World Wide Web," in *Proc. IJCAI*, 1997, pp. 770–777.

[14] O. Nasraoui and C. Petenes, "Combining Web usage mining and fuzzy inference for Website personalization," in *Proc. WebKDD*, 2003, pp. 37–46.

[15] O. Nasraoui and R. Krishnapuram, "One step evolutionary mining of context sensitive associations and Web navigation patterns," in *Proc. SIAM Int. Conf. Data Mining*, Arlington, VA, Apr. 2002, pp. 531–547.

[16] O. Nasraoui and R. Krishnapuram, "An evolutionary approach to mining robust multi-resolution web profiles and context sensitive URL associations," *Int. J. Comput. Intell. Appl.*, vol. 2, no. 3, pp. 339–348, 2002.

[17] R. Cooley, B. Mobasher, and J. Srivastava, "Data preparation for mining World Wide Web browsing patterns," *J. Knowl. Inf. Syst.*, vol. 1, no. 1, pp. 5–32, 1999.

[18] M. T. Hassan, K. N. Junejo, and A. Karim, "Learning and predicting key Web navigation patterns using Bayesian models," in *Proc. Int. Conf. Comput. Sci. Appl. II*, Seoul, Korea, 2009, pp. 877–887.

[19] M. Awad, L. Khan, and B. Thuraisingham, "Predicting WWW surfing using multiple evidence combination," *VLDB J.*, vol. 17, no. 3, pp. 401–417, May 2008.

[20] M. Awad and L. Khan, "Web navigation prediction using multiple evidence combination and domain knowledge," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 6, pp. 1054–1062, Nov. 2007.

[21] Internet Traffic Archive. [Online]. Available: http://ita.ee.lbl.gov/html/traces.html

[22] Y. Fu, H. Paul, and N. Shetty, "Improving mobile Web navigation using N-Gram prediction model," *Int. J. Intell. Inf. Technol.*, vol. 3, no. 2, pp. 51–64, 2007.

[23] M. Perkowitz and O. Etzioni, "Adaptive Web sites: An AI challenge," in *Proc. IJCAI Workshop*, Nagoya, Japan, 1997.

[24] C. R. Anderson, P. Domingos, and D. S. Weld, "Adaptive Web navigation for wireless devices," in *Proc. IJCAI Workshop*, Seattle, WA, 2001.

[25] D. W. Albrecht, I. Zukerman, and A. E. Nicholson, "Pre-sending documents on the WWW: A comparative study," in *Proc. 16th IJCAI*, 1999, pp. 1274–1279.

[26] I. Zukerman, W. Albrecht, and A. Nicholson, "Predicting user's request on the WWW," in *Proc. 17th Int. Conf. UM*, 1999, p. 393.

[27] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[28] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1999.

[29] L. A. Granka, "Thorsten Joachims, Geri Gay, Eye-tracking analysis of user behavior in WWW search," in *Proc. SIGIR*, 2004, pp. 478–479.

[30] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–39, Apr. 2011.

Mamoun A. Awad received the B.Sc. degree in computer science from the University of Baghdad, Baghdad, Iraq, in June 1994, the M.S. degree in computer science from Wichita State University, Wichita, KS, in May 1999, and the Ph.D. degree in software engineering from the University of Texas at Dallas, Richardson, in December 2005.

He is currently an Assistant Professor with the Faculty of Information Technology, United Arab Emirates University, Al Ain, United Arab Emirates, where he has taught and conducted research since August 2006. His current areas of research are Web prediction, data mining, and intrusion detection.

Issa Khalil received the B.Sc. and the M.S. degrees in computer engineering from the Jordan University of Science and Technology, Irbid, Jordan, in 1994 and 1996, respectively, and the Ph.D. degree in computer engineering from Purdue University, West Lafayette, IN, in 2007.

Immediately thereafter, he was a Postdoctoral Researcher with the Dependable Computing Systems Laboratory, Purdue University, until he joined the Faculty of Information Technology, United Arab Emirates University (UAEU), Al Ain, United Arab Emirates, in August 2007, where he is currently an Associate Professor. His research interests span the areas of wireless and wireline communication networks. He is particularly interested in security, routing, and performance of wireless sensor, ad hoc, and mesh networks. His current research is funded by grants from National Research Foundation, Emirates Foundation, and UAEU.