Web Navigation Prediction Using Multiple Evidence Combination and Domain Knowledge

Mamoun A. Awad and Latifur R. Khan

***Abstract*—Predicting users' future requests in the World Wide Web can be applied effectively in many important applications, such as web search, latency reduction, and personalization systems. Such application has traditional tradeoffs between modeling complexity and prediction accuracy. In this paper, we study several hybrid models that combine different classification techniques, namely, Markov models, artificial neural networks (ANNs), and the All-Kth-Markov model, to resolve prediction using Dempster's rule. Such fusion overcomes the inability of the Markov model in predicting beyond the training data, as well as boosts the accuracy of ANN, particularly, when dealing with a large number of classes. We also employ a reduction technique, which uses domain knowledge, to reduce the number of classifiers to improve the predictive accuracy and the prediction time of ANNs. We demonstrate the effectiveness of our hybrid models by comparing our results with widely used techniques, namely, the Markov model, the All-Kth-Markov model, and association rule mining, based on a benchmark data set.**

***Index Terms*—Artificial neural networks (ANNs), association rule mining (ARM), Dempster's rule, Markov model, N-gram.**

I. Introduction

WEB PREDICTION is the problem of predicting the next web page that a user might visit after surfing in a website. The importance of web prediction originates from the fact that various applications, such as latency reduction, web search, and recommendation systems, can be made more effective through the use and the improvement of web prediction.

One of the early applications of web prediction is the latency of viewing of web documents [6]. Traditional solutions are based on caching and prefetching [2], [3], [9]. Other advanced intelligent methods [10], [11] acquire knowledge from surfers' previous path history and utilize that in prediction. Pandey *et al.* [10] present an intelligent prefetching method based on a proxy server using association rule mining (ARM) to generate association rules that are later used to predict future requests.

World Wide Web (WWW) prediction can also improve search engines. The entire structure of the WWW can be pictured as a connected graph, where each node corresponds to a website, and surfers navigate from one node to another. The distribution of the visits over all WWW pages can be computed and used in reweighting and reranking results. In such scenario, we consider the surfer path information to be more important than the keywords that were entered by the user [14].

Another application of web prediction is recommendation systems, in which we try to find the top k users having the same interests or tastes to a target user record. ARM is a well-known model that is used in recommendation systems. Mobasher *et al.* [7] propose the frequent item set graph to match an active user session with frequent item sets and predict the next page that the user is likely to visit. Prediction for the active session is based on the confidence of the corresponding association rule.

Other prediction models that are widely used in WWW predictions and its related applications include k nearest neighbors (NN), artificial neural network (ANN), fuzzy interference, and Markov model. Joachims *et al.* [18] propose the kNN-based recommender WebWatcher. The WebWatcher is a learning tour guide agent that extracts knowledge from user's previous clicks and from the hypertext structure. Nasraoui and Krishnapuram [19] propose a web recommendation system using fuzzy inference. Clustering is applied to group profiles using hierarchical unsupervised niche clustering. Context-sensitive uniform resource locator (URL) association is inferred using a fuzzy-approximate-reasoning-based engine. Levene and Loizou [1] compute the information gain from the navigation trail to construct a Markov chain model to analyze user navigation pattern through the web. The main contribution of [1] is that they present a mechanism to estimate the navigation trail. Pitkow and Pirolli [5] explore pattern extraction and pattern matching based on the Markov model that predicts future surfing paths. Longest repeating subsequences (LRS) is proposed to reduce the model complexity (not predictive accuracy) by focusing on significant surfing patterns.

Our work is related to the path-based prediction model using the N-gram model [14] and the LRS model [5]. However, our approach differs from them in the following ways: First, only one path-based prediction technique is used when combining different N-gram models. Second, the main focus of LRS is to reduce the modeling complexity by reducing the data set. Third, all these models are probabilistic, i.e., it depends on the frequencies of patterns/occurrences in the training set. Therefore, our model can predict some values that Markov models cannot (i.e., our model can predict for some unobserved values). In the work of Nasraoui and Krishnapuram [19], the focus is to use a set of URL predictors by creating a neural network for each profile independently with a separate training set. Their goal is to overcome the high complexity of the architecture and training in case that one neural network is used. In our work, we not only use a set of predictors but also fuse them

Manuscript received October 17, 2007; revised February 21, 2007. This paper was recommended by Associate Editor M. Kamel.

M. A. Awad was with the University of Texas at Dallas, Richardson, TX 75083-0688 USA. He is now with the United Arab Emirates University, Al Ain, UAE (e-mail: mamoun.awad@uaeu.ac.ae).

L. R. Khan is with the Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688 USA (e-mail: lkhan@utdallas.edu).

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/TSMCA.2007.904781

in a hybrid model for prediction. Our goal is to improve the accuracy using different prediction techniques, namely, ANN, the Markov model, the All-Kth model, and using different N-gram models.

One important subtlety of web prediction is that web prediction is a multiclass problem of a large number of classes (11 700 classes in our experiments). Here, we define a class (or a label) as a unique identifier that represents a web page in a web site. Most multiclass techniques, such as one-versus-one and one-versus-all, are based on generalizing binary classifiers, and prediction is resolved by checking against all these binary classifiers. As a result of that, prediction accuracy is very low [4], because the prediction model has many conflicting outcomes from the classifiers.

There are several problems with the current state-of-the-art solutions. First, models such as Markov and ARM models are unable to generalize beyond training data [5]. This is because prediction using ARM and LRS pattern extraction is done based on choosing the path of the highest probability in the training set; hence, any new surfing path is misclassified, because it has zero probability. Second, prediction using ARM suffers from well-known limitations including scalability and efficiency [7], [13]. Finally, many of the previous methods have ignored domain knowledge as a means to improving prediction.

In this paper, we present a new approach to improving the accuracy in web prediction. Our approach is based on generating a hybrid prediction model by fusing two different classification models. We use four classification models, namely: 1) ANNs; 2) ARM; 3) Markov model; and 4) *All-Kth-model*. ARM and Markov model are powerful techniques for predicting seen data, i.e., already observed data; however, they cannot predict beyond training data (see Section III-A). On the other hand, the All-Kth model and ANN are powerful techniques that can predict beyond training data. In other words, the ANN and All-Kth models can predict some values that the Markov model and ARM cannot. We combine the All-Kth model with ANN by fusing their outcomes using Dempster's rule.

Nonetheless, when dealing with a large number of classes or when there is a possibility that one instance may belong to many classes, their predictive power may decrease. To overcome these shortcomings, we extract domain knowledge from the training data and incorporate such knowledge during prediction to improve prediction time and accuracy. Specifically, domain knowledge is used to eliminate irrelevant classes and reduce the conflict during prediction. Notice that we combine different prediction models in which each model has different strengths and drawbacks over other models. We strive to overcome major drawbacks in each technique and improve the predictive accuracy for the final hybrid model.

The contribution of this paper is given as follows: First, we use ANN in web navigation. Second, we incorporate domain knowledge in ANN prediction to eliminate irrelevant classes and to improve prediction time and accuracy. Third, we fuse ANN, the Markov model, and All-Kth-Markov classifiers in a hybrid prediction model using Dempster's rule [17] to improve prediction accuracy and to overcome the drawbacks of using each model separately. Finally, we compare our hybrid model with different models, namely, Markov model, ARM, All-Kth-ARM, All-Kth-Markov, and ANN using a standard benchmark data set and demonstrate the superiority of our method.

The organization of this paper is given as follows: In Section II, we present the background of the N-gram concept and sliding window. In Section III, we present different prediction models that are used in web prediction. In Section IV, we present the utilization of domain knowledge to improve prediction. In Section V, we present a new hybrid approach combining ANN, the Markov model, and the All-Kth-Markov model in web prediction using Dempster's rule for evidence combination. In Section VI, we compare our results with other methods using a standard benchmark training set. In Section VII, we summarize this paper and outline some future research.

II. N-Gram Representation of Paths

In web prediction, the available source of training data is the users' sessions, which are the user's history of navigation within a period of time. User sessions are extracted from the logs of the web servers, and it contains sequences of pages/clicks that the users have visited, time, data, and the period of time that the user stays in each page. In web prediction, the best known representation of the training session is the N-gram. N-gram is tuples of the form $\langle X_1, X_2, \ldots, X_n \rangle$ that depict sequences of page clicks by a population of users surfing a website. Each component of the N-gram takes a specific page id value that identifies a web page. For example, the N-gram $\langle X_{10}, X_{21}, X_4, X_{12} \rangle$ depicts 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 instances have the same length [5], [7]. For example, if we apply a sliding window of size 3 on the N-gram $\langle X_{10}, X_{21}, X_4, X_{12}, X_{11} \rangle$, we will have the following 3-gram sessions: $\langle X_{10}, X_{21}, X_4 \rangle$, $\langle X_{21}, X_4, X_{12} \rangle$, and $\langle X_4, X_{12}, X_{11} \rangle$. In general, the number of additional sessions using sliding window w applied on session A is $|A| - w + 1$, where $|A|$ is the length of session A.

In this paper, we also use the term *number of hops*, which is related to the sliding window. The number of hops for a session of length N is $N - 1$, i.e., the number of clicks (or hops) that the user makes to reach the last page in the session. When applying sliding window of size w, the number of hops in the resulted subsessions is $w - 1$. In the previous example, the number of hops in the resulted 3-gram sessions is 2.

III. Prediction Models

In this section, we briefly present various prediction models that have been used in web prediction. First, we present the Markov model; next, we present the ANN model along with improvement modifications.

A. Markov Model

The basic concept of the Markov model is to predict the next action, depending on the result of previous actions. In

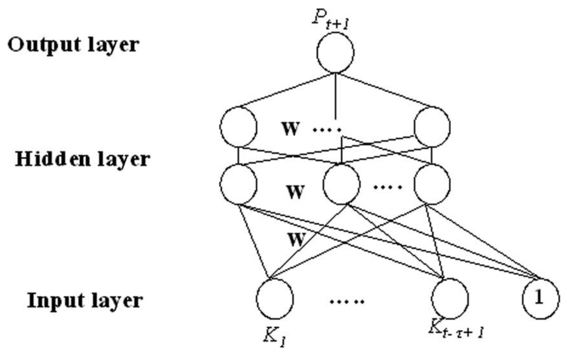


Fig. 1. Design of ANN.

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 he/she has visited $k-1$ pages, i.e.,

$$\begin{aligned}&\Pr(P_k|P_{k-1},\ldots,P_{k-n})\\&\quad=\Pr(S_k=P_k|S_{k-1}=P_{k-1},\ldots,S_{k-n}=P_{k-n})\quad(1)\end{aligned}$$

where P_i is a web page, and S_i is the corresponding state in the Markov model state diagram. Notice that the Markov model cannot predict for a session that does occur in the training set, because such session will have zero probability. Alternatively, one can generate all orders of the Markov models and utilize them in the prediction. This model is called all-Kth orders [5], [7]. The idea here is that, for a given session x of length k, the kth-order Markov model is used in the prediction. If the kth-order Markov model cannot predict for x, the $(k-1)$th-order Markov model is considered for prediction using a new session x' of length $k-1$. x' is computed by ignoring the first page id in x. This process repeats until prediction is obtained. Thus, unlike the basic Markov model, the all-Kth orders Markov model can predict beyond training data, and it fails only when all orders of basic Markov models fail to predict.

B. ANNs

ANN is a very powerful and robust classification technique that has been used in many applications and domains [15]. In this paper, we employ a network of two layers that uses the backpropagation algorithm for learning. The backpropagation algorithm attempts to minimize the squared-error function.

A typical training example in web prediction is $\langle[k_{t-\tau+1},\ldots,k_{t-1},k_t]^T,d\rangle$, where $[k_{t-\tau+1},\ldots,k_{t-1},k_t]^T$ is the input to the ANN and d is the target web page. Notice that the input units of the ANN in Fig. 1 are τ previous pages that the user has recently visited, i.e., $[k_{t-\tau+1},\ldots,k_{t-1},k_t]^T$, where k is a web page id. The output of the network is a Boolean value and not probability. We approximate the probability of the output by fitting a sigmoid function after the ANN output (see Section V-A for details). The approximated probabilistic output becomes $o'=f(o(I))=p_{t+1}$, where I is an input session and $p_{t+1}=p(d|k_{t-\tau+1},\ldots,k_t)$. We choose the sigmoid function

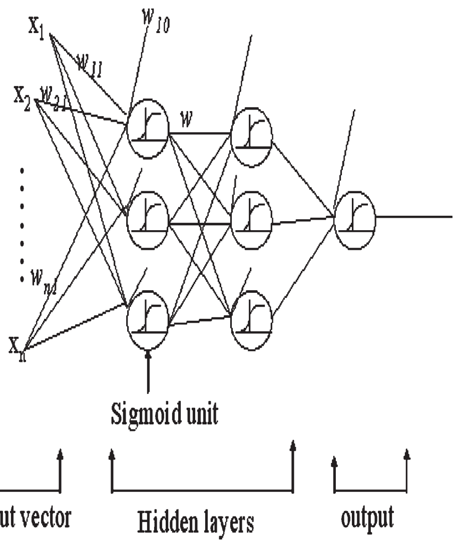


Fig. 2. ANN design in our implementation.

$$o=\sigma(w.I)\quad\sigma(y)=\frac{1}{1+e^{-y}}\tag{2}$$

as a transfer function, so that the ANN can handle nonlinearly separable data sets [15].

In (2), I is the input to the network, O is the output of the network, W is the matrix of weights, and σ is the sigmoid function. We implement the backpropagation algorithm for training the weights. The backpropagation algorithm employs gradient descent to attempt to minimize the squared error between network output values and the target values of these outputs. In our implementation, we set the step size, to update the ANN weights, dynamically based on the distribution of the classes in the data set. First, we set the step size to large values when updating the training examples that belong to low-distribution class and vice versa. This is because, when the distribution of the classes in the data set varies widely (for example, positive examples are equal to 10% and negative examples are equal to 90%), the network weights converge toward the examples from the class of larger distribution, which causes a slow convergence. Second, we adjust the learning rates slightly by applying a momentum constant to speed up the convergence of the network. Fig. 2 presents our multilayer ANN design that we use in our experiments. As we can see, the ANN is composed of two fully connected hidden layers. Each layer is composed of three neurons.

IV. Domain Knowledge and Classification Reduction

In web prediction, the number of classes/labels is large. Each page id is considered as a different label/class. For example, in our data set, we have 11 700 different page ids. Recall that, when using one-versus-one or one-versus-all, we have to consult many classifiers to resolve prediction. As a result, prediction time may increase, conflict can arise among classifiers,

	1	2	…	N
1	0	*freq(1,2)*	…	*freq(1,N)*
2	*freq(2,1)*	0	…	*freq(2,N)*
….	*freq(..,1)*	*freq(…,2)*	…	*freq(..,N)*
N	0	*freq(N,2)*	…	0

Fig. 3. Frequency matrix.

and prediction accuracy becomes low. One way to reduce/filter this large number of outcomes is to use domain knowledge in what we call *frequency matrix*. Frequency matrix is defined as an $N \times N$ matrix, where N is the number of web pages (see Fig. 3). The first row and column represent the enumeration of web page ids. Each entry in the matrix represents the frequency that the users have visited two pages in a sequence. For example, entry (1, 2) in Fig. 3 contains the frequency of users who have visited page 2 after 1. Notice that $freq(x, x)$ is always zero. We can use the frequency matrix to eliminate/filter the number of classifiers during prediction as follows: For a given session $X = \langle x_1, x_2, \ldots, x_n\rangle$ and a classifier C_i, we exclude C_i in the prediction process if $freq(x_n, c_i) = 0$, where x_n is the last page id that the user has visited in testing session X.

The frequency matrix represents the first order of Markov model. One can extend that to a higher order frequency matrix. In this case, an nth-order frequency matrix corresponds to the nth-order Markov model. Notice that the increase of frequency matrix order leads to fewer number of classes in prediction. For example, given a testing session $S_3 = \langle p_1, p_2, p_3\rangle$, the following relation holds:

$$|B_1| \geq |B_2| \geq |B_3|$$

where

$$\begin{aligned} B_1 &= \{x| < p_3, x > \in T\} \\ B_2 &= \{x| < p_2, p_3, x > \in T\} \\ B_3 &= \{x| < p_1, p_2, p_3, x > \in T\} \end{aligned}$$

where T is the training sessions, x is a page id, B_i is the set of outcomes by applying a frequency matrix of order i, and $|B_i|$ is the length of set B_i. Hence, there is a tradeoff between the number of classifiers during prediction (i.e., accuracy) and the order of frequency matrix. Based on our observations and experiments, we find that first-order frequency matrix is adequate to balance such tradeoff and to reduce the number of classifiers in prediction without affecting the accuracy. (See Section VI-D for details.)

V. Hybrid Model for Web Prediction Using Dempster's Rule

In this section, we present our hybrid model for web prediction, which is based on Dempster's rule for evidence combination, using the ANN and Markov models as bodies of evidence. In our model, prediction is resolved by fusing two separate classifiers models, namely: 1) ANN and 2) Markov model (see Fig. 4).

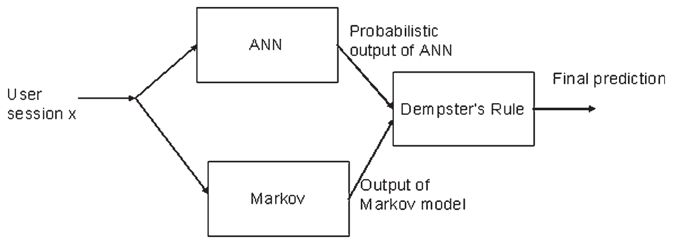


Fig. 4. Hybrid model using the Dempster's rule for evidence combination.

Dempster's rule is one part of the Dempster–Shafer Evidence Combination frame for fusing independent bodies of evidence. In Dempster's rule, the sources of evidence should be in the form of basic probability. Since the ANN output value is not probability [15], we need to convert/map this output into a posterior probability $P(class|input)$.

In this section, we will present, first, a method to convert the ANN output into a posterior probability by fitting a sigmoid function after the ANN output [16]. Next, we present the background of the Dempster–Shafer theory.

A. Fitting Sigmoid After ANN Output

We have implemented the backpropagation learning algorithm based on minimizing the squared-error function. Hence, the output of ANN cannot be considered probability. Since we are using ANN as an independent body of evidence in the Dempster's rule frame, we should consequently map the output of ANN into probability.

One interpretation of the output of ANN, in the context of the classification problem, is an estimate of the probability distribution. There are several ways to interpret the ANN output in terms of probability. One traditional way is to estimate the probability density function (pdf) from the training data. The assumption here is that we know that the training data follow some distribution (typically the normal distribution). The normal distribution is widely used as a model parameter in which analytical techniques can be applied to estimate such parameters [19], [22] as mean and standard deviation.

Another approach is to consider learning to minimize a probabilistic function, instead of squared error, such as the cross entropy shown in (3). Once learning is done, the output of the network is an estimate of the pdf. In (3), D is the training set, t_d is the target class of example d, and o_d is the output of ANN, i.e.,

$$\min - \sum_{d \in D} t_d \log(o_d) + (1 - t_d) \log(1 - o_d). \tag{3}$$

Since the backpropagation algorithm minimizes the squared-error function, we choose to implement a parametric method to fit the posterior $p(y = 1|f)$ directly, instead of estimating the class-conditional densities $p(f|y)$ [16], where y is the target class and f is the output function of ANN. The output of ANN is computed as follows:

$$f(I) = \begin{cases} 1, & \text{if } \sigma(I) \geq 0.5 \\ -1, & \text{otherwise} \end{cases} \tag{4}$$

where I is the input to the network, and σ is the output of the sigmoid transfer function defined as in (2). It follows that class-conditional densities between the margins are apparently exponential [16]. Bayes' rule on two exponential suggests using a parametric form of sigmoid as follows:

$$P(y=1|f)=\frac{1}{1+\exp(Af+B)}. \tag{5}$$

This sigmoid model is equivalent to assuming that the output of ANN is proportional to the log odds of a positive example. Parameters A and B of (5) are fitted using maximum-likelihood estimation and can be found by minimizing the negative log likelihood of training data, which is a cross-entropy error function (3). o_d in (3) is defined as follows:

$$o_d=\frac{1}{1+\exp(Af_d+B)}. \tag{6}$$

The minimization in (3) is a two-parameter minimization. Hence, it can be performed in many optimization algorithms. For robustness, we implement the model-trust minimization algorithm based on the Levenberg–Marquardt algorithm [17].

B. Dempster–Shafer Evidence Combination

The Dempster–Shafer theory is a mathematical theory of evidence [17], which is considered to be a generalization of the Bayesian theory of subjective probability. The Dempster–Shafer theory is based on two ideas. The first idea is the notion of obtaining degrees of belief for one question based on subjective probabilities for a related question, and Dempster's rule for combining such degree of belief when they are based on independent items of evidence. Since we use two independent sources of evidence, namely, ANN and Markov model, we are interested in the latter part of the Dempster–Shafer theory, namely, Dempster's rule. See [17] for more details regarding this theory.

Some may question why we do not use boosting and bagging rather than Dempster's rule to improve the classifier accuracy. We prefer Dempster's rule over boosting and bagging because boosting and bagging require partitioning the data set into a large number of independent bootstrap samples (> 1000) and then generating a classifier for each partition separately [12]. Hence, there is a computation overhead not only in training but also in preprocessing and prediction. Furthermore, boosting and bagging cannot perform effectively if the data set does not have enough points for each class/label. In web prediction applications, many pages are sparse in the data set, because they receive very few clicks.

C. Using Dempster–Shafer Theory in Web Prediction

We have two sources of evidence: 1) the output of ANN and 2) the output of Markov model. These two models operate independently. Furthermore, we assume that, for any session x for which it does not appear in the Markov model, the Markov prediction probability is zero. If we use Dempster's rule for combination of evidence, we get the following:

$$m_{\mathrm{ANN,Markov}}(C)=\frac{\sum_{A,B\subseteq\Theta,A\cap B=C} m_{\mathrm{ANN}}(A)m_{\mathrm{Markov}}(B)}{\sum_{A,B\subseteq\Theta,A\cap B\neq\phi} m_{\mathrm{ANN}}(A)m_{\mathrm{Markov}}(B)} \tag{7}$$

where m_{ANN} is the probabilistic output of ANN, m_{Markov} is the output of Markov model, $C\in 2^{\Theta}$ is a hypothesis (for example, what is the prediction of a testing session?), and Θ is the frame of discernment. A frame of discernment is an exhaustive set of mutually exclusive elements (hypothesis, propositions).

In web prediction, we can simplify this formulation because we have only beliefs for singleton classes (i.e., the final prediction is only one web page and it should not have more than one page) and the body of evidence itself $(m(\Theta))$. This means that, for any proper subset A of Θ for which we have no specific belief, $m(A)=0$. After eliminating zero terms, we get the simplified Dempster's combination rule, for a web page P_C in (8) which is shown at the bottom of the page. Since we are interested in ranking the hypothesis, we can further simplify (8), where the denominator is independent of any particular hypothesis, as follows:

$$\begin{aligned}rank(P_C)\propto\, & m_{\mathrm{ANN}}(P_C)m_{\mathrm{Markov}}(P_C)\\ & + m_{\mathrm{ANN}}(P_C)m_{\mathrm{Markov}}(\Theta)+m_{\mathrm{ANN}}(\Theta)m_{\mathrm{Markov}}(P_C).\end{aligned} \tag{9}$$

$\propto$ is the "is proportional to" relationship. $m_{\mathrm{ANN}}(\Theta)$ and $m_{\mathrm{Markov}}(\Theta)$ represent the uncertainty in the bodies of evidence for m_{ANN} and m_{Markov}, respectively. For $m_{\mathrm{ANN}}(\Theta)$ and $m_{\mathrm{Markov}}(\Theta)$ in (9), we use the following. For ANN, we use the output of ANN to compute the uncertainty. We call the output of ANN for specific session x as the margin because the ANN weights correspond to the separating surface between classes and the output ANN is the distance from this surface. Uncertainty is computed based on the maximum margin of all training examples as follows:

$$m_{\mathrm{ANN}}(\Theta)=\frac{1}{\ln(e+\mathrm{ANN}_{\mathrm{margin}})}. \tag{10}$$

$\mathrm{ANN}_{\mathrm{margin}}$ is the maximum distance of training examples from the margin. For Markov model uncertainty, we use the

$$m_{\mathrm{ANN,Markov}}(P_C)=\frac{m_{\mathrm{ANN}}(P_C)m_{\mathrm{Markov}}(P_C)+m_{\mathrm{ANN}}(P_C)m_{\mathrm{Markov}}(\Theta)+m_{\mathrm{ANN}}(\Theta)m_{\mathrm{Markov}}(P_C)}{\sum_{A,B\subseteq\Theta,A\cap B\neq\phi} m_{\mathrm{ANN}}(A)m_{\mathrm{Markov}}(B)} \tag{8}$$

maximum probability of training examples as follows:

$$m_{\text{Markov}}(\Theta) = \frac{1}{\ln(e + \text{Markov}_{\text{probability}})}. \tag{11}$$

$\text{Markov}_{\text{probability}}$ is the maximum probability of training examples. Note that, in both models, the uncertainty is inversely proportional to the corresponding maximum value.

Here, we would like to show the basic steps that are involved in web prediction using Dempster's rule.

```
Step 1) Train ANN (see Section III-B).
Step 2) Map ANN output a probability (see Section V-A).
Step 3) Compute Uncertainty (ANN) (see Section V-C, (10)).
Step 4) Construct Markov Model (see Section III-A).
Step 5) Compute Uncertainty of Markov model (see Section V-C, (11)).
Step 6) **For each** testing session $x$, **do**
    Step 6.1) Compute $m_{\text{ANN}}(x)$ and output ANN probabilities for different pages.
    Step 6.2) Compute $m_{\text{Markov}}(x)$ and output Markov probability for different pages.
    Step 6.3) Compute $m_{\text{ANN,Markov}}(x)$ using (9) and output the final prediction.
Step 7) Compute prediction accuracy. // see Section VI-C.
```

VI. Evaluation

In this section, we first define the prediction measurements that we use in our results. Second, we present the data set that we use in this paper. Third, we present the experimental setup. Finally, we present out results. In all models, we use the N-gram representation of paths [5], [7].

The following definitions will be used in the succeeding sections to measure the performance of the prediction. Pitkow and Pirolli [5] have used these parameters to measure the performance of the Markov model. These definitions are given as follows: $Pr(match)$ is the probability that a penultimate path that was observed in a validation set was matched by the same penultimate path in the training set. $Pr(hit|match)$ is the conditional probability that page x_n is correctly predicted for the testing instance $s = \langle x_{n-1}, x_{n-2}, \ldots, x_{n-k}\rangle$ and s matches a penultimate path in the training set. $Pr(hit)$ is defined as $pr(hit) = pr(match) \times pr(hit|match)$. $Pr(miss|match)$ is the conditional probability that page x_n is incorrectly classified, given that its penultimate path matches a penultimate path in the training set. $Pr(miss)$ is defined as $pr(match) \times pr(miss|match)$. Since we are considering the generalization accuracy and the training accuracy, we add two additional measurements that take into account the generalization accuracy, namely, $Pr(hit|mismatch)$ and overall accuracy. $Pr(hit|mismatch)$ is the conditional probability that page x_n is correctly predicted for the testing instance $s = \langle x_{n-1}, x_{n-2}, \ldots, x_{n-k}\rangle$ and s does not match any penultimate path in the training set. The overall accuracy is defined as $pr(hit|mismatch) \times pr(mismatch) + pr(hit|match) \times pr(match)$. The overall accuracy considers both matching and mismatching testing examples in computing the accuracy. The following relations hold for the preceding measurements:

$$Pr(hit|match) = 1 - pr(miss|match) \tag{12}$$

$$Pr(hit)/Pr(miss) = Pr(hit|match)/Pr(miss|match). \tag{13}$$

$Pr(hit|match)$ corresponds to the training accuracy, because it shows the proportion of training examples that are correctly classified. $Pr(hit|mismatch)$ corresponds to the generalization accuracy, because it shows the proportion of unobserved examples that are correctly classified. The overall accuracy combines both.

A. Data Set

For equal comparison purposes and in order to avoid duplicating already existing work, we have used the data that were collected by Pitkow and Pirolli [5] from Xerox.com for the dates May 10, 1998 and May 13, 1998. Several numbers of attributes are collected using the aforementioned method, which includes the Internet Protocol address of the user, time stamp with date and starting time, visiting URL address, referred URL address, and the browser information or agent [20].

B. Experimental Setup

We have implemented the backpropagation algorithm for multilayer neural network learning. In our experiments, we use a dynamic learning rate setup based on the distribution of the examples from different classes. Specifically, we setup the learning rate inversely to the distribution of the class, i.e., we set the learning rate to a high value for low-distribution class and vice versa.

In ARM, we generate the rules using the Apriori algorithm proposed in [9]. We set the $minsupp$ to a very low value $(minsupp = 0.0001)$ to capture the pages that were rarely visited. We implement the recommendation engine that was proposed by Mobasher *et al.* [7]. We divide the data set into three partitions. Two partitions are used in training, and one partition is used in testing.

C. Results

In this section, we present and compare the results of prediction using four different models, namely: 1) ARM; 2) ANN; 3) the Markov model; and 4) the hybrid model. In addition, we consider the *All-Kth-Markov model* and *All-Kth-ARM model*. In the following, we will refer to the results of combining the Markov model with ANN as the Dempster's rule and combining ANN with the All-Kth-Markov model as the All-Kth-Dempster's rule.

We consider up to seven hops in our experiments for all models. Results vary based on the number of hops, because different patterns are revealed for different numbers of hops. Furthermore, we introduce the concept of ranking in our results. Rank n means that prediction is considered to be correct if the predicted page happens to be among the top n pages that has the highest confidence.

TABLE I
PROBABILITY MEASUREMENTS USING ONE HOP AND RANK 1

	ARM	Markov	ANN	Dempster-Rule
Pr(Match)	0.590	0.590	0.590	0.590
Pr(Hit|Match)	0.063	0.199	0.152	0.192
Pr(Hit)	0.037	0.118	0.09	0.114
Pr(Miss|Match)	0.937	0.8	0.847	0.807
Pr(Miss)	0.555	0.474	0.502	0.478
Pr(Hit|MisMatc)	0	0	0.108	0.108
Pr(Hit)/Pr(Miss)	0.067	0.249	0.179	0.238
Over all Hit/Miss	0.038	0.134	0.155	0.188
Overall accuracy	0.037	0.118	0.134	0.158

TABLE II
RESULTS OF USING THREE HOPS AND RANK 1

	ARM	All-Kth-ARM	Markov	All-Kth-Markov	ANN	Dempster's-Rule	All-Kth-Dempster's-Rule
Pr(match)	0.376	0.376	0.376	0.376	0.376	0.376	0.376
Pr(hit|match)	0.043	0.103	0.231	0.230	0.156	0.232	0.232
Pr(hit)	0.016	0.038	0.087	0.086	0.059	0.087	0.087
Pr(miss|match)	0.956	0.89	0.768	0.769	0.843	0.767	0.767
Pr(miss)	0.360	0.337	0.289	0.289	0.289	0.288	0.288
Pr(hit|mismatch)	0	0.044	0	0.105	0.109	0.109	0.147
Pr(hit)/Pr(miss)	0.045	.114	0.3	0.299	0.185	0.302	0.302
Over all hit/miss	0.016	0.071	0.095	0.179	0.145	0.184	0.218
Overall accuracy	0.016	0.066	0.087	0.152	0.127	0.155	0.179

In Table I, there are several points to note. First, the value of $pr(hit|mismatch)$ is zero for both ARM and the Markov model, because neither model can predict for the unobserved data. Second, the Dempster's rule achieves the best scores using all measurements. For example, the training accuracy $pr(hit|match)$ for ARM, Markov, ANN, and Dempster's rule is 6%, 19%, 15%, and 19%, respectively. The overall accuracy for ARM, Markov, ANN, and Dempsters' rule is 3%, 11%, 13%, and 15%, respectively. Third, even though the $pr(hit|match)$ for ANN is less than that for the Markov model, the overall accuracy for ANN is better. This is because $pr(hit|mismatch)$ is zero in case of the Markov model, while it is 10% in case of ANN. Finally, notice that ARM has the lowest prediction results. The ARM uses the concept of frequent item sets, instead of *item lists* (ordered item set); hence, the support of one path is computed based on the frequencies of that path and its combinations. In addition, ARM is very sensitive to the $minsupp$ values. This might cause important patterns to be lost or mixed. Table II shows the results using three hops and rank 1. Notice that All-Kth-Markov outperforms the Markov model, and the All-Kth-ARM outperforms the ARM model. That is because lower orders of the models are consulted in case prediction is not possible for higher orders. As a result of this, $pr(hit|mismatch)$ is not zero in such models. For example, the $pr(hit|mismatch)$ for All-Kth-Markov and All-Kth-ARM are 10.5% and 4.4%, respectively. In addition, combining the All-Kth-Markov model with ANN using Dempster's rule has boosted the final prediction; for example, the overall accuracy for ARM, All-Kth-ARM, Markov, All-Kth-Markov, ANN, Dempster's rule, and All-Kth-Dempster's rule is 1.6%, 6.6%, 8.7%, 15.2%, 12.7%, 15.5%, and 17.9%, respectively.

In Figs. 5 and 6, the accuracy approximately increases linearly with the rank. For example, in Fig. 5, the $pr(hit|match)$ for All-Kth-Markov is 23%, 29%, 34%, 38%, 42%, 46%, 50%, and 54% for ranks 1–8, respectively. In Fig. 6, the overall accuracy of All-Kth-Markov models is 12, 17, 21, 24, 27, 30, and 35 for ranks 1–8, respectively.

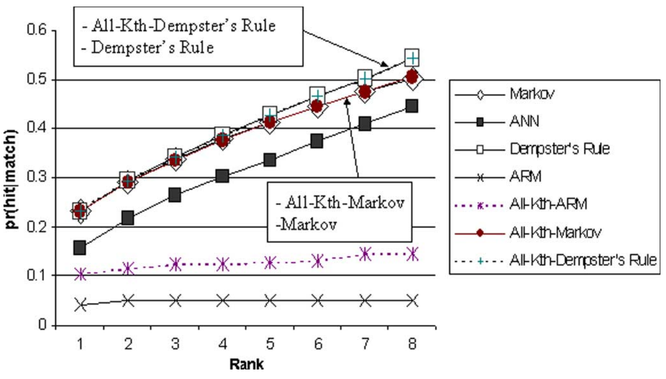


Fig. 5. $pr(hit|match)$ results using three-hop sessions and ranks from 1 to 8.

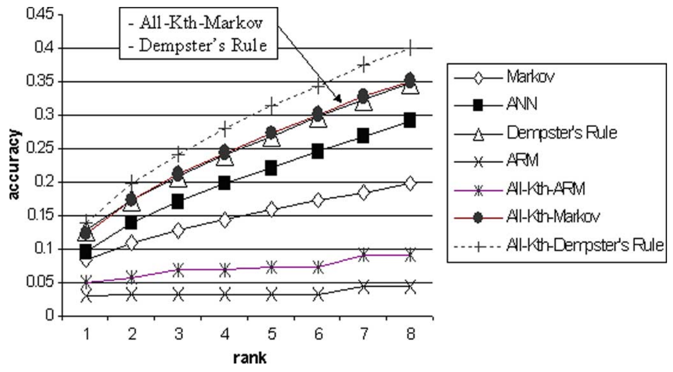


Fig. 6. Overall accuracy results using two-hop sessions and ranks from 1 to 8.

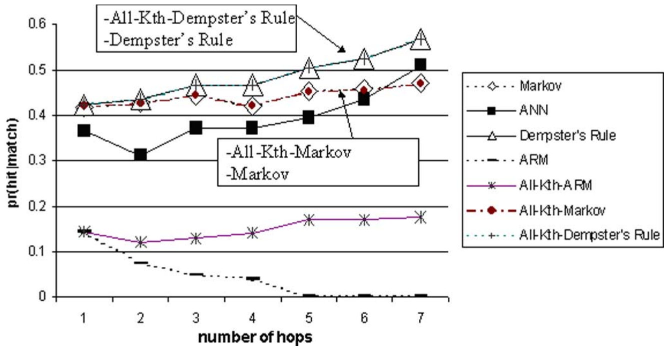


Fig. 7. Comparable results of all techniques based on $pr(hit|match)$ using rank 6.

Fig. 7 presents $pr(hit|match)$ results of using rank 6. Notice that the Dempster's rule and All-Kth-Dempster's rule methods outperform all other techniques.

In Fig. 8, we notice that All-Kth-Dempster's rule has achieved the best overall accuracy, because it combines ANN and the All-Kth-Markov model. Both models have a high training and generalization accuracy. For example, the overall accuracy using four hops for Markov, ANN, Dempster's rule, ARM, All-Kth-ARM, All-Kth-Markov, and All-Kth-Dempster's rule is 7%, 11%, 13%, 1%, 6%, 15%, and 16%, respectively. In addition, notice that ANN has outperformed the Markov model based on overall accuracy. This is because ANN generalizes better than the Markov model beyond training data.

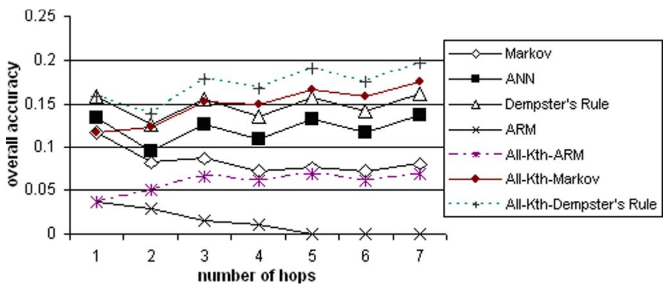


Fig. 8. Comparable results based on the overall accuracy using rank 1.

TABLE III
AVERAGE PREDICTION TIME WITH/WITHOUT DOMAIN KNOWLEDGE

Model	Average Prediction Time With Domain Knowledge (milliseconds)	Average Prediction Time Without Domain Knowledge (milliseconds)
Markov	0.567	0.544
All-Kth-Markov	1.17	0.801
ANN	6.41	556.6
Dempster's Rule	1.11	788.12

All-Kth-Dempster's rule proves to combine the best of both the ANN and All-Kth-Markov models, because it has kept its superiority over all techniques using all measurements and using different numbers of hops.

D. Effect of Domain Knowledge on Prediction

As we mentioned previously in Section IV, we have extended this model to include higher orders of domain knowledge. Recall that a frequency matrix of order n corresponds to a Markov model of order n.

Table III shows that the average prediction time using domain knowledge is 0.567, 1.17, 6.41, and 1.11 ms for the Markov, All-Kth-Markov, ANN, and Dempster's rule models, respectively. The average prediction time without using domain knowledge is 0.544, 0.801, 556.0, and 788.0 ms for the Markov, All-Kth-Markov, ANN, and Dempster's rule models, respectively. It is very evident that prediction time is reduced dramatically for ANN and Dempster's rule. The overhead in prediction without domain knowledge is a consequence of loading a very large number of classifiers, i.e., 4563 classifiers, and consulting them to resolve prediction. The prediction time in case of the Markov model and All-Kth-Markov has not been affected, because such models, contrary to ANN, can handle a multiclass problem without the used of an on-VS-all model.

In part A of Fig. 9, the overall accuracy without using any domain knowledge is 18.4%, 18.4%, 0.5%, and 15.7% for Markov, All-Kth-Markov, ANN, and Dempster's rule, respectively. The overall accuracy in case of using first-order frequency matrix (DK) is 18.4%, 18.5%, 21.6%, and 24.5% for Markov, All-Kth-Markov, ANN, and Dempster's rule, respectively. Recall that the domain knowledge is based on the frequency matrix of order n, which is another representation of the nth order of the Markov model; hence, the overall accuracy for the basic knowledge is already included in such models. On the other hand, the performance of ANN and Dempster's rule is

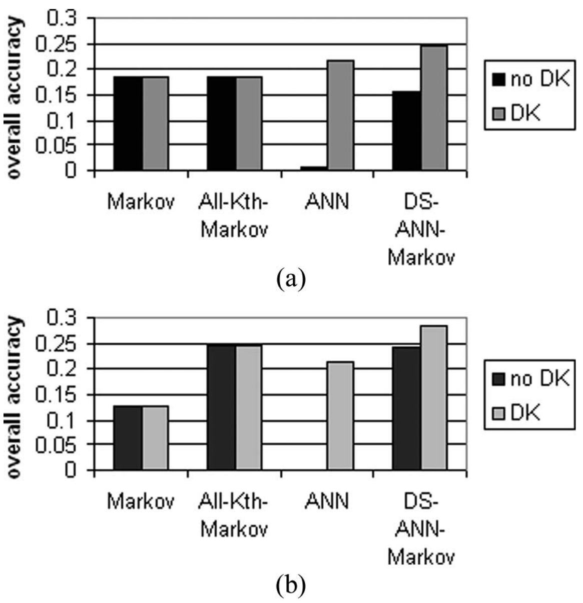


Fig. 9. Comparable results using the overall accuracy with/without domain knowledge. (a) One hop using rank 3. (b) Three hops using rank 3.

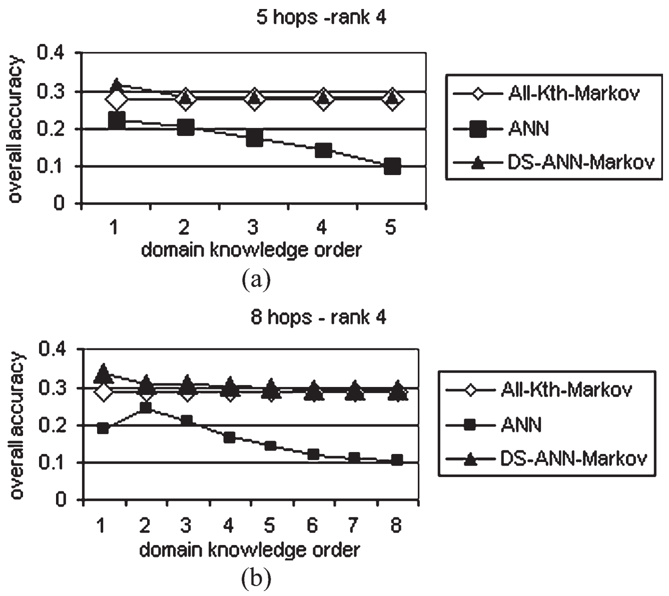


Fig. 10. Effect of domain knowledge order on the overall accuracy. (a) Five hops using rank 4. (b) Eight hops using rank 4.

affected by not using any domain knowledge, and the overall accuracy has dropped largely. Similar results can be seen in Fig. 9(b) when using three hops and rank 4. Fig. 10 presents the effect of using different orders of domain knowledge on the overall accuracy. Since we obtained similar results when using different rankings and different number of hops, we only show the results for five and eight hops using rank 4. Recall that, in the previous experiments, we considered only the first-order frequency matrix. Here, we consider a frequency matrix of different orders as domain knowledge applied to the All-Kth-Markov model, ANN, and Dempster's rule. The three curves (from top to bottom) in each subfigure represent the overall accuracy of All-Kth-Markov, ANN, and Dempster's rule. For example, the overall accuracy for Dempster's rule is 31%, 28%, 28%, 28%, and 28% using domain knowledge of orders 1, 2, 3, 4, and 5, respectively. Notice that the use of higher order for domain knowledge did not improve the accuracy. On the contrary, it slightly affects the overall accuracy negatively. This can be related to the tradeoff between the number of classifiers to consult and the order of domain knowledge. Using higher order domain knowledge leads to less number of classes to consult. This may positively affect the accuracy and speed up the retrieval process. However, this might exclude correct classes, decrease the accuracy, and finally offsets the improvement of accuracy. Conversely, not using domain knowledge leads to consulting a huge number of classifiers that cause conflict. From Fig. 10, we find that using domain knowledge of order 1 or 2 can balance such tradeoffs, because accuracy is not affected dramatically.

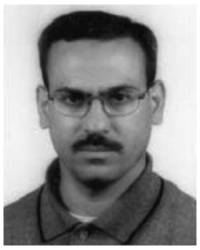
VII. Conclusion and Future Work

In this paper, we use a hybrid method in web prediction based on Dempster's rule for evidence combination to improve prediction accuracy. We used two sources of evidence/prediction in our hybrid method. The first body of evidence is ANNs. To improve the prediction of ANN further, we incorporated different orders of domain knowledge in prediction to improve prediction accuracy. The second body of evidence is the widely used Markov model, which is a probabilistic model. Furthermore, we applied the *All-Kth-Markov* model. The *All-Kth-Dempster's rule* proves its effectiveness by combining the best of ANN and the *All-Kth-Markov* model, as demonstrated by the fact that its predictive accuracy has outperformed all other techniques.

We would like to extend our research in the following directions. First, we would like to study the impact/effect of other features in the session's logs by extracting statistical features from the data set to improve accuracy. Next, we would like to perform more experiments and analyses on the effect of the frequency matrix order on prediction. Finally, we would like to use boosting and bagging in the same context, and compare it with our hybrid approach.

References

[1] M. Levene and G. Loizou, "Computing the entropy of user navigation in the web," *Int. J. Inf. Technol. Decis. Making*, vol. 2, no. 3, pp. 459–476, Sep. 2003.
[2] Q. Yang, H. Zhang, and T. Li, "Mining web logs for prediction models in WWW caching and prefetching," in *Proc. 7th ACM SIGKDD Int. Conf. KDD*, Aug. 26–29, 2001, pp. 473–478.
[3] K. Chinen and S. Yamaguchi, "An interactive prefetching proxy server for improvement of WWW latency," in *Proc. 7th Annu. Conf. INEt*, Kuala Lumpur, Malaysia, Jun. 1997.
[4] V. Chung, C. H. Li, and J. Kwok, "Dissimilarity learning for nominal data," *Pattern Recognit.*, vol. 37, no. 7, pp. 1471–1477, Jul. 2004.
[5] J. Pitkow and P. Pirolli, "Mining longest repeating subsequences to predict World Wide Web surfing," in *Proc. 2nd USITS*, Boulder, CO, Oct. 1999, pp. 139–150.
[6] J. Griffioen and R. Appleton, "Reducing file system latency using a predictive approach," in *Proc. Summer USENIX Tech. Conf.*, Cambridge, MA, 1994, pp. 197–207.
[7] 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, pp. 9–15. Held at CIKM.
[8] 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, pp. 207–216.
[9] D. Duchamp, "Prefetching hyperlinks," in *Proc. 2nd USITS*, Boulder, CO, Oct. 1999, pp. 127–138.
[10] A. Pandey, J. Srivastava, and S. Shekhar, "A web intelligent prefetcher for dynamic pages using association rules—A summary of results," in *Proc. SIAM Workshop Web Mining, Report No. 01-004*, 2001.
[11] 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.
[12] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.
[13] 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, pp. 158–167.
[14] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *Proc. 7th Int. WWW Conf.*, Brisbane, Australia, 1998, pp. 107–117.
[15] E. Parzen, "On the estimation of a probability density function and mode," *Ann. Math. Stat.*, vol. 33, no. 3, pp. 1065–1076, Sep. 1962.
[16] J. Platt, "Probabilistic outputs for SVMs and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*. Cambridge, MA: MIT Press, 1999.
[17] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton Univ. Press, 1976.
[18] T. Joachims, D. Freitag, and T. Mitchell, "WebWatcher: A tour guide for the World Wide Web," in *Proc. IJCAI*, 1997, pp. 770–777.
[19] 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.
[20] 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.
[21] T. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[22] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1992.



Mamoun A. Awad received the B.Sc. degree in computer science from Baghdad University, Baghdad, Iraq, in June 1994, the M.S. degree in computer science from Wichita State University, Wichita, Kansas, 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 in the Information Technology College, United Arab Emirates University, Al Ain, where he has taught and conducted research since August 2006. He has already published 15 conference proceeding papers, book chapters, and journal articles. His current areas of research are data mining, intrusion detection, and Web prediction.

Latifur R. Khan received the B.Sc. degree in computer science and engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in November 1993 and the M.S. and Ph.D. degrees in computer science from the University of Southern California, Los Angeles, in December 1996 and August 2000, respectively.

He is currently an Associate Professor in the Department of Computer Science, University of Texas at Dallas (UTD), Richardson, where he has taught and conducted research since September 2000, and the Director of the UTD Database and Data Mining Laboratory. He is currently on the editorial board of North Holland's *Computer Standards and Interface Journal*, published by Elsevier. He has published over 80 papers in prestigious journals and conference proceedings. His research interests include data mining, multimedia information management, and semantic Web and database systems.

Dr. Khan was a Committee Member in numerous prestigious conferences, symposiums, and workshops, including the ACM SIGKDD Conference on Knowledge Discovery and Data Mining.