

# Student Behavioral Model Based Prefetching in Online Tutoring System

Ping Ji      Jim Kurose      Beverly Woolf

Department of Computer Science

University of Massachusetts

{jiping,kurose,bev}@cs.umass.edu

Umass Computer Science Technical Report 2001-27

## 1 Overview

Web-based tutoring systems provide important advantages as a supplement to traditional classroom instruction. MANIC (Multimedia Asynchronous Networked Individualized Courseware) is an interactive multimedia WWW-based tutoring system that was originally developed in 1997 and was used by more than 200 users during the Spring 1997 semester to listen to, and view, the stored audio lectures and lecture notes for a full-semester senior-level Networking course at the University of Massachusetts [9]. Previous work has been done to analyze and improve the performance of MANIC from both system's and users' perspective [9] providing empirical and analytical characterizations of observed user behavior in MANIC. The work in [9] focused on studying the session-level behavior (e.g, the length of individual sessions) and interactive user behavior (e.g., the time between starting/stopping/pausing the audio within a session). [14] built and analyze a student model for MANIC, and determining the level of difficulty and the learning style preferences of a student by using a Naive Bayes Classifier.

Differing from previous work on MANIC, we will use a Hidden Markov Model (HMM) approach to capture students' behavior individually and study the use of HMMs to implement prediction algorithms for prefetching lecture notes. Some past research has been done to predict HTTP requests [13] and predict requests to web servers [2], but they were all from the server's (system's) point of view. Our focus is on characterizing the behavior of individual users. Work has also been done to construct a user model for tutoring systems using machine learning techniques [1]. Some of the corresponding research studies the interaction of teaching activities and learning behaviors [10], and attempts to determine the students' learning goal [3]. However, the focus of their research is to

select appropriate teaching strategies for a tutoring system. Varying from this related research, the purpose of our study on user behavior is to capture students' access activities to a web-based tutor. Additionally, none of the past related work uses Hidden Markov Model to drive the prediction process. Sarukkai, [12], used Markov chains to model web link sequences, but did not consider individual user behavior.

Our goal in this project is to use an hidden Markov model approach to model students' behavior using the web-based educational system — MANIC , to understand how students interact with this online tutoring system, so that we can predict future browsing actions of specific users and predictively prefetch the next slide for a student. This work consists of the following parts:

- We use empirical data gathered from previous online access records of MANIC. Each student has an individual log file recording his/her browsing action (i.e., next slide, previous slide, etc.). The log files are modified to sequences of actions that are used in determining the parameters of the Hidden Markov Models.
- For each student, we use a Hidden Markov Model (HMM) to model his/her accessing behavior.
- The accuracy of an HMM parameter estimation process is evaluated by feeding an output sequence generated by a known Markov Model into the HMM and comparing the parameters of the known model with those of the re-estimated model.
- We examine the performance of three algorithms that use an HMM in different ways to predict user behavior. We compare these algorithms across a range of model sizes (number of hidden states), different values of an *error threshold*, which is a bound of the number of wrong predictions, and different *session window sizes*, which controls the history data that is used to re-estimate the parameters of a HMM. We will introduce the definition of *error threshold* and *session window size* in detail later.

Our results show that a Hidden Markov Model can capture the user behavior of most of the students using MANIC very well. For a few students, their browsing preferences are hard to predict. However, by varying the method of using different *history data* to re-estimate an HMM, those students' behavior can be predicted better.

This report is organized as follows. In Section 2, we will give a brief description for the problem being studied in this synthesis project. Section 3 introduces the concept of a Hidden Markov Model based on Rabiner's HMM notation [11], and presents the user behavior model of MANIC. Section 4 proposes three observation data use algorithms, which use observation data (i.e., empirically measured student interactions with MANIC) in different ways to determine the HMM parameters. Section 5 and Section 6 introduces two approaches of determining

the prediction decision. And based on either of them we evaluate the prediction performance provided by using different observation data use algorithms. In Section 7, we conclude our work.

## 2 Problem Description

MANIC was used by more than 200 users during the Spring 1997 semester to listen to, and view, the stored audio lectures and lecture notes for a full-semester senior-level course on Computer Networking [9]. From the access log files of MANIC, we can identify several basic *browsing actions*, such as go to the next slide, go to the previous slide, use the index choose the next slide, and turn on/off audio.

In this project, our goal is to predict the next *action* of a user, so that if there is a new slide to be viewed, the system can prefetch it in advance. Therefore, the user actions that we are most interested in are those used to select a new slide. More specifically, the *next slide call* and the *index call* are very important. There are two ways for a student to show the next slide, one is caused by the user clicking the “next” button, the other is caused implicitly by continuous audio payout. We will focus on the study of predicting the students’ path through the slides, using a Hidden Markov Model (HMM) to model each individual student’s behavior.

## 3 Hidden Markov Models as User Behavioral Models

### 3.1 HMM

The concepts underlying the Hidden Markov Model approach we employ in this study are based on [4, 11, 8, 15]. In this project, we use Rabiner’s HMM notation [11], in which  $q_t$  and  $O_t$  are variables denoting the HMM state and observed output at time  $t$ , respectively.  $\pi_i$  is the stationary probability of state  $S_i$ ,  $a_{ij}$  is the transition probability from state  $S_i$  to state  $S_j$ , and  $b_i(k)$  is the probability of generating output symbol  $k$  when the HMM is in state  $S_i$ .  $\lambda$  denotes the parameter set  $(A, B, \pi)$ , where  $A$  is the matrix of  $a_{ij}$ ,  $B$  is the matrix of  $b_i(k)$  and  $\pi$  is the matrix of stationary probabilities  $\pi_i$ . A central problem in HMM’s is to determine  $\lambda$  from a sequence of observed output symbols. Also,  $N$  is the number of hidden states and  $M$  is the number of output symbols.

In Rabiner’s version of the HMM, an observed output is a probabilistic function of the state of the HMM. That is, the resulting model (which is called a hidden Markov model) is a doubly embedded stochastic process with an underlying stochastic process that is not observable, but can only be observed through another set of stochastic

processes that produce the sequence of observations [11]. Based on the sequences of observations, the parameter set  $\lambda$  can be estimated using the *Baum-Welch* algorithm, which we will describe in detail later.

In our study of students' behavior in the MANIC system, user actions are interpreted as observed outputs of HMMs. One HMM is associated with one user (student). Based on a given observed output sequence  $O$ , the *forward* algorithm is used to estimate  $P[O|\lambda]$ , where the *forward variable*  $\alpha_t(i)$  is defined as

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i | \lambda) \quad (1)$$

i.e., the probability of the partial observation sequence,  $O_1 O_2 \cdots O_t$ , and state  $S_i$  at time  $t$ , given model parameters  $\lambda$ . Here,  $\alpha_t(i)$  can be computed inductively, as follows:

1) Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N \quad (2)$$

2) Induction:

$$\alpha_{t+1}(j) = \left[ \sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \quad 1 \leq t \leq T-1, 1 \leq j \leq N \quad (3)$$

3) Termination:

$$P(O|\lambda) = \sum_{i=1}^N \alpha_T(i) \quad (4)$$

where  $T$  is the number of transitions (equivalently observed outputs) in the output sequence,  $O$ . Working together with the *forward algorithm*, the *backward* algorithm also plays a role in the parameter re-estimation scheme for a HMM. Similar to the *forward* algorithm, in the *backward* algorithm, a *backward variable*  $\beta_t(i)$  is defined as

$$\beta_t(i) = P(O_{t+1} O_{t+2} \cdots O_T | q_t = S_i, \lambda) \quad (5)$$

i.e., the probability of the partial observation sequence from  $t+1$  to the end, given state  $S_i$  at time  $t$  and model parameters  $\lambda$ . We can similarly solve for  $\beta_t(i)$  inductively as follows:

1) Initialization:

$$\beta_T(i) = 1, \quad 1 \leq i \leq N \quad (6)$$

2) Induction:

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \quad t = T-1, T-2, \dots, 1; \quad 1 \leq i \leq N \quad (7)$$

The initialization step 1) arbitrarily defines  $\beta_T(i)$  to be 1 for all states  $S_i$ . Step 2) shows that in order to have been in state  $S_i$  at time  $t$ , and to account for the observation sequence from time  $t+1$  on, we have to consider all possible states  $S_j$  at time  $t+1$ , accounting for the transition from  $S_i$  to  $S_j$ , as well as observation  $O_{t+1}$  in state  $S_j$ , and then accounting for the remaining partial observation sequence from state  $S_j$  onwards.

Let us now consider the problem of estimating the parameters,  $\lambda$ , of a HMM with a given number of states. For a HMM, our goal is to estimate its parameter set  $\lambda = (A, B, \pi)$  to maximize the conditional probability  $P(O|\lambda)$ . For doing this, the *Baum-Welch* algorithm is implemented as follows:

1. Let initial model be  $\lambda$ ;
2. Compute new model  $\bar{\lambda}$  based on  $\lambda$  and observation sequence  $O$ ;
3. If  $\log P(O|\bar{\lambda}) - \log P(O|\lambda) < \Delta$ , then stop;
4. else set  $\lambda \leftarrow \bar{\lambda}$  and goto step 2;

Here  $\Delta$  is a very small value. The re-estimated parameter set  $\bar{\lambda}$  is iteratively used in place of current parameters  $\lambda$ , until the probability of  $O$  being observed from the model reaches some limiting point. The final result of this reestimation procedure is called a *maximum likelihood* estimate of the HMM.

Clearly, the key step in Baum-Welch algorithm is step 2), where we compute a new  $\bar{\lambda}$  based on current parameters  $\lambda$  and the observation sequence  $O$ . For doing this we need to define and calculate the following variables:

- Define  $\xi(i, j)$  as the probability of being in state  $S_i$  at time  $t$  and in state  $S_j$  at time  $t+1$ .

$$\xi(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_t(j)}{P(O|\lambda)} = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \quad (8)$$

Note,  $\xi(i, j)$  depends on  $O_t$ , but following the notation of Rabiner, we suppress showing this explicit dependency.

- Define  $\gamma_t(i)$  as the probability of being in state  $S_i$  at time  $t$ , given the observation sequence  $O$ .

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j) \quad (9)$$

- $\sum_{t=1}^T \gamma_t(i)$  is the expected number of times state  $S_i$  is visited during the  $T$  transition steps
- $\sum_{t=1}^{T-1} \xi_t(i, j)$  is the expected number of transitions from state  $S_i$  to state  $S_j$  in  $T$  transition steps
- $\bar{a}_{ij} = \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i} = \frac{\sum_t \xi_t(i, j)}{\sum_t \gamma_t(i)}$
- $\bar{b}_j(k) = \frac{\text{expected number of times in state } S_j \text{ and observing symbol } k}{\text{expected number of times in state } S_j} = \frac{\sum_{t, O_t=k} \gamma_t(j)}{\sum_t \gamma_t(j)}$
- $\bar{\pi}_i = \text{probability of being in state } S_i \text{ at time } (t = 1) = \gamma_1(i), \quad \sum_{i=1}^N \bar{\pi}_i = 1$
- $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$

Here,  $\bar{A}$  is the matrix of the new estimated transition probabilities,  $\bar{a}_{ij}$ ;  $\bar{B}$  denotes the matrix of the new estimated probability distribution of generating output symbol ( $k$ ) at each state,  $\bar{b}_j(k)$ ; and  $\bar{\pi}$  denotes the matrix of the new estimated stationary probability of each state,  $\bar{\pi}_i$ .  $\bar{a}_{ij}$ ,  $\bar{b}_j(k)$  and  $\bar{\pi}_i$  are computed based on the old HMM ( $\lambda$ ), and together  $(\bar{A}, \bar{B}, \bar{\pi})$  constitute the new HMM parameter set,  $(\bar{\lambda})$ .

We have presented how the parameters of a HMM are iteratively re-estimated using the *Baum-Welch* algorithm. The accuracy of a HMM's parameter-estimation process can be examined by following steps: feed an output sequence generated by a *known* Markov Model into the HMM (with the same number of states as the *known* model) to be estimated, have the HMM "trained" using that output sequence (i.e., estimate the HMM parameters from the observed output sequency using *Baum-Welch*), and compare the parameters of the original *known* model with those of the re-estimated HMM. Our HMM parameter re-estimation process is written based on the package from [5]. We evaluate the re-estimation process of our HMM for MANIC user behavior using the above approach.

### 3.2 User Behavioral Model

The problem of predicting students' behavior for the MANIC system is to predict the *action* that a student takes for the next step based on his/her previous *browsing actions*. For each user, we can convert his/her log file into sessions of actions. We define one session as beginning when a user starts using MANIC to view a sequence of slides until that user goes offline; for a detailed discussion of the notion of a student session in MANIC, see [9]. During each session, a user's possible actions are *call the next slide*, *call the previous slide*, *use index*, and *start*.

We use a symbol to denote one action of a user. Thus we have a symbol set as  $V = \{1,2,3,4\}$ , which simply represents the *next*, *previous*, *index*, and *start* actions respectively. By this mapping, we translate the original user log files into sequences of symbols, where each sequence of symbols indicates a MANIC session.

Recall that with an HMM, we are assuming there is an underlying stochastic process that is not observable, and that this hidden stochastic process forms a Markov chain. At each state of that hidden Markov model, we consider the *action symbol* of a student as a random variable, having a probability distribution that is a function of the state. We also assume that the symbol sequences are used as observed outputs of that Hidden Markov Model. Based on the sequences of observations, the parameters of a HMM for each user will be estimated by using the *forward-backward* and *Baum-Welch* algorithms.

## 4 Prediction Algorithm Design

When an HMM ( $\lambda$ ) has been obtained based on the history data of a student, we can use this HMM to predict the next browsing action. Given the observed actions and given a Hidden Markov Model ( $\lambda$ ), we can evaluate the probability distribution of the next output by using the *forward* variable  $\alpha_t(i)$  identified in last section. Recall that  $\alpha_t(i)$  is the probability of the partial observation sequence,  $O_1 O_2 \cdots O_t$ , and state  $S_i$  at time  $t$ , given the model parameter  $\lambda$ . By using  $\alpha_t(i)$ , the probability of symbol  $k$  ( $k \in V$ ) appearing as output  $O_{t+1}$ ,  $P(O_{t+1} = k)$ , can be estimated as follows,

$$\begin{aligned} P(O_1 = k) &= \sum_{i=1}^N \pi_i b_i(k) \\ P(O_{t+1} = k) &= \sum_{j=1}^N \sum_{i=1}^N \alpha_t(i) \cdot a_{ij} \cdot b_j(k) \end{aligned} \quad t = 1, 2, \dots, T - 1 \quad (10)$$

where  $k \in V$  and  $T$  is the maximum number of steps of a specific session.  $P(O_t = k)$  can be used to make a prediction decision, deciding which action to predict, for the next step. Note that each of the possible actions may have a non-zero probability of occurring. Which of these possible actions should we predict? In this project, we will evaluate two schemes of selecting a symbol as the predicted output of an HMM, based on the estimated occurring probability,  $P(O_t = k)$ . One is to predict the action with the highest probability of occurrence, i.e.,  $\arg \max_k \{P(O_t = k)\}$ , as the model's prediction for the next step. The other is to use a threshold,  $\delta$ , to control the prediction decision. We will discuss these two action selection schemes in Section 5 and Section 6 successively.

For now, independent to the action selection schemes, we examine three different approaches for using observation data in determining the HMM parameter,  $\lambda$ . We will call these algorithms "observation data use" (ODU) algorithms below

- ODU Algorithm 1 (ODU1): Initially, an HMM with a random parameter set is generated. On loading the first observation sequence of a user, the parameters of that HMM are re-estimated. When a new session starts, the newly re-estimated HMM is used to predict that student's behavior. When a session is finished, the HMM of the student is re-estimated offline based on *all* of the student's session(s) thus far.
- ODU Algorithm 2 (ODU2): ODU2 is a modified algorithm based on ODU1. In ODU2, instead of re-estimating the HMM at the end of each session, we record a number of wrong predictions ( $c$ ), which counts the number of predictions that do not match the corresponding real actions. If  $c$  exceeds a threshold  $\phi$ , a re-estimation of HMM will be triggered. After this HMM is re-estimated again using *all* of the observations thus far for that student, the *updated* HMM will be used to do the prediction for this session from then on, and  $c$  is reset to 0. At the end of each session, the HMM is always re-estimated, regardless whether  $c$  is larger than  $\phi$ .

The reason for developing this algorithm is to avoid a consistently poor prediction for a very long session caused by a non-adaptive HMM. A student could possibly change his/her learning style for one session. In this case, if we use the HMM that is estimated by old data, in some cases, it will not predict the student's new behavior very well. In ODU1, the re-estimation only happens at the end of each session, which does not (by definition) consider the changes of user behavior during the session being predicted.

- ODU Algorithm 3 (ODU3): ODU3 is a modified version of ODU2. ODU3 not only sets a threshold  $\phi$  for the number of wrong predictions, but also has a *sliding window*. Only the observed sessions (sequences) that are contained in the sliding window contribute to the re-estimation process. More precisely, if the window size,  $w$ , is 3, then only the most recent 3 sessions will be used for the parameter re-estimation. In this way, the history data that lies outside of that window will not be considered. This algorithm reduces the effect of *old* history data, which may not represent the current study style of a student. The *sliding window* scheme was also used in [6], which focused on the study of using hidden Markov Models as user profiles for an anomaly detection task in a Human Computer Interface Modeling scenario.

In the following two Sections, we will evaluate the prediction performance for all of those three ODU algorithms, by using two different action selection schemes.

## 5 Experiment Results and Evaluation, Using $\arg \max_k \{P(O_t = k)\}$ as the Prediction

As we discussed in previous section, for each output action of an HMM, we can use the forward variable,  $\alpha_t(i)$ , to estimate the probability of symbol  $k$  appearing,  $P(O_t = k)$ . Intuitively, given the value  $P(O_t = k)$  for each symbol  $k \in V$ , we should always consider the action with the highest probability of occurrence,  $\arg \max_k \{P(O_t = k)\}$ , as the model's prediction for an output. In this Section, we will use this action selection scheme (we call it action selection scheme 1 below) to evaluate the prediction performance of three ODU algorithms. In Section 6, we will consider a second action selection scheme, which will be called action selection scheme 2, and we will also discuss the motivation of using that method and evaluate the performance of three ODU algorithms again, by using action selection scheme 2.

### 5.1 Notation Definition

From the students' log files of MANIC, we estimate that around 75% of the students' actions are *call for next slide* (the output symbol  $k = 1$ ). More formally, let us define

$$P[k = 1] = \frac{\sum_{n=1}^{\tilde{N}} \sum_{t=1}^{T_n} \mathbf{1}(O_t = 1)}{\sum_{n=1}^{\tilde{N}} T_n} \quad (11)$$

where the random variable  $k$  denotes an output symbol for a single step,  $\tilde{N}$  is the total number of students,  $T_n$  is the number of actions that student  $n$  has, and  $\mathbf{1}(P)$  takes the value one when the predicate  $P$  is true and zero otherwise. By this definition, the observed behavior of MANIC users shows that  $P[k = 1] \simeq 0.75$ . This observation tells us that the most frequent student action is to *call for next slide* ( $k = 1$ ). The accuracy of prediction for action 1 is thus crucial to the quality of a prediction scheme.

A natural question to ask here is how to evaluate the quality of a prediction algorithm. Intuitively, since  $P[k = 1] \simeq 0.75$ , if we predict *all* of the students' actions as 1, we will have 75% chance to predict correctly across every observed output. However, although by this naive way, every 1 in the observed sequence will have been correctly predicted, every action in the observed sequences that is *not* 1 will have been predicted *wrong*! This introduces a tradeoff between accurately predicting action 1s and not predicting an action as 1 when the observed output is indeed not 1.

Based on the above argument, we define two random variables and a set of notation using these two random variables to evaluate the performance of an ODU algorithm. Let's define  $P_t$  as the random variable of a *predicted* output at time  $t$ , and recall that  $O_t$  denotes the random variable of the *observed* output at time  $t$ , of an HMM. Let us next define the following fractions,

1.  $F[P_t = 1 \mid O_t = 1]$ : the fraction of correctly predicted 1 actions;

$$\begin{aligned} F[P_t = 1 \mid O_t = 1] &= \frac{\text{\# of observed 1 actions that are also predicted as 1}}{\text{\# of observed 1 actions}} \\ &= \frac{\sum_{t=1}^{T_n} \mathbf{1}(O_t = 1, P_t = 1)}{\sum_{t=1}^{T_n} \mathbf{1}(O_t = 1)} \end{aligned} \quad (12)$$

2.  $F[O_t = 1 \mid P_t = 1]$ : the fraction of correctly matched (or hit) 1 predictions;

$$\begin{aligned} F[O_t = 1 \mid P_t = 1] &= \frac{\text{\# of 1 predictions that are also observed as 1}}{\text{\# of 1 predictions}} \\ &= \frac{\sum_{t=1}^{T_n} \mathbf{1}(P_t = 1, O_t = 1)}{\sum_{t=1}^{T_n} \mathbf{1}(P_t = 1)} \end{aligned} \quad (13)$$

3.  $F[P_t = 1 \mid O_t \neq 1]$ : the fraction of wrong predicted *non-1* actions;

$$\begin{aligned} F[P_t = 1 \mid O_t \neq 1] &= \frac{\text{\# of observed non-1 actions that are predicted as 1}}{\text{\# of observed non-1 actions}} \\ &= \frac{\sum_{t=1}^{T_n} \mathbf{1}(P_t = 1, O_t \neq 1)}{\sum_{t=1}^{T_n} \mathbf{1}(O_t \neq 1)} \end{aligned} \quad (14)$$

4.  $F[O_t = 1 \mid P_t \neq 1]$ : the fraction of wrong matched (or missed) *non-1* predictions;

$$\begin{aligned} F[O_t = 1 \mid P_t \neq 1] &= \frac{\text{\# of non-1 predictions that are observed as 1}}{\text{\# of non-1 predictions}} \\ &= \frac{\sum_{t=1}^{T_n} \mathbf{1}(O_t = 1, P_t \neq 1)}{\sum_{t=1}^{T_n} \mathbf{1}(P_t \neq 1)} \end{aligned} \quad (15)$$

The first two fractions,  $F[P_t = 1 \mid O_t = 1]$  and  $F[O_t = 1 \mid P_t = 1]$ , reflect the accuracy of predicting 1 actions. We would like these first two fractions to be as large as possible. On the other hand, the later two fractions,

$F[P_t = 1 \mid O_t \neq 1]$  and  $F[O_t = 1 \mid P_t \neq 1]$ , capture a measure of *prediction mistake*. When the observed action is not action 1 and we predict that it is (evaluated by  $F[P_t = 1 \mid O_t \neq 1]$ ), a prefetch that maybe un-needed will be executed. When we do not predict the next action as action 1 and observed action turns to be an action 1 (evaluated by  $F[O_t = 1 \mid P_t \neq 1]$ ), this leads to a *missing* prefetch. For these reason, we would like the last two conditional probabilities to be as small as possible. In addition to the above four fractions, we also define

5.  $F[Match]$ : the fraction of predictions that matche the observed action. For a sequence of predictions and actions, we estimate  $F[Match]$  as:

$$\begin{aligned}
 F[Match] &= \frac{\text{\# of actions that have correct predictions}}{\text{\# of actions in a sequence}} \\
 &= \frac{\sum_{t=1}^{T_n} \mathbf{1}(O_t = P_t)}{T_n}
 \end{aligned} \tag{16}$$

$F[Match]$  evaluates the overall prediction accuracy of a prediction algorithm. Actions 2, 3, and 4 are also considered as factors affecting the measurement of  $F[Match]$ . Obviously, a large value of  $F[Match]$  is preferred.

## 5.2 Experiment Results and Analysis of ODU Algorithms

For each access sequence of a student, a prediction algorithm, using any of our three ODU methods proposed in Section 4, can produce a file tracing each observed action and its corresponding predicted actions. An example file might look like:

```

4 4 3 2 1
1 4 3 2 1
3 4 3 2 1
3 3 4 1 2
1 1 3 4 2
1 1 3 4 2
4 4 1 3 2

```

where the integers are action symbols. In this trace file, each row represents one action of a student. Integers in the first column indicate the real (observed) actions that a student takes. Integers in the later four columns represent the prediction results in order of decreasing likelihood. For example, in the first row, the real action of a student is ‘4’ (start). Our prediction result shows that action ‘4’ has the highest probability of occurrence for that action

(the second integer in the first row is 4), action ‘3’ (index) has the next highest probability of occurrence, the probability of action ‘2’ (previous) occurring is less than both ‘4’ and ‘3’, and the probability of action ‘1’ occurring is the smallest. Our action prediction scheme always picks that action with the probability of occurrence, which is shown in the second column in this example file. Therefore, the prediction performance of an algorithm can be calculated by examining the first two columns of these trace files.

Using the action symbols listed in the first two columns of each trace file, in Figure 1, we show the performance of ODU1 with 4 hidden states over all of the students by evaluating the five performance measures defined in equation (12) to equation (16). On the y-axis, Figure 1 plots the values for multiple fractions,  $F[P_t = 1 | O_t = 1]$ ,  $F[O_t = 1 | P_t = 1]$ ,  $F[P_t = 1 | O_t \neq 1]$ ,  $F[O_t = 1 | P_t \neq 1]$  and  $F[Match]$  and the x-axis of Figure 1 indicates student (user) ID number.

From Figure 1, we observe that most of the students have fractions  $F[Match]$ ,  $F[P_t = 1 | O_t = 1]$  and  $F[O_t = 1 | P_t = 1]$  that are higher than 75% and fractions  $F[P_t = 1 | O_t \neq 1]$  and  $F[O_t = 1 | P_t \neq 1]$  that are lower than 35%. These results match our expectation for good prediction accuracy ( $F[Match]$ ,  $F[P_t = 1 | O_t = 1]$ ,  $F[O_t = 1 | P_t = 1]$ ), and low prediction mistakes ( $F[P_t = 1 | O_t \neq 1]$ ,  $F[O_t = 1 | P_t \neq 1]$ ). However, Figure 1 only provides an intuition of these performance results. We would like to further quantify the performance of an ODU algorithm. For doing that, we evaluate the number of students who have high accurate predictions, (fractions of  $F[Match]$ ,  $F[P_t = 1 | O_t = 1]$ , and  $F[O_t = 1 | P_t = 1]$  being greater or equal to 0.75), and the number of students who suffer low prediction mistakes (fractions of  $F[P_t = 1 | O_t \neq 1]$  and  $F[O_t = 1 | P_t \neq 1]$  being lower or equal to 0.35). Table 1 to Table 3 will show these results for each of the ODU algorithms shortly.

We also notice that although there were more than 200 students registered in MANIC system, not all of the students’ data can be really used. For instance, a student could possibly only register but not browse MANIC at all. Or the log file of a student may be so short that we removed it from consideration (our cutoff points for doing so was the logfiles with only one access session). Due to these considerations, the number of students with valid data that we can investigate is reduced to around 110. Note that, in Figure 1, some of  $F[O_t = 1 | P_t = 1]$  fractions are shown to have a negative value. In the case that there is no *call next slide action* being predicted for the actions of some students ( $\sum_{t=1}^{T_n} \mathbf{1}(P_t = 1) = 0$ ),  $F[O_t = 1 | P_t = 1]$  is set to a negative value ( $-0.02$ ). Similarly,  $\sum_{t=1}^{T_n} \mathbf{1}(O_t = 1)$ ,  $\sum_{t=1}^{T_n} \mathbf{1}(P_t \neq 1)$ , and  $\sum_{t=1}^{T_n} \mathbf{1}(O_t \neq 1)$  are all possibly being zero. Figure 1 omits those cases.

Recall that we define  $N$  as the number of states for a hidden Markov model. In Table 1, we show the prediction results for ODU1 for different value of  $N$ . Each row of Table 1 records the performance results corresponding

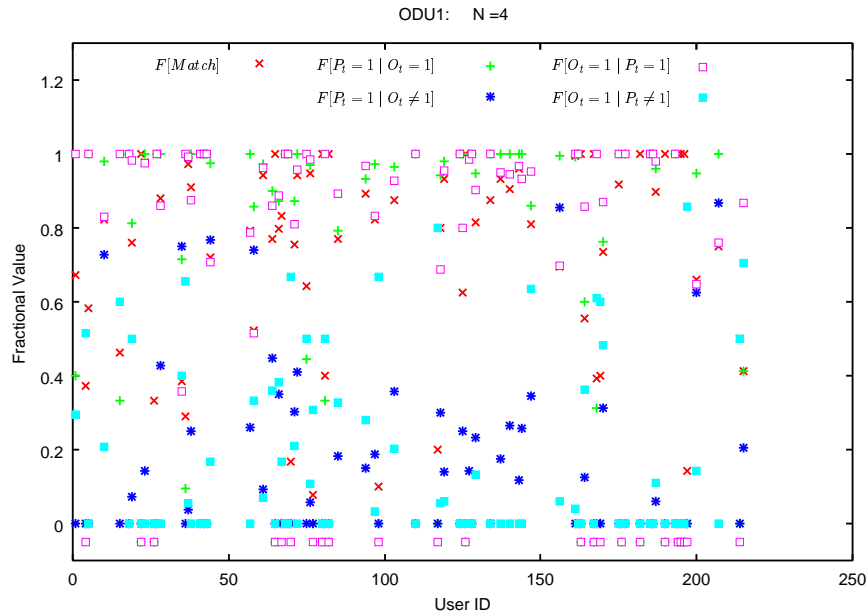


Figure 1: Fraction Distributions for ODU1,  $N = 4$

$N$	Number of Students				
	High Fractions ( $\geq 0.75$ )			Low Fractions ( $\leq 0.35$ )	
	$F[Match]$	$F[P_t = 1   O_t = 1]$	$F[O_t = 1   P_t = 1]$	$F[P_t = 1   O_t \neq 1]$	$F[O_t = 1   P_t \neq 1]$
3	60	54	58	66	63
4	60	53	56	70	67
5	61	54	56	71	68

Table 1: ODU1, Using Different  $N$

to an examined value of  $N$  for ODU1. We inspect the numbers of students, from whom individually we have a high prediction accuracy (with fractions of  $F[Match]$ ,  $F[P_t = 1 | O_t = 1]$ , and  $F[O_t = 1 | P_t = 1]$  being greater or equal to 0.75) and low prediction mistakes (with fractions of  $F[P_t = 1 | O_t \neq 1]$  and  $F[O_t = 1 | P_t \neq 1]$  being lower or equal to 0.35), that are provided by using an underlying hidden Markov model with  $N$  states. Intuitively, by changing the size,  $N$ , of a hidden Markov model, the performance of ODU1 should be different. However, we observe from Table 1 that, by varying the size of an underlying HMM, the numbers of students under different fraction constraints differ insignificantly for ODU1. In the mean time, although using HMMs with  $N = 4$  or  $N = 5$  has slightly better performance (more students satisfy the fraction constraints) than using a model with  $N = 3$ , the complexity of ODU1 is noticeably higher as the size of a model gets larger.

$\phi$	Number of students				
	High Fractions ( $\geq 0.75$ )			Low Fractions ( $\leq 0.35$ )	
	$F[Match]$	$F[P_t = 1   O_t = 1]$	$F[O_t = 1   P_t = 1]$	$F[P_t = 1   O_t \neq 1]$	$F[O_t = 1   P_t \neq 1]$
1	79	67	68	73	81
2	71	63	64	74	77
3	67	58	61	73	77
4	66	57	62	70	74

Table 2: ODU2,  $N = 4$ , Using Different  $\phi$

$w$	Number of students				
	High Fractions ( $\geq 0.75$ )			Low Fractions ( $\leq 0.35$ )	
	$F[Match]$	$F[P_t = 1   O_t = 1]$	$F[O_t = 1   P_t = 1]$	$F[P_t = 1   O_t \neq 1]$	$F[O_t = 1   P_t \neq 1]$
2	73	60	62	72	77
3	75	63	62	71	78
5	72	63	62	73	78
7	71	63	62	71	78

Table 3: ODU3,  $N = 4$ ,  $\phi = 2$ , Using Different  $w$

Let us next consider ODU2. Recall that  $\phi$  is defined as the *error threshold*, which triggers the HMM re-estimation process. When the number of mismatches between the prediction and the real action is higher than  $\phi$ , the parameters of an HMM are re-estimated under ODU2. In Table2, the performance of ODU2 is evaluated for varying  $\phi$ . We observe that the smaller  $\phi$  the larger the number of students satisfying the fraction constraints, but also the more complex the computation. Meanwhile, comparing the performance results of ODU1 and ODU2, we notice that ODU2 performs significantly better than ODU1.

In ODU3, rather than using the *error threshold*  $\phi$ , we define a sliding window of size  $w$ , which controls the proportion of history data being used to re-estimate a HMM. For a window size,  $w = 3$ , for example, only the last 3 sessions of a student are used in determining  $\lambda$  (By contrast, all of a student’s past sessions are used in ODU1 and ODU2). We examine the performance of ODU3 by varying  $w$ , and show the results in Table 3. The performance of ODU3 for different value of  $w$  is quite comparable. Compared to ODU2, ODU3 doesn’t have remarkable improvement too, though it performs significantly better than ODU1.

Thus far, we have examined the performance of three ODU algorithms by evaluating the experimental result

for each student individually and counting the number of students satisfying the performance fraction constraints. We have found that ODU2 and ODU3 have a higher number of students satisfying the performance fraction constraints. We also seen that smaller value of  $\phi$  provides better performance of ODU2 compared with ODU1. In addition to this evaluation method, we can also investigate the quality of an ODU algorithm by evaluating the accuracy of a prediction for all the actions taken by all the students. In the next Section, instead of studying prediction performance individually, we will focus on evaluating all of the actions taken by all students.

## 6 Evaluations Based on Action Selection Scheme 2

Recall that our goal for this project is to understand how students interact with MANIC, so that we can predictively prefetch the *next slides* for a student correctly. The meaning of *correctly* predicting and prefetching action 1 is twofold. For all of the *real* 1 actions, we should predict them as much as possible (a high probability  $P(P_t = 1|O_t = 1)$ ), so that a *next slide* can be prefetched whenever it is necessary. In the meantime, when we predict the next action as 1, we expect that the real action is more likely to be 1 (a high probability  $P(O_t = 1|P_t = 1)$ ), so that the prefetched *next slide* won't be useless. In the following Sections, we will call  $P(O_t = 1|P_t = 1)$  as *precision* and  $P(P_t = 1|O_t = 1)$  as *recall* [7]<sup>1</sup>.

Action selection scheme 1 that is used in Section 5 selects the action that has the highest probability of occurrence,  $\arg \max_k \{P(O_t = k)\}$ , as the prediction for an observed output. We have investigated the correctness of predicting action 1 by that action selection scheme 1 for all of the students' actions and found 0.892 precision as well as 0.899 recall (by using ODU1 with 4 hidden states). These results show that action selection scheme 1 provides high accuracy of predicting action 1. *Can the prediction correctness for action 1 be better?*

Based on our observation, for many *real* 1 actions, the HMM does not predict action 1 as the most likely action, i.e., the action with the highest rank. This situation could be caused by the use of an HMM,  $\lambda$ , that does not capture a student's behavior quite well. To have more 1 actions predicted correctly, we propose a new method, action selection scheme 2, which does not consider the rank of an action symbol's occurring probability to determine the prediction. In action selection scheme 2, a threshold  $\delta$  is set for controlling the prediction of action 1. Specifically, if  $P(O_t = 1)$  is greater or equal to  $\delta$ , 1 will be predicted; for all of other cases, it simply predicts the output as a *non-1* action. We examine the *precision* and *recall* of action selection scheme 2 and compare the result with that of action selection scheme 1 below.

---

<sup>1</sup>Precision is defined as *found correct/found total* and recall is defined as *found correct/known correct* in [7]

## 6.1 Evaluating Action Selection Scheme 2

For evaluating the performance of using a fixed threshold,  $\delta$ , to make the prediction decision, we would like to know the *mistake* that a prediction could possibly make by using this action selection scheme.

We define random variable  $X_t$  to be the predicted probability of occurrence of action 1 at time  $t$ , and define  $\delta$  as a threshold value for  $P(O_t = 1)$  and modify our prediction schemes so that action 1 is predicted only when the  $P(O_t = 1)$  is greater or equal to  $\delta$ ; otherwise the next output is predicted not to be 1. Thus we have,

- $P[O_t \neq 1 | X_t \geq \delta]$ : represents the empirical fraction of wrong predictions given we predict the action to be 1 using threshold  $\delta$ ,

$$P[O_t \neq 1 | X_t \geq \delta] = \frac{\sum_{n=1}^{\tilde{N}} \sum_{t=1}^{T_n} \mathbf{1}(O_t \neq 1, X_t \geq \delta)}{\sum_{n=1}^{\tilde{N}} \sum_{t=1}^{T_n} \mathbf{1}(X_t \geq \delta)} \quad (17)$$

In this case, *precision* can be calculated by  $1 - P[O_t \neq 1 | X_t \geq \delta]$ .

- $P[X_t < \delta | O_t = 1]$ : represents the empirical fraction of real 1 actions that we miss to predict using threshold  $\delta$ ,

$$P[X_t < \delta | O_t = 1] = \frac{\sum_{n=1}^{\tilde{N}} \sum_{t=1}^{T_n} \mathbf{1}(O_t = 1, X_t < \delta)}{\sum_{n=1}^{\tilde{N}} \sum_{t=1}^{T_n} \mathbf{1}(O_t = 1)} \quad (18)$$

In this case, *recall* can be calculated by  $1 - P[X_t < \delta | O_t = 1]$ .

By varying  $\delta$ , we hope to minimize  $P[O_t \neq 1 | X_t \geq \delta]$  and  $P[X_t < \delta | O_t = 1]$ . Based on ODU1, using 4 hidden states, we evaluated the trend of these two conditional fractions as a function of  $\delta$ , and show them in Figure 2.

In Figure 2, the x-axis corresponds to threshold  $\delta$ , and the y-axis plots  $P[O_t \neq 1 | X_t \geq \delta]$  and  $P[X_t < \delta | O_t = 1]$ . The decreasing curve represents  $P[O_t \neq 1 | X_t \geq \delta]$ , and the increasing curve represents  $P[X_t < \delta | O_t = 1]$ . Figure 2 can be used to determine a reasonable threshold  $\delta$  for controlling the prediction. For instance, if we want high precision, then  $\delta$  should be set to a large value. For example, assume  $\delta = 0.8$ . In this case, only those 1 actions with an occurring probability higher than or equal to 0.8 are predicted as the next output action. In this case, from Figure 2, we notice that the precision is 0.92 ( $P[O_t \neq 1 | X_t \geq 0.8] = 0.08$ ). However, one should also notice the tradeoff that the recall is only 0.77 ( $P[X_t < 0.8 | O_t = 1] = 0.23$ ). On the other hand, if we want a high recall and can tolerate a low precision, then a low value of  $\delta$  should be used. We can see from Figure 2 that, when

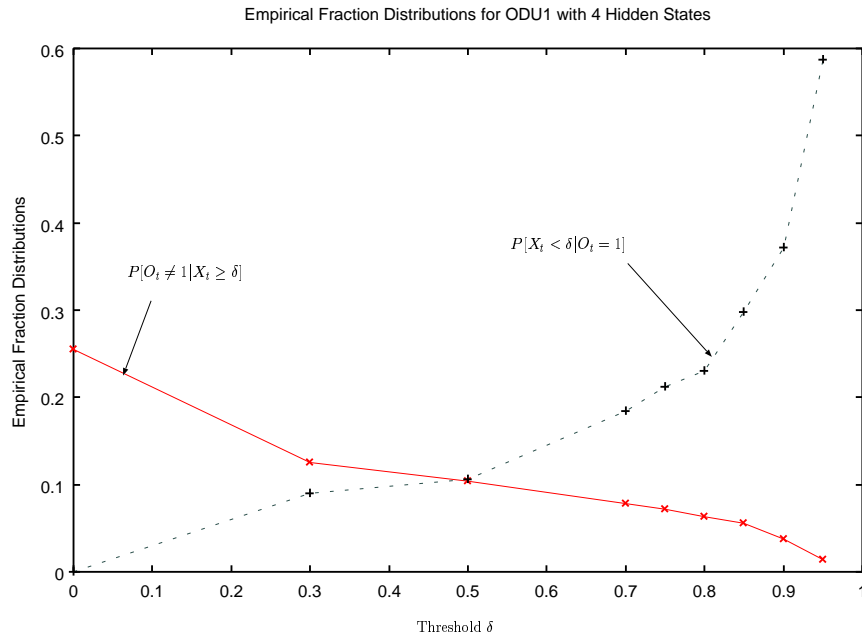


Figure 2: Probability Distributions for  $P[O_t \neq 1 | X_t \geq \delta]$  and  $P[X_t < \delta | O_t = 1]$  of ODU1 with  $N = 4$

$\delta$  is set to 0.3, there are only around 3.5% 1 actions being missed to be predicted, the recall is 0.965. Still, in this threshold, a prediction for action 1 is wrong 13% of the time, which provides 0.87 precision.

A more important observation from Figure 2 is that, when  $\delta$  is set to 0.3, although the precision of action selection scheme 2 is slightly lower than that of action selection scheme 1, the 0.965 recall of action selection scheme 2 is significantly higher than the 0.899 recall of action selection scheme 1. This tells us that the prediction correctness for action 1 can be improved by using action selection scheme 2.

## 6.2 Performance Evaluation of ODU Algorithms Using Action Selection Scheme 2

As discussed in the previous section, there is a tradeoff between precision and recall. The evaluation method shown in Figure 2 can provide a solution for how to set an appropriate value of  $\delta$  given different performance requirement. We can also use this approach to compare the performance of ODU algorithms.

Figure 3 shows the value of  $P[O_t \neq 1 | X_t \geq \delta]$  over different values of  $\delta$  for ODU1 with 4 hidden states, ODU2 with *error threshold*  $\phi = 2$ , and ODU3 with a sliding window of size  $w = 3$ . From this graph, we observe that ODU2 performs significantly better than ODU1 and provides the smallest value of  $P[O_t \neq 1 | X_t \geq \delta]$  among the

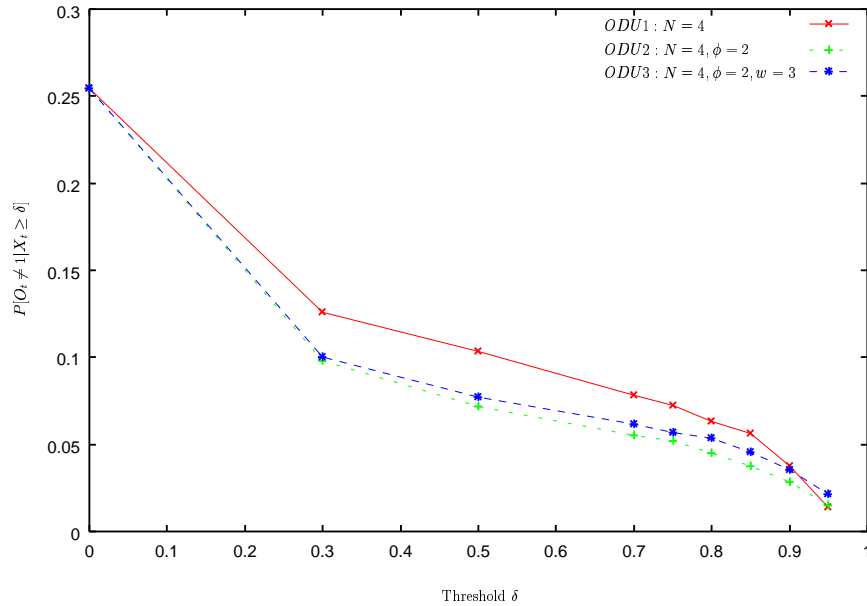


Figure 3: Probability Distribution of  $P[O_t \neq 1 | X_t \geq \delta]$  Comparing Three ODU Algorithms

three algorithms. ODU3, again, doesn't provide a remarkable improvement of the performance. These results are compatible with the evaluation results in Section 5.

Meanwhile, Figure 4 shows the value of  $P[X_t < \delta | O_t = 1]$  over different values of  $\delta$  also for ODU1 with 4 hidden states, ODU2 with *error threshold*  $\phi = 2$ , and ODU3 with a sliding window of size  $w = 3$ . Again, ODU2 performs significantly better than both ODU1 and ODU3, it also provides the smallest value for  $P[X_t < \delta | O_t = 1]$  among these three algorithms. In Figure 4, the performance of ODU3 significantly distinguishes from it of ODU1 or ODU2. However, although the performance of ODU3 is better than the performance of ODU1, it is still not as good as the performance of ODU2.

Base on above results, we conclude that ODU2 has the best performance.

## 7 Conclusions and Discussions

In this project, we studied the underlying concepts of Hidden Markov Models, using Rabiner's notation of the HMM. The log files of students, who used MANIC system, were translated into sequences of output symbols. For each student, an HMM was developed to capture his/her browsing behavior. The sequences of action symbols

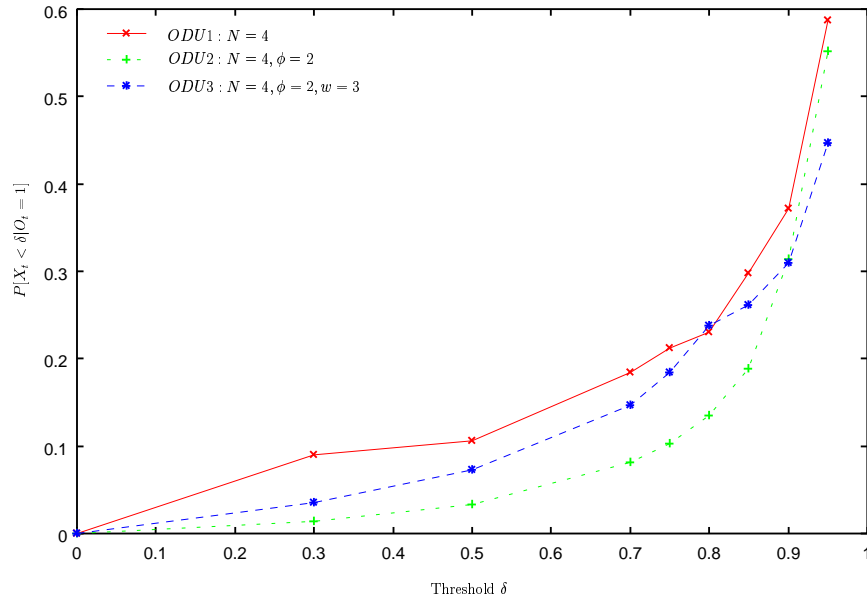


Figure 4: Probability Distribution of  $P[X_t < \delta | O_t = 1]$  Comparing Three ODU Algorithms

were then used as the output of HMMs. We focussed our study on predicting the students' *call the next slide* action. Three ODU algorithms were proposed, implemented and evaluated. Based on the occurring probabilities of different actions, we also proposed two schemes for determining the prediction: selecting the action with the highest occurring probability, or using threshold  $\delta$  to determine the prediction. Based on all of the experiment results, ODU2 shows the best performance, and action selection scheme 2 provides better prediction correctness for predicting action 1.

It's worth mentioning that we also tried a third action selection scheme that choose the action as the next output using the probability density function of the output symbol ( $f_K(k)$ , where  $K$  is the random variable defined as the output action, and  $k \in V$ ). Unfortunately, this scheme shows varying prediction results. A student's behavior may be predicted very well once, but the same sequence of behavior might be poorly predicted for some other time. Due to this observation, we didn't explore this action selection scheme any further. In the meantime, we have shown that when a HMM was perfect (i.e., the probability that action  $k$  occurs, as predicted by the HMM, always equals to the actual probability of the appearance of action  $k$ ), the scheme of using  $\arg \max_k \{P(O_t = k)\}$  should be at least as good as the third action selection scheme. However when a HMM does not exactly capture a user's behavior, the third action selection scheme would possibly provide better prediction results. We will provide a proof and discussion in the Appendix.

Also we find that, in the real case, the students' behavior is quite different. An ODU algorithm might fit to most of students very well but does not capture some students' behavior. We say that ODU2 has the best performance based on the evaluation results for all of the students. Nonetheless, we also notice that for a few of students ODU1 works the best, for a few of other students ODU3 works the best.

In a summary, a Hidden Markov Model can be used to study a student's browsing behavior for an on-line tutoring system such as MANIC. It captures the study preference of most of the students and provides good prediction results.

## References

- [1] Joseph Beck. *ADVISOR: A Machine-Learning Architecture for Intelligent Tutor Construction*. PhD thesis, Umass at Amherst, 2001.
- [2] E. Cohen, B. Krishnamurthy, and J. Rexford. Efficient algorithms for predicting requests to web servers. In *IEEE INFOCOM Conference*, 1999.
- [3] Pablo R. de Buen, Sunil Vadera, and Eduardo F. Morales. Machine learning in lacepro multi-functional framework. In *Sixth International Conference on User Modeling*, June 1997.
- [4] Frederick Jelinek. *Statistical Methods for Speech Recognition (Language, Speech, and Communication)*, chapter 2. MIT Press, Jan 1999.
- [5] Tapas Kanungo. [www.cfar.umd.edu/~kanungo](http://www.cfar.umd.edu/~kanungo).
- [6] Terran Lane. Hidden markov models for human/computer interface modeling. In *IJCAI-99 Workshop on Learning About Users*, pages 35–44., 1999.
- [7] Leah S. Larkey, Paul Ogilvie, M. Andrew Price, and Brenden Tamilio. Acrophile: an automated acronym extractor and server. In *ACM DL*, pages 205–214, 2000.
- [8] Iain L. MacDonald and Walter Zucchini. *Hidden Markov and Other Models for Discrete-valued Time Series*. Chapman & Hall, 1997.
- [9] Jitendra Padhye and Jim Kurose. An empirical study of client interactions with a continuous-media courseware server. In *NASSDAV*, 1998.
- [10] M. Quafafou, A. Mekaouche, and N. H.S. Multiviews learning and intelligent tutoring systems. In *Seventh World Conference on Artificial Intelligence in Education*, 1995.
- [11] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 37(2):257–286, Feb 1989.
- [12] Ramesh R. Sarukkai. Link prediction and path analysis using Markov chains. *COMP-NET-AMSTERDAM*, 33(1–6):377–386, June 2000.
- [13] S.Schechter, M. Krishnan, and M.D.Smith. Using path profiles to predict http requests. In *7th International World Wide Web Conference*, April 1998.
- [14] Mia Stern and Beverly P. Wolf. Adaptive content in an online lecture system. In *International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, 2000.
- [15] X.D.Huang, Y.Ariki, and M.A.Jack. *Hidden Markov Models for Speech Recognition*. Edinburgh university Press, 1990.

## Appendix: Comparison of Action Selection Scheme 1 and Action Selection Scheme 3

*Action Selection Scheme 1 (Strategy 1)*: For predicting each observation action,  $\arg \max_k \{P(O_t = k)\}$  is chosen as the prediction.

*Action Selection Scheme 3 (Strategy 3)*: For predicting each observation action, based on the probability density function of the output symbol of an HMM ( $f_K(k)$ , where  $K$  is the random variable defined as the output action, and  $k \in V$ ), a biased coin is tossed. The action shown by the coin is chosen as the prediction.

**Claim:** When an HMM was perfect (i.e., the probability that action  $k$  occurs, as predicted by the HMM, always equals to the actual probability of the appearance of action  $k$ ), Strategy 1 should be at least as good as Strategy 3.

**Define:**

- $O_t$ , the random variable of the observed action at time  $t$ ;
- $O'_t$ , the random variable of the output action of a HMM model at time  $t$ ;
- $P_t$ , the random variable of the predicted action at time  $t$ ;
- $V$ , the set of action symbols;

**Proof:**

1. For Strategy 1, the expected probability of the predicted action being the same as the observed action,  $E_1[P(O_t = P_t)]$ , is

$$\begin{aligned}
 E_1[P(O_t = P_t)] &= \sum_{i \in V} P(O_t = P_t | O_t = i) \cdot P(O_t = i) \\
 &= \sum_{i \in V} P(P_t = i | O_t = i) \cdot P(O_t = i)
 \end{aligned} \tag{19}$$

Base on Strategy 1,

$$P(P_t = i | O_t = i) = \begin{cases} 1, & i = \arg \max_k \{P(O'_t = k)\} \\ 0, & otherwise \end{cases} \tag{20}$$

Substitute equation (20) into equation (19),

$$\begin{aligned} E_1[P(O_t = P_t)] &= 1 \cdot P(O_t = \arg \max_k \{P(O'_t = k)\}) \\ &= P(O_t = \arg \max_k \{P(O'_t = k)\}), \quad k \in V \end{aligned} \quad (21)$$

2. For Strategy 3, the expected probability of the predicted action being the same as the observed action,  $E_3[P(O_t = P_t)]$ , is

$$\begin{aligned} E_3[P(O_t = P_t)] &= \sum_{i \in V} P(O_t = P_t | O_t = i) \cdot P(O_t = i) \\ &= \sum_{i \in V} P(P_t = i | O_t = i) \cdot P(O_t = i) \end{aligned}$$

Base on Strategy 3,

$$P(P_t = i | O_t = i) = P(O'_t = i) \quad (22)$$

$\implies$

$$\begin{aligned} E_3[P(O_t = P_t)] &= \sum_{i \in V} P(O'_t = i) \cdot P(O_t = i) \\ &\leq \sum_{i \in V} P(O_t = i) \cdot P(O'_t = \arg \max_k \{P(O'_t = k)\}) \\ &= \left( \sum_{i \in V} P(O_t = i) \right) \cdot P(O'_t = \arg \max_k \{P(O'_t = k)\}) \\ &= P(O'_t = \arg \max_k \{P(O'_t = k)\}) \end{aligned} \quad (23)$$

$\implies$

$$E_3[P(O_t = P_t)] \leq P(O'_t = \arg \max_k \{P(O'_t = k)\}), \quad k \in V \quad (24)$$

3. From equation (21) and inequality (24), we found that when  $O_t = O'_t$ ,  $E_3[P(O_t = P_t)] \leq E_1[P(O_t = P_t)]$ , which tells us that when an HMM is *perfect*, Strategy 1 is at least as good as Strategy 3.

However when an HMM does not exactly capture a student's behavior,  $O_t \neq O'_t$ , Strategy 3 could possibly provide better prediction result.

**Example for  $O_t \neq O'_t$ :** Assume there are 4 action symbols in  $V$ . We show the probability of occurrence of each  $i \in V$  that is provide by an HMM,  $P(O'_t = i)$ , and the observed probability of occurrence for each action,  $P(O_t = i)$ , in the table below.

Action Symbol ( $i$ )	$P(O'_t = i)$	$P(O_t = i)$
1	0.5	0.4
2	0.45	0.45
3	0.02	0.08
4	0.03	0.07

From Strategy 1, we have

$$\begin{aligned}
 E_1[P(O_t = P_t)] &= P(O_t = \arg \max_k \{P(O'_t = k)\}) \\
 &= P(O_t = 1) \\
 &= 0.4
 \end{aligned}$$

From Strategy 3, we have

$$\begin{aligned}
 E_3[P(O_t = P_t)] &= \sum_{i \in V} P(O'_t = i) \cdot P(O_t = i) \\
 &= 0.5 \cdot 0.4 + 0.45 \cdot 0.45 + 0.02 \cdot 0.08 + 0.03 \cdot 0.07 \\
 &= 0.4062 \\
 &> 0.4
 \end{aligned}$$