

Performance Evaluation with Hidden Markov Models

E. De Souza E Silva, R. Leão, Richard R. Muntz

▶ **To cite this version:**

E. De Souza E Silva, R. Leão, Richard R. Muntz. Performance Evaluation with Hidden Markov Models. Performance Evaluation of Computer and Communication Systems (PERFORM), Oct 2010, Vienna, Austria. pp.112-128, 10.1007/978-3-642-25575-5_10 . hal-01586903

HAL Id: hal-01586903

https://inria.hal.science/hal-01586903

Submitted on 13 Sep 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Performance evaluation with Hidden Markov Models *

E.de Souza e Silva, R.M.M. Leão
Federal Univ. of Rio de Janeiro - COPPE/PESC
P.O Box 68511 . RJ 21941-972 . Brazil
{edmundo,rosam}@land.ufrj.br

Richard R. Muntz
UCLA - CS Departament
4732 Boelter Hall . LA . CA 90024 . USA
muntz@cs.ucla.edu

Abstract. The use of hidden Markov models (HMMs) has found widespread use in many different areas. This chapter focuses on HMMs applied to the performance evaluation of computer systems and networks. After presenting a brief review of background material on HMMs, applications such as channel delay and loss characteristics, traffic modeling and workload generation are surveyed. The power of HMMs as predictors of performance metrics is also highlighted. We conclude by presenting a few features of the module of the Tangram-II performance evaluation tool that is targeted to HMMs.

Keywords: hidden Markov models, performance evaluation, network applications

1 Introduction

Hidden Markov models (HMM) have been used in a myriad of applications that include speech recognition, signal processing, artificial intelligence, computational biology, finance, image processing, and medical diagnosis, to name a few. Reference [2] provides a comprehensive bibliography of over 300 articles on these applications. The first major successful application of HMMs was in speech recognition [13] and was followed by application in a widespread set of application areas. The HMM framework provides a rich theoretical basis that can be adapted to many different applications. Despite the large body of literature on HMM applications, its use in performance evaluation and computer communication is relatively new. The introduction of the application of HMMs in performance modeling motivates the current chapter.

In performance modeling, the analyst usually has a clear understanding of how the system to be analyzed works. If a Markovian model is the paradigm of choice, the main task is to select the appropriate system state variables and the range of possible values for each. For instance, if the system can be modeled by a single server queue and the goal is to predict the waiting time for a given arrival rate, the state variable of choice is the number of customers queued for service. Other state variables may be introduced, for instance, to represent a phase-type service time distribution (instead of the exponential distribution) or to represent changes in the mean arrival rate with time.

Another kind of model that falls in this category is availability modeling. In this case, state variables represent components that can be operational or in different modes of failure. Among the questions one may answer is the fraction of time the system is available to the user and the probability that the system fails in a given time interval.

Usually, models like a simple queueing system or dependability model, are parameterized from some prior knowledge of the system behavior. A key point to have in mind is that, in these models, one takes the point of view that the system state is directly observable. On the contrary, with HMM the system state of the underlying process is assumed to *not* be directly observable. Rather one can observe some values that are a probabilistic function of the state of the underlying Markov process. (Thus the origin of the name *Hidden* Markov Model.)

* This work is supported in part by grants from CNPq, NSF and FAPERJ.

To clarify these ideas, consider the following simple example. Suppose that one is interested in predicting the behavior of a customer who is browsing through the web pages of an online book store. Assume that we would like to determine the probability that a specific customer is ready to order an item based on their past behavior within their session. Suppose that, after studying the behavior of many customers, the analyst decides to build a simple 4-state Markovian model. Assume that the customer's states are chosen as follows: *just browsing* (JB), searching for items in the store, *interested in a product* (IP) and *ready to order an item* (RO). For convenience we also introduce a *leaving* (LV) state to the model to indicate that a customer can leave after being in any of the previous states. (A transition to state LV to corresponds to a session ending and a new session starts in this specific example with a transition to state JB.)

To estimate the transition probabilities, we might for example, follow the trajectory of a set of customers, where the customers indicate along the way their current state of intent. Then, with this training data, one might estimate the state transition probabilities from this example data using the relative frequency of transitions from the chosen states. Figure 1(a) illustrates one possible Markov chain and its transition probabilities.

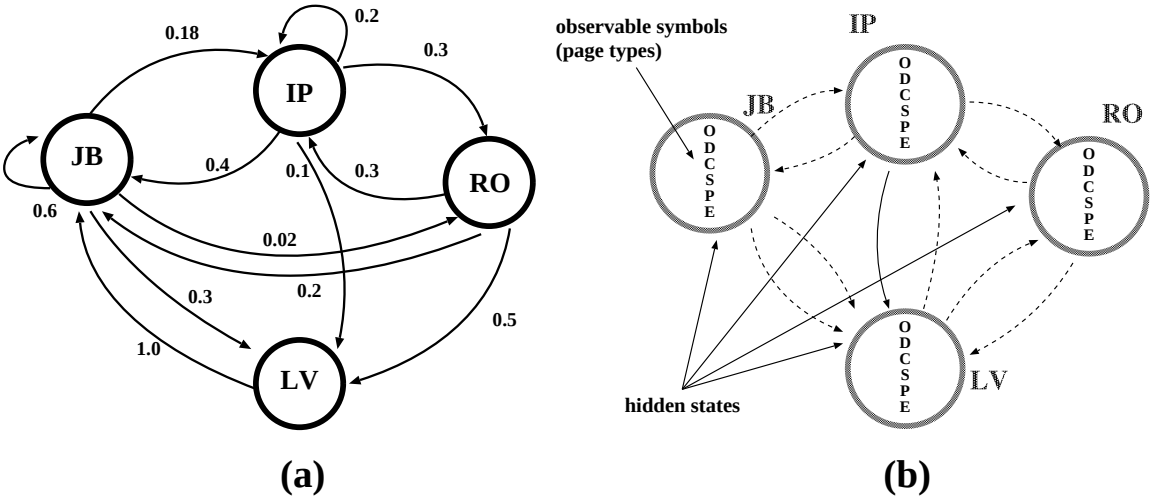


Fig. 1. An example of (a) a simple Markovian model of customer behavior; (b) an application example of HMMs.

Since we assume that a state change occurs with each page click, one might ask the following question: what is the probability that a customer is in the ready to order state after visiting a particular sequence of pages? This might be used, for example, to provide the user with some specific promotion information. Clearly, the efficacy of the model depends on its accuracy to answer the questions such as this.

In the above model it is assumed that we have access to data that includes the *intention* of a customer as each page is visited. If available, this can be used to estimate transition probabilities between states and also the conditional probabilities for the page type given the state of user intent. However this state information may only be available under special circumstances such as when collecting training data. After model construction, when applying the model to online customers, the user's state of intent is not directly observable. An even more challenging circumstance in building a HMM would be when the training data only includes the type of pages

a customer visits and not their state of intent. For example, assume that there are 7 different types of pages in the web site: product overview (O), product details (D), set of products within a category (C), shopping cart (S), purchase (P), exit (E). It is easy to record the type of pages the user clicks on, but we cannot know the customer's intent directly. As a consequence, we cannot directly observe the state of the customer's intent (JB, IP, RO). But it is intuitive that the states are correlated with the sequence of observations. In other words, the Markov states JB, IP, RO are *hidden* in the training data but the sequence of page types being visited is known and provides some evidence of of what the user's state of intent is.

Assume that, somehow, we are able to determine the transition probabilities of the hidden chain and, for each hidden state, the conditional probability of emitting an observable *symbol*. (In this example, the probability of clicking on a particular page type (O, D, etc.). Figure 1(b) shows the hidden states and observable symbols of the model. A typical question to be asked is what is the most probable hidden state (that is, the customer intent) given the observed sequence of pages visited up to the current time.

However, as mentioned above, in general we do not know a priori the transition probabilities of the hidden chain. In fact, in most problems, we do not even know how many states the hidden chain contains. (Imagine that you not only cannot perform an experiment, as described earlier, where a set of customers voluntarily provide the information of their intent but you may not even a priori know the number of or interpretation of the "states of intent".) In this case, you know little about the hidden MC and can only record the sequence of page clicks for a number of sessions. But to answer the question of the previous paragraph we must first build a completely specified hidden MC (including all transition probabilities) and also the probability of clicking a page of each type conditioned on the current state of the hidden MC. The only information available may be the sequence of page clicks. In this case, the problem is to find the unknown model parameters that maximize the probability that the observed sequence is generated by the model. (In general this will depend on (a) an a priori probability distribution over the space of possible models and (b) the conditional probability of the observed sequence(s) given a specific model.)

Now suppose that many of the customers of the book store are registered and they provide their age upon registration. Suppose you divide the customers into 2 groups: young and mature folks. Then you build, as above, two different models optimized for each type of customer. (Here one assumes that data to build the models consists of observation sequences where one knows the type of the customer (young or mature) for each sequence.) After building the two models from such example sequences, a third typical type of question is, given a specific user session and the sequence of page clicks observed (but not the customer type), what is the probability that the customer is young versus mature? This is equivalent to asking which of the two models better describes the sequence of page clicks observed.

It should be clear from the above examples that there are different questions that the analyst can ask from a HMM as compared to the traditional Markov models. In this chapter we intend to provide a few examples where HMMs have been used in performance evaluation (including computer communication). After presenting, in section 2, the notation used and a brief review of introductory material to provide the mathematical foundation for the subsequent sections, in section 3 we survey a few models that have been proposed in the literature. In section 4 we outline the main features of the Tangram-II module specifically tailored to deal with HMMs and in section 5 we conclude the chapter.

2 Preliminaries

In this section we first summarize the notation to be used in the rest of this chapter. In the previous section we described a typical type of application of HMMs and the types of question that the model may be used to answer in the context of that application. Here we summarize, independent of a particular application, a more formal definition of a HMM and the most common categories of problems across applications. Later sections will discuss particular applications in the context of performance evaluation.

2.1 Notation the Main HMM Problems

Below we introduce the notation for the parameters of a HMM.

1. The number of states, N, in the underlying (hidden) Markov chain. The i-th state is denoted by s_i. (In the example of section 1 $N = 4$.)
2. The (hidden) state at time t is denoted by q_t. So, the sequence of states up to time T is $q_1 q_2 \dots q_T$.
3. The number M, of distinct observations. The observed values are denoted by v_k for $1 \leq j \leq M$. ($M = 6$ in the example of section 1.)
4. The observed value at time j is denoted O_j. So, the sequence of observations up to time T is $O_1 O_2, \dots O_T$. A shorthand notation for the sequence of observations up to time T is $\mathcal{O}_T$.
5. The state transition probabilities: $\mathbf{P} = [p_{i,j}]$, where $p_{i,j} = P[q_{t+1} = s_j | q_t = s_i]$ for all t.
6. The conditional distribution for observed value conditioned on the state of the underlying MC: $b_j(k) = P[O_t = v_k | q_t = s_j]$. $\mathcal{B} = \{b_j(k)\}$ denotes the set of all such conditional probability distributions.
7. $\boldsymbol{\pi}^{(1)}$ will denote the initial state probability vector, i.e., $\boldsymbol{\pi}^{(1)} = [\pi_1^{(1)}, \pi_2^{(1)}, \dots, \pi_N^{(1)}]$ where $\pi_i^{(1)} = P[q_1 = s_i]$.
8. The model $\mathcal{M}$ is defined by the combination of $\boldsymbol{\pi}$, $\mathbf{P}$ and $\mathcal{B}$, and we use the notation $\mathcal{M} = (\mathbf{P}, \mathcal{B}, \mathbf{p})$ to indicate the complete set of model parameters.

Suppose that know the parameters of a HMM, for instance, the parameters for the HMM of Figure 1(b) that models the page types accessed by an user. Assume that we want to generate a sequence of page types visited by an user. From the model, a sequence of observable symbols can be easily generated as follows.

1. Choose an initial state according to $\pi^{(0)}$.
2. Set $t = 1$.
3. Choose $O_t = v_k$ according to the conditional probability distribution for state q_t, i.e., if $q_t = s_i$, choose v_k with probability $b_i(k)$.
4. Choose the next state according to the state transition probabilities.
5. Set $t = t + 1$ if $t < T$, else terminate.

In section 1 we describe an example used to introduce typical problems the analyst is faced with. For instance, if the model $\mathcal{M}$ is known, from a given (observable) sequence of page accesses $\mathcal{O}_T$, what is the probability that the model is in each of the hidden states at T? In addition, how to estimate the model parameters $(\mathbf{P}, \mathcal{B}, \boldsymbol{\pi})$ if the only information available are sequences of page accesses? Formally, we can identify four basic problems:

Problem 1:
Given the observation sequence $\mathcal{O}_T = O_1 O_2 \dots O_T$ and a model $\mathcal{M}$, compute the probability of observing the output sequence $\mathcal{O}_T$ given the underlying model $\mathcal{M}$, i.e., $P[\mathcal{O}_T | \mathcal{M}]$.

Problem 2:
Given the observation sequence $\mathcal{O}_T = O_1 O_2 \ldots O_T$ and a model $\mathcal{M}$, how to determine a *best* state sequence $\mathcal{Q}_T = q_1, q_2, \ldots, q_T$ which *best explains* the output sequence $\mathcal{O}_T$. In order words, in this problem we want to unveil the hidden states in the model. To have a well specified problem here we need to define more formally what is meant by *best*, i.e., define the specific objective function. Two possibilities are: (i) for each time t what is the most probable state of the underlying MC, and (ii) what is the most probable sequence of states. Note that these are different in general. For example, the answer to (i) could have $s_t = s_i$ and $s_{t+1} = s_j$ but if $p_{i,j}$ is 0 the sequence of states $\langle s_i, s_j \rangle$ could never occur.

Returning to the example of section 1, if the transition JB to RO is zero (instead of 0.02 as in Figure 1) the sequence *just browsing* to *ready to order* could never occur, but one can calculate the probability that the customer is *ready to order* after observing a sequence of page visits.

Problem 3:
Given the observation sequence $\mathcal{O}_T = O_1 O_2 \ldots O_T$, construct an underlying model $\mathcal{M}$, such that $P[\mathcal{O}_T|\mathcal{M}]$ is maximized.

Problem 4:
Given several possible models $\mathcal{M}_i \;\; 1 \leq i \leq K$ and a sequence of observed values, what is the probability that $\mathcal{M}_j$ is the actual model, i.e. $P[\mathcal{M}_j|\mathcal{O}]$. In the example of customers browsing an online bookstore, the two models could be one for a young customer and one for a mature customer.

2.2 A brief discussion of algorithms for solving these basic types of problems.

Problem 1. P$[\mathcal{O}_T|\mathcal{M}]$:

An iterative algorithm exists for computing $P[\mathcal{O}_T|\mathcal{M}]$ which has complexity N^2T [13].

Define $\alpha_t(\mathcal{O}_t, q_t = s_i|\mathcal{M}) = P[\mathcal{O}_t, q_t = s_i|\mathcal{M}]$ where $\mathcal{O}_t = O_1 O_2 \ldots O_t$, i.e., the first t observed values. Thus $\alpha_t(\mathcal{O}_t, q_t = s_i)$ is the probability of the first t observations **and** ending up in state s_i at time t. For convenience in notation we may sometimes drop the explicit reference to the model $\mathcal{M}$ but it is assumed unless otherwise stated. So we use the shorthand, $\alpha_t(\mathcal{O}_t, q_t = s_i)$ and sometimes even $\alpha_t(\mathcal{O}_t, s_i)$. Then: $\alpha_{t+1}(\mathcal{O}_{t+1}, q_{t+1} = s_j|\mathcal{M}) = \sum_{i=1}^{N} \alpha_t(\mathcal{O}_t, q_t = s_i|\mathcal{M}) p_{i,j} b_j(O_{t+1})$

Starting with $\alpha_1(O_1, s_i) = \boldsymbol{\pi}^{(0)}[s_i] b_i(O_1)$, the above expression is used to calculate the values of α at time $t+1$ from the values for α at time t. It should be clear that:

$$P[\mathcal{O}_T|\mathcal{M}] = \sum_{i=1}^{N} \alpha(\mathcal{O}_T, q_T = s_i|\mathcal{M})$$

The computational complexity for each time increment is clearly N^2 since there are N values to calculate at each increment and calculating each value takes on order N calculations.

Similarly an iterative backward algorithm exists using an auxiliary function $\beta_t(\mathcal{O}_{t+1,T}, q_t = s_i|\mathcal{M}) = P[\mathcal{O}_{t+1,T}|q_t = s_i, \mathcal{M}]$ where $\mathcal{O}_{t+1,T} = O_{t+1} O_{t+2} \ldots O_T$, i.e., the observed values from $t+1$ to T. (Again, see [13] for more details.)

Problem 2.
There is an algorithm for a specific version of Problem 2: computing $P[q_T = s_i|\mathcal{M}, \mathcal{O}]$. (Note that, for convenience, the notation here has dropped the subscript T from $\mathcal{O}_T$.)

From the solution to Problem 1 we know how to compute $P[\mathcal{O}, q_T = s_i|\mathcal{M}]$ and since $P[\mathcal{O}|\mathcal{M}] = \sum_{\text{states},i} P[\mathcal{O}, q_T = s_i|\mathcal{M}]$ we have an expression for $P[\mathcal{O}|\mathcal{M}]$ as well. Since:

$$P[q_T = s_i|\mathcal{O}, \mathcal{M}] = \frac{P[q_T = s_i, \mathcal{O}|\mathcal{M}]}{P[\mathcal{O}|\mathcal{M}]} = \frac{P[q_T = s_i, \mathcal{O}|\mathcal{M}]}{\sum_{\text{states},s_i} P[\mathcal{O}, q_T = s_i|\mathcal{M}]}$$

In terms of the $\alpha_T(\mathcal{O}_T, s_i)$ functions that were introduced earlier, we have:

$$P[q_T = s_i|\mathcal{O}, \mathcal{M}] = \frac{\alpha_T(\mathcal{O}, s_i)}{\sum_{\text{states},s_j} \alpha_T(\mathcal{O}, s_j)}$$

Problem 3.
One important problem is to determine the model parameters to fit the observed data. In other words, how to estimate the model parameters in order to maximize the probability of the observation sequence. Here we assume that the number of states in the underlying MC (N) is known. Roughly, the procedure for adjusting the model is based on the *maximum likelihood estimation* (MLE) method and a technique derived from the *Expectation-Maximization* (EM) algorithm known as Baum-Welch iterative procedure to obtain a local maximum for the likelihood function [18]. For HMMs, the Expectation step evaluates an auxiliary function (based on the likelihood function) $L(\overline{\mathcal{M}}, \mathcal{M}) = \sum_{\mathcal{O}_T} \log P[\mathcal{O}_T, \mathcal{Q}_T|\mathcal{M}]P[\mathcal{Q}_T|\mathcal{O}_T, \overline{\mathcal{M}}]$, where $\overline{\mathcal{M}}$ is the current estimated model parameters, $\mathcal{Q}_T$ is the sequence of hidden states during $(1, T)$ and $P[\mathcal{O}_T, \mathcal{Q}_T|\mathcal{M}]$ is the joint likelihood of observable variables and hidden states. The Maximization step obtains the new model parameter values which maximizes $L(\overline{\mathcal{M}}, \mathcal{M})$.

Maximizing the above function leads to simple formulas for HMMs, as follows. Let $\mathcal{M}^{(k)}$ denote the model parameters after the k-th iteration. $\mathcal{M}^{(0)}$ is the initial estimate.

Define $\phi_t^{(k)}(i, j) = P[q_t = s_i, q_{t+1} = s_j|\mathcal{O}, \mathcal{M}^{(k)}]$, i.e., conditioned on the observation sequence and the k-th estimate of the model parameters. These quantities can be computed from the forward and backward computations as introduced earlier for Problem 1.

The next iteration of model parameters is based on the EM algorithm. For example for the $(k + 1)$-st model iteration, the estimate for the transition probabilities are given by:

$$p^{(k+1)}(i, j) = \frac{\sum_{t=1}^{T-1} \phi_t^{(k)}(i, j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \phi_t^{(k)}(i, j)}$$

Similar expressions are used to find the next iteration of values for initial state probabilities and the conditional probabilities for observation values conditioned on the current MC state. It can be shown that this iteration converges to at least a local maximum likelihood solution for the model parameters [13]. Note that one interesting feature of the procedure is that model parameters can be re-estimated based on new observations.

Problem 4.
Basically this is a classification problem and one simply applies Bayes Theorem.

$$P[\mathcal{M}_j|\mathcal{O}] = \frac{P[\mathcal{M}_j]P[\mathcal{O}|\mathcal{M}_j]}{\sum_{i=1}^{K} P[\mathcal{M}_i]P[\mathcal{O}|\mathcal{M}_i]}$$

The a priori probabilities $P[\mathcal{M}_i]$ also have to be known or estimated. The remaining components in the above expression can be computed using the procedure described for Problem 1.

2.3 Remarks

In the sections above we briefly summarized introductory material on HMMs. In what follows we make a few additional remarks which are useful to understand the remaining sections.

The choice of N. When building a model, the analyst may not be able to associated some significance to each hidden state nor have a clear idea on the number of hidden states that should be employed. In the example of section 1, the description of the problem leads naturally to hidden states associated with the user intent, and the number of states N naturally follows from this. However, in many application models, there may be no meaning that can be attributed to the hidden states and, therefore, N must be guessed. In addition, even when one can think of some meaning to the hidden states, the value of N may not be evident. Referring to the online book store example, one can easily imagine other user intents. For instance, one may choose to break the *interested in a product* state into two, such as *showing some interest* and *very interested* in a product. What would be the "best" model to use?

Deciding on the value of N depends largely on the analyst experience. This issue is somewhat similar to that of determining which state variables to use when one is building a Markov chain model. We should be aware that the choice of N impacts the accuracy of the model to obtain the desired answer as well as the cost to solve the model. There is some formalism that can be used to help the analyst. This problem is treated in [14]. There, the estimator proposed is based on the entropy for a stationary stochastic process. A function of the model entropy is selected and evaluated for different values of N. The smallest value of N is chosen such that, by increasing N, no significant improvement in the function value is observed.

The hidden Markov chain structure. Another issue is the choice of a proper model structure. In order to apply the EM-algorithm, the analyst has to guess initial values for the entries of the state transition matrix $\mathbf{P}$. In general, one may choose $p_{ij} \neq 0$ for all i and j. However, depending on the particular problem, the analyst may have a good grasp on the structure of the underline Markov chain. For instance, in the online bookstore example, one may know a priori that the probability of jumping from the *just browsing* state to the *ready to order* state is so small that it can be neglected. It is easy to see that, if $p_{ij}^{(0)} = 0$, then $p_{ij}^{(k)} = 0$ for all the subsequent steps of the EM-algorithm. Thus, the hidden Markov chain initial structure is preserved throughout the iterations. A good initial structure may lead to a more accurate and efficient parameter estimation result for the problem under investigation.

Similar comments apply to the initial choice of the conditional probabilities of symbol emission, $b_j^{(0)}(k)$'s. From Figure 1(b) the probability of clicking page O (*set of products within a category*) while in state *ready to order* may be very low and perhaps can be ignored. If $b_j^{(0)}(k) = 0$, then $b_j^{(n)}(k) = 0$ for the remaining n steps of the EM-algorithm. These kind of structures may be known from some prior knowledge of the model and can be *forced* during the analysis.

Hierarchical models and Hidden Semi-Markov models The most common type of HMM is presented in Figure 1(b), where a probability distribution for symbol emission is associate to each of the hidden Markov states. The model operates by outputting a unique symbol per visit of a hidden state.

In some problems, however, it may be helpful to use special structures for symbol emission within a state and to relax the assumption that a unique symbol is emitted per visit to any hidden state. This leads to a hierarchy of HMMs (HHMM) [11]. (In [11], examples drawn from the areas of analysis of natural text and cursive handwriting are given.)

One simple hierarchy is to associate a discrete time Markov chain to each hidden state. In this case, a symbol is output at every visit of the *lower level* Markov chain. Section 3 presents examples of structures proposed for two different applications. Figure 2 shows one such structure. One issue is the choice of the number of symbols that can be output (or, equivalently, the number of lower level Markov chain states that can be visited) per visit of a hidden state. One can choose, for instance, to fix the number of symbols that can be emitted per hidden state visit. Another choice is to exit the hidden state once the second level Markov chain reaches an "absorbing state". In this second case, the number of symbols emitted may be unbounded, depending on the structure of the second level Markov chain.

It can be shown that that the parameter estimation procedure for HMMs can be extended to handle hierarchical structures [11]. For special structures such as that proposed in [15], the application of the EM-algorithm leads to simple equations for estimating the model parameters and, in this case, it is computationally advantageously to use this hierarchical structure in contrast with general HMMs structures.

There is another generalization of HMMs called hidden semi-Markov models (HSMM). A HSMM is similar to a HMM, except that the duration of a hidden state is explicitly modeled. More specifically, at each hidden state, a *segment* of symbols is emitted according to some distribution. An issue is the way to compute the probability of emitting a particular segment. One possibility is to assume "conditional independence" where each symbol in a sequence is drawn from the same symbol distribution associated with a hidden state, independently of the previous symbols at that state. However, an arbitrary joint distribution for the segment symbols can also be assumed. It is not difficult to see that, depending on the distribution used for a segment, a HSMM can be reduced to a HMM. In addition, there are similarities between a HSMM and a HHMM.

As in the case of HHMM, one must extend the estimation algorithms to handle HSMM. See [23] for one such extension and the references therein.

3 Hidden Markov Models used in Performance Evaluation

3.1 Modeling Network Channel Losses and Delay Characteristics

HMMs have been used to develop traffic generation models useful, for instance to include in simulation studies where loss and delay characteristics metrics impact the application under investigation. On the other hand, HMMs have also been used as predictors of these metrics for online applications. The goal in this case is to adapt the application behavior to better cope with the predicted channel conditions.

Salamatian and Vaton [14] were among the first to propose the use of a discrete time HMM to model the loss process of an Internet channel. In their model, one out of two symbols can be emitted: the first indicates a packet loss in the path from source to destination and the second, the complimentary event. The choice of N is based on the entropy of the loss process, as outlined in section 2. After parameterizing the model from real loss traces, they concluded that the loss process was accurately modeled using only 4 hidden states. This is a significant improvement from previous works that employed a k-th order Markov model [22].

Since losses and delay have jointly an adverse impact on the performance of some applications and they are usually correlated, models have been proposed that try to capture these two metrics in a single HMM. In [17] a HMM is built from the "observations" collected by probing packets. Each of the first $M-1$ symbols represent a delay range after the collected delay samples are discretized. The M-th symbol indicates a loss. The number of hidden states was obtained from studies that compared the autocorrelation of the real traces w.r.t. those generated by the model. Since the HMM is basically a discrete process, obtained from the samples collected at discrete

points in time, the authors also built a continuous time HMM from the discrete version, to emulate delay and losses at arbitrary points of time. The transformation from $\mathbf{P}$ to a generator matrix uses the relation: $\mathbf{P} = e^{\Delta \mathbf{Q}}$, where Δ is an interval between samples.

The dynamics of the packet loss process in the Internet can be extremely variable over different time scales and it is difficult to capture the process characteristics with a regular HMM model. Layered models, such as HHMM have been used in these cases.

In [16] a 2-layer HHMM model with two time scales is proposed. At the top of the hierarchy, a Markov chain models different channel loss conditions at a coarse time scale. Each MC state h corresponds to loss rates in a specified range such as *no loss*, *minor loss rates*, *serious loss rate*. The second layer contains a number of HMMs, each similar in structure to that proposed by [14]. A HMM $\mathcal{M}_h$ in the lower hierarchy corresponds to state h of the top layer MC. In this paper, each level in the hierarchy is parameterized *independently* of the other and, as such, the parameter estimation method differs from the usual HHMM parametrization where the corresponding optimization problem would include the entire 2-layer model. Therefore, the transition probabilities of the top MC model are immediately estimated from the relative frequencies of transitions between two loss rates in different ranges, independently of the observed loss bursts.

HMMs can be used as predictors of events in the future. For instance, video or voice applications could benefit from estimating future channel loss statistics to better adapt their encoding algorithms to varying network conditions. In addition, a good predictor for channel characteristics could be used by applications that have the choice of selecting two or more distinct paths to communicate with the application peer.

The work of [16] uses the proposed HHMM to infer the length of loss bursts that an application would experience in the immediate future in a source-destination path. Let T be the length of the prediction interval. From the current observable state of the top-layer MC h and its transition probabilities, the most probably state in the next time interval T is obtained and the loss process is assumed to be that modeled by $\mathcal{M}_h$ during that interval. Therefore the statistics on burst length are immediately inferred.

Reference [15] provides an example of a 2-level HHMM used as a packet loss model to predict losses for an online VoIP application. In their model the hidden states can be thought of as modeling different channel loss conditions at a course time scale, similar to the top layer MC of [16]. Each observation consists of a batch of S packet outcomes, and thus models the loss process at a finner time scale. Clearly, the emission of S symbols per hidden state can be modeled as a HMM in which a hidden state emits one of the 2^S possible sample paths for S packets and so, even for moderate values of S, the model becomes intractable. This is an example where the use of HHMM can be used to obtain significant savings. We can restrict the distribution of the observations in a batch by assuming that losses are generated following a 2-state Gilbert model and, as such, only two parameters, for each upper level state, need to be estimated instead of 2^S probabilities. It should be clear that the Gilbert model parameter values are correlated with the hidden state and model the short-term loss process. Therefore, this HHMM can capture loss process changes at different time scales.

The predictive model is re-estimated from continuous real time measures. From the measured loss, the current channel conditions are evaluated. For instance, the current hidden state is inferred by solving Problem 2. For a given current state, and applying transient analysis over the HHMM, loss process statistics are inferred in the future. The work in [15] applies this technique to select the proper FEC scheme at the packet level for VoIP applications, and shows that a measure of voice quality can be significant improved with little packet transmission overhead.

3.2 Traffic Modeling

Traffic modeling has been an important area of research since traffic impacts the dimensioning of the Internet resources. Traffic models have been used in many areas such as admission control, resource allocation, capacity planning, and traffic classification, to name a few. HMMs have received attention as the traffic model of choice for these problems because of their ability to capture important traffic statistical characteristics with only a relatively small number of states. These studies focus on modeling the packet flow either generated by an individual application or that of the aggregate traffic on a single channel.

Often both the inter-arrival packet times and packet sizes from the traffic generated by an application exhibit significant auto-correlation. These two metrics then have been used to construct HMMs for different objectives.

In [1] a HMM is used for identifying distinct applications by observing the traffic on their TCP connection. The authors show that the sizes of just the first few TCP packets provide good evidence for classifying the traffic.

Both the packet size and packet inter-arrival time are employed in [6,19] to build their HMM. The objective of [6] is to build accurate traffic models for the packet flow generated by each of several applications such as SMTP, HTTP, network games and instant messaging. It was shown that the models capture well the marginal distribution and auto-covariance of the packet inter-arrival time and packet size for the studied applications. In a related work [5] these models are employed for traffic classification, similarly to what is described in section 2, Problem 4. (See also [19] for another similar example of network traffic classification.)

One of the objectives of a packet level traffic model is to help with the task of capacity planning, and so it is important to devise accurate models for the aggregate traffic in a channel. Reference [9], is an example of such application. There, a HMM is used for modeling the aggregate traffic at each link of a given network structure (nodes connected to links). An optimization problem is defined in which the link capacities are obtained such that the total network cost is minimized under the constraint that different QoS metrics (for different services using the network) are satisfied. The QoS metrics employed are the loss probability and the round trip time (RTT) in a source-destination path. The observed traffic rates at each link are discretized in 30 levels and the observed symbols of the HMM correspond to these levels. The HMM proposed has a branch-Erlangian structure with 4 hidden states. From the HMM, for each link, a function was obtained which maps the delay in a link from the link capacity. This function is then used for the optimization problem.

3.3 Workload generation

In this set of applications, HMMs are used to generate synthetic workload at the application level, instead of the packet level of section 3.2. Similarly to other HMM model applications, the workload at the application level may involve different time scales. For example, when a user clicks on a hyperlink to access a web page, the browser sends requests for that page and its in-line objects. This *ON* period is followed by an *OFF* interval where the user reads the page contents and no requests are sent to the server. Two time scales are then apparent: one that represents the duration of the OFF periods and another that represents the intervals between requests generated during the *ON* period. As mentioned in previous sections, HSMMs and HHMMs are useful tools to represent different time scales.

In [21] a HSMM is proposed to model a web user's behavior. The hidden states are associated with web pages visited by a web user and the duration of a hidden state is related to the number of HTTP requests received by the web server when the users clicks the corresponding web page. Transition between hidden states characterizes the user's browsing behavior from one page to

another. The sequence of observations is a vector where the first element is a request for a specific object and the second is the interval between the current and the previous request. In [21] the proposed HSMM is used for anomaly detection. The model is trained using a sequence of requests made by normal web users and an abnormal user is detected by comparing the expected entropy calculated from the model with that from the monitored user.

HSMMs are also used to represent mobile users and their requirements (e.g. bandwidth to transmit a video) in [12]. The observations are attributes of a mobile user such as measured geo-location (which is different from the real location due to measurement errors) and user requirements. The hidden states represent the real location, directions of movement and speed ranges which are not directly observable. The transitions between hidden states define the user path from a source to a destination state. The information obtained from the HSMM model can be used, for instance to characterize traffic streams arriving from mobile users at a wireless device and consequently to determine the user resource allocation and admission control.

In what follows we describe a student behavioral model when accessing a video server of a real distance learning application. The model, proposed in [4], tries to capture the sequence of commands students issue when viewing pre-recorded lectures. The lectures are from the CEDERJ Computer Science graduate course [8] and the model is parameterized from thousands of logs collected automatically at the video server. Each lecture consists of a pre-recorded video, slides synchronized with the video stream and a series of topics which are associated to particular time points in the video. The students have full DVD-like control such as play, pause, jump (forward or backward) at any time.

The proposed model is a hierarchical HMM inspired by the work of [10]. Transitions between hidden states represent transitions between slides during a user session, but there is no one-to-one correspondence between hidden states and the number of slides in a lecture. Figure 2 illustrates the hierarchical HMM. The observed symbols are play, jump forward, rewind, pause, end of session, next slide.

Note that the structure of the discrete time Markov chains for each hidden state is constrained and, as such, invalid sequence of actions cannot be generated as, for instance, two consecutive pauses.

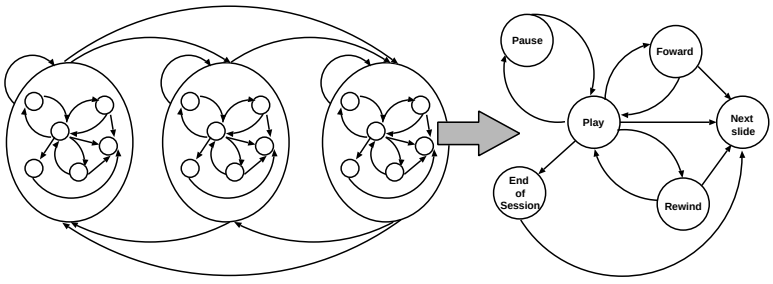


Fig. 2. The hierarchical HMM.

The main goal of the proposed HHMM is to generate a synthetic workload used, for instance, to size the video server. The HHMM is capable of generating a sequence of actions but does not include the duration of each action (pause, play, jump forward, etc). The distributions for each action are estimated independently, directly from the trace logs and used jointly with the HHMM to generate the workload.

4 The Tangram-II tool

Modeling tools are essential to support the development of performance models. Despite the fact that there are a few tools available for handling HMMs, most of them are targeted to specific application areas such as speech recognition and, to our knowledge, none is embedded in a performance evaluation environment. The Tangram-II tool [3, 7] has been developed over the last 20 years to provide a useful environment for performance analysts to build and experiment with their models. The tool has a unique modeling paradigm for specifying a model, a rich set of analytical tools, modules to support the calculation of many measures of interest, simulators that include a hybrid fluid and event driven simulator and a traffic generator, all integrated in a single environment.

Concerning HMMs, Tangram-II has a specific module that allows users to work with regular or hierarchical HMMs from the tool's interface. The tool supports three different types of HHMM, each corresponding to specific assumptions for the second layer in the hierarchy. In the first type, a Gilbert Markov model is used for symbol emission, and it is assumed that a fixed number of symbols is emitted at each visit to a hidden state (the state in the top hierarchy). The second type also assumes that the number of symbols emitted per hidden state is fixed, but a general Markov chain can be specified by the user to model the distribution of emissions. The third type of HHMM is similar to the second but the lower layer Markov chain is assumed to contain an absorbing state. When the absorbing state is reached, the top level Markov chain is assumed to make a transition according to the hidden state transition probability matrix $\mathbf{P}$. This last type of HHMM allows for the modeling of variable symbol emissions batch sizes for each hidden state.

When creating a HMM one difficulty is to input the initial values for the hidden state transition matrix $\mathbf{P}$, especially if the hidden chain contains a non-trivial number of states. Tangram-II facilitates this task by allowing the user to specify the hidden chain with *high level* objects, using the same *Model Specification* module used for regular Markovian models. Any Markov chain model associated with the hidden states can also be specified using the same modeling description paradigm.

Once the model is constructed, the user must specify which state variables are associated with the hidden model and the lower layer model. Figure 3(a) shows a snapshot of one of the tool's interface that allows the user to choose from one of the four types of HMMs. Figure 3(b) shows the interface for an example where a 2-level HHMM is modeled using a Gilbert model for the lower level hierarchy. Finally, Figure 3(c) depicts the parameters that are automatically extracted from the model's specification module. Empty slots are left for the user to include additional initial parameter values, such as the initial probabilities for the hidden states.

Once the model is specified, the user can select from several solution algorithms. Three of them are associated with the first three HMM problems outlined in section 2. Note that each of the algorithms are adapted to take into account the characteristics of the type of model (regular HMM or HHMM) used.

Before selecting a solution, the user must load the observations collected and stored, for instance, in a trace file. Loading the trace file is done through a special plugin, the MTK HMM plugin. From the data, the user can, for instance, train the model, or calculate the log likelihood of the observation sample. The user can also simulate a previously constructed HMM and generate a sample of symbols. In addition, the tool implements a forecast algorithm, that can predict the probability distribution for the symbols that will follow after a given observed sample. Details can be found in the tool's manual available at [20].

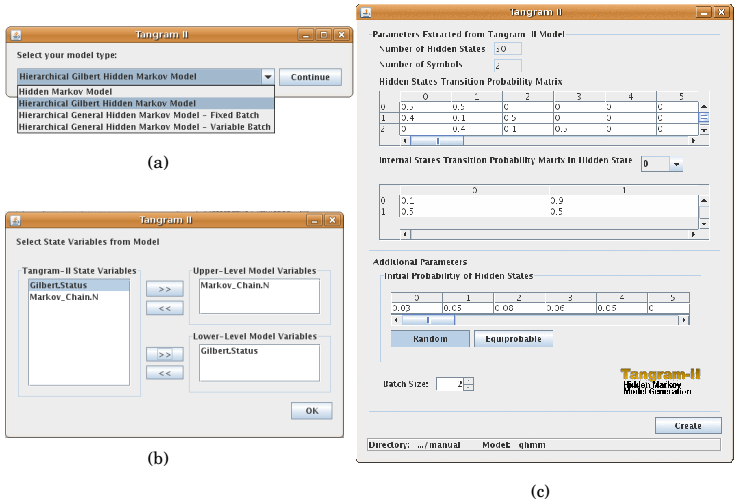


Fig. 3. (a) Model selection interface; (b) State variable selection interface; (c) Additional parameter specification interface

5 Conclusion

Hidden Markov models have proven to be a useful tool for many different areas. In performance evaluation, networking and workload generation have been the predominant application subareas so far. After outlining the mathematical background, we surveyed three applications that served to exemplify the usefulness of HMMs. We also present a modeling tool, Tangram-II that implements the main algorithms for solving the problems surveyed and provides a useful graphical interface to help in the analysis process.

Many modeling issues still remain open, such us the proper choice of the number of hidden states in the model, the choice of the initial parameter values for problem 3 and the initial structure for the underlying Markov chains. Nevertheless, the modeling paradigm is useful when the analyst has little knowledge of how the underlying system works and instead has just access to observations to construct a model that represents the system under study. HMMs have also proven useful for predicting future behavior of measures of interest. We expect many new applications of HMMs to performance problems in the future.

References

1. Laurent Bernaille, Renata Teixeira, and Kave Salamatian. Early application identification. In *CoNEXT'06*, pages 1–12. ACM, 2006.
2. Olivier Cappé. Ten years of HMMs, March 2001.
3. Rosa M.L.R. Carmo, Luis R. de Carvalho, Edmundo de Souza e Silva, Morganna C. Diniz, and Richard R. Muntz. Performance/Availability Modeling with the TANGRAM-II Modeling Environment. *Performance Evaluation*, 33(1):45–65, 1998.
4. Carolina C. L. B. de Vielmond and Rosa M. M. Leão and Edmundo de Souza e Silva. A hierarchical HMM for iterative users acessing a multimedia server (in portuguese). In *Brazilian Symposium on Computer Networks*, pages 469–482, 2007.
5. A. Dainotti, W. de Donato, A. Pescape, and P. Salvo Rossi. Classification of network traffic via packet-level Hidden Markov Models. In *IEEE GLOBECOM 2008*, pages 1–5, 2008.

6. Alberto Dainotti, Antonio Pescapé, Pierluigi Salvo Rossi, Francesco Palmieri, and Giorgio Ventre. Internet traffic modeling by means of hidden markov models. *Computer Networks*, 52(14):2645–2662, 2008.
7. Edmundo de Souza e Silva and Daniel R. Figueiredo and Rosa M.M. Leão. The TANGRAM-II integrated modeling environment for computer systems and networks. *Performance Evaluation Review*, 36(4):64–69, 2009.
8. Edmundo de Souza e Silva and Rosa M. M. Leão and Anna D. Santos and Jorge Allyson Azevedo and Bernardo C. Machado Netto. Multimedia Supporting Tools for the CEDERJ Distance Learning Initiative applied to the Computer Systems Course. In *22th ICDE World Conference on Distance Education*, pages 1–11, 2006.
9. Edmundo de Souza e Silva and Rosa M. M. Leão and Marcos B. Trindade and Ana Paula C. da Silva and Bruno F. Ribeiro and Flávio P. Duarte and Jorge A. Azevedo. A methodology for dimensioning IP networks with QoS using Hidden Markov Models. Technical report, UFRJ-COPPE-PESC, 2005.
10. Fernando Silveira Filho and Edson H. Watanabe and Edmundo de Souza e Silva. Adaptive forward error correction for interactive streaming over the Internet. In *IEEE Globecom'06*, pages 1–6, 2006.
11. S. Fine, Y. Singer, and N. Tishby. The hierarchical Hidden Markov Model: Analysis and applications. *Machine Learning*, 32:41–62, 1998.
12. Hisashi Kobayashi, Shun-Zheng Yu, and Brian L. Mark. An integrated mobility and traffic model for resource allocation in wireless networks. In *3rd ACM international workshop on Wireless mobile multimedia (WOWMOM'00)*, pages 39–47. ACM, 2000.
13. L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, February 1989.
14. Kavé Salamatian and Sandrine Vaton. Hidden markov modeling for network communication channels. *Performance Evaluation Review*, 29(1):92–101, 2001.
15. Fernando Silveira and Edmundo de Souza e Silva. Predicting packet loss statistics with hidden Markov models. *Performance Evaluation Review*, 35(3):19–21, 2007. (extended updated version in http://www.land.ufrj.br/ẽdmundo).
16. Shu Tao and Roch Guerin. On-line estimation of internet path performance: An application perspective. In *IEEE Infocom*, 2004.
17. Wei Wei, Bing Wang, and Don Towsley. Continuous-time hidden markov models for network performance evaluation. *Performance Evaluation*, 49(1-4):129–146, 2002.
18. Lloyd R. Welch. Hidden markov models and the baum-welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4):10–14, December 2003.
19. Charles Wright, Fabian Monrose, and Gerald M. Masson. Hmm profiles for network traffic classification. In *the 2004 ACM workshop on Visualization and data mining for computer security(VizSEC/DMSEC'04)*, pages 9–15. ACM, 2004.
20. www.land.ufrj.br. The TANGRAM-II manual.
21. Yi Xie and Shun-Zheng Yu. A large-scale hidden semi-markov model for anomaly detection on user browsing behaviors. *IEEE/ACM Transactions on Networking*, 17(1):54–65, 2009.
22. Maya Yajnik, Sue Moon, Jim Kurose, and Don Towsley. Measurement and modelling of the temporal dependence in packet loss. In *IEEE Infocom'04*, 1999.
23. Shun-Zheng Yu and Hisashi Kobayashi. An efficient forward-backward algorithm for an explicit-duration hidden markov model. *IEEE Signal Processing Letters*, 10(1):11–14, 2003.