Maximum A Posteriori Path Estimation with Input Trace Perturbation: Algorithms and Application to Credible Rating of Human Routines

Daniel H. Wilson
Robotics Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh PA 15213
dwilson@cs.cmu.edu

Matthai Philipose
Intel Research
6th Floor
1100 NE 45th Street
Seattle WA 98115
matthai@cs.washington.edu

Abstract

Rating how well a routine activity is performed can be valuable in a variety of domains. Making the rating inexpensive and credible is a key aspect of the problem. We formalize the problem as MAP estimation in HMMs where the incoming trace needs repair. We present polynomial time algorithms for computing minimal repairs with maximal likelihood for HMMs, Hidden Semi-Markov Models (HSMMs) and a form of HMMs constrained with a fragment of the temporal logic LTL. We present some results to show the promise of our approach.

1 Introduction

Rating how well a person performs a routine activity is a broadly useful capability with many applications: professors train medical students by rating their execution of established procedures, caregivers assess the well-being of their wards by rating how well they are able to perform activities of daily living, and managers and workflow experts identify poorly performed procedures that cause bottlenecks in a system. Although rating routine activity is certainly useful, as conventionally done it is also very expensive – each activity performance requires a dedicated human observer (often an expert). Many situations where gauging the performance of routine activities could be helpful are therefore either not rated at all, or rated in a cursory manner. Clearly, an opportunity exists for automated techniques to reduce the cost of rating. In this paper, we explore methods for automatically rating performances of routine activities.

The basic classification task of rating, going from observations to scores, is amenable to a variety of standard approaches. Rating becomes challenging, however, if we wish to make it both *incrementally inexpensive* and *credible*. We define an incrementally inexpensive rater to be a rater in which the extra cost of rating a new activity is relatively low. The main determinant of cost is whether rating a new activity requires a custom classifier to be developed from scratch, or whether a generic classifier of some kind can be easily customized to the task. A credible rater is one that is both *relevant* and *transparent*. By relevant, we mean that the classification model for a particular rating task should reflect constraints on activity performance that are important to those using the rating. For example, a professor grading anesthesiology students performing an intubation may want to indicate what her notion of good performance is. By transparency, we mean that the system should be able to justify why it has assigned a particular rating. Ideally, the justification should be *constructive*, in that it should suggest how a low-rated performance may be altered to obtain a high-rated one.

Our techniques for rating activity routines are designed to satisfy the above requirements. To lower incremental cost, we choose a representation that is easily learned: all activities to be rated in our system are modeled by variants of Hidden Markov Models (HMMs). We intend that these models, especially given simple prior information, can be learned easily from training examples. More crucially, we formulate the justification for a rating relative to this model generically as the set of edits required on the trace generated by the rated activities; we therefore do not require special identification and modeling of errors and their causes. A fundamental weakness of these models is that they are first order, preventing them from capturing certain important correlations. We augment the Markov models with an intuitive constraint formalism (a small fragment of the temporal logic LTL [1]) that allows raters to explicitly state relevant constraints. Given these relevant and easy-to-construct models, we formulate rating as the likelihood of (possibly edited) observation sequences.

The core of this paper consists of efficient algorithms to compute maximum likelihood paths of minimally edited versions of incoming observations with respect to various representations for activities, including HMMs, HSMMs and temporally constrained HMMs. The algorithms use the dynamic programming technique used to great effect by the well-known Viterbi algorithm. A preliminary evaluation shows the promise of our techniques.

2 Overview

In this section, we describe how we expect our system to be used, and we sketch how our system supports this usage model. In this paper, our goal is to develop a system that rates how well an elder performs day-to-day activities. Such a system is of great interest to the eldercare industry. In theory, caregivers will assess the elders' well-being by consulting ratings summaries and credible explanations of performance deficits. For example, the system may recognize that an elder is no longer able to prepare their daily bowl of soup, and report why (e.g., can't reach cabinet or difficulty holding spoon).

To end-users, our system represents activities as a set of steps. Each step has a duration and a set of observed actions performed, and is succeeded by other steps. For instance, the activity "making soup" for a particular elder may have the following steps: "preheat water," "open can," "mix and boil ingredients," "serve," and "clean up." The step "open can," may have an average duration of 45 seconds and contain the following actions: "use utensil drawer," "use can opener," "use can," and "use pantry door."

For concreteness, we will assume in what follows that we are using RFID-based [4] sensors that will directly sense the action of using particular objects. Therefore, all of our actions are of the form "use X" where X is some object. Inherently, our system requires that actions are observable by sensors. Given an activity trace (i.e., a trace of actions that constitutes a particular execution of an activity), our system provides a rating (e.g., *pass* or *fail*). If the grade is a *fail*, the system provides an alternate sequence of actions as close to the original as possible that would have elicited a *pass* grade (essentially a constructive justification of the grade). In more detail, use of the system proceeds as follows:

Learning the model A human demonstrator performs the routine in an exemplary fashion. The system collects traces $Y_1, \ldots, Y_n$ of the routine. Each trace Y_i is a sequence of time-stamped observations $y_{i1}, \ldots, y_{im_i}$ of the demonstrator's actions. The traces are used to learn a dynamic stochastic model (either an HMM or an HSMM) with parameters λ. The hidden states $s_1, \ldots, s_N$ of the model correspond to the "activity steps" above, and are labelled $l_1, \ldots, l_N$ with the names of the step.

Adding global constraints Typically, the first-order model learned in the previous step cannot capture important higher-order correlations. For instance, in a successful soup-making routine, the stove, if it is used, should eventually be turned off after it is turned on. The turning on would happen in the "preheat water" step, but the turning off may not happen until the end of the "serve" step. The human rater explicitly adds a set $\mathcal{C}$ of constraints on the sequence of hidden states or observations that specify these required higher-order correlations. In this case, a possible constraint would be of the form use("stove control knob") $\mathcal{E}$ use("stove control knob"), read as "a use of a stove control knob should eventually succeeded by a use of a stove control knob".

Learning rating thresholds A human rater rates each trace Y_i with a rating $r_i \in \{pass, fail\}$. Let the *constrained MAP likelihood* of trace Y given λ and $\mathcal{C}$, $\hat{l}_Y = \text{CMAP}(M, Y, \mathcal{C})$, be the likelihood of the path $\hat{S}_Y$ with maximum a posteriori (MAP) likelihood given λ and Y that satisfies $\mathcal{C}$. We perform a simple thresholding computation to calculate the likelihood threshold L such that, given the classification function *R(l) = if* $l < L$ *then fail else pass*, $R(\hat{l}_{Y_i}) = r_i$ for as many of the Y_i as possible. Intuitively, L separates the passes from the fails.

Generating a rating and justification Given the constrained model $(\lambda, \mathcal{C})$ and threshold L, the automated rater is ready for use. The person to be rated generates a trace $Y = y_1, \ldots, y_m$ to be rated automatically. The rater finds the constrained MAP likelihood $\hat{l}_Y$ and path $\hat{S}_Y = (\hat{s}_1, y_1), \ldots, (\hat{s}_m, y_m)$ for Y, and assigns it the rating $r = R(\hat{l}_Y)$. If $r = fail$, the rater attempts to produce a *repaired trace* trace $Y' = y'_1, \ldots, y'_m$ such that the edit distance between Y and Y' is as small as possible, and $\hat{l}_{Y'} > L$. In other words, Y' is the closest trace to T that passes. The rater offers r as the rating for the activity and, if appropriate, $\delta_{\hat{S}_Y, \hat{S}_{Y'}}$, the set of edits needed to transform $\hat{S}_Y$ into $\hat{S}_{Y'}$, as the justification for the rating.

As described above, our rating system employs two key non-standard pieces of machinery.

1. A method to compute the repaired observation trace T', that is a minimum edit distance from a given trace T whose likelihood is above a pre-specified threshold L.
2. A method to compute the constrained MAP likelihood function CMAP$(M, T, \mathcal{C})$.

3 Trace Repair for Hidden Markov Models

A Hidden Markov Model (HMM) $\lambda = (A, B, \pi)$ is a commonly used stochastic model for dynamic systems [5]. We formally pose the trace repair problem as a variation of estimating the most likely state sequence given a sequence of observations (classically solved via the Viterbi algorithm). An HMM is defined as follows. Let $Q_A = \{q_1, \ldots, q_N\}$ be the states of the process being modeled, and $O_B = \{o_1, \ldots, o_M\}$ the observation signals possibly generated by the process. We use meta-variables s_t and y_t to denote the states and observations respectively at time t. A_{ij} is the probability $p(s_{t+1} = q_j | s_t = q_i)$ of transitioning from state q_i at time t to q_j at time $t+1$ for any t; B_{ij} is the probability $p(y_t = o_j | s_t = q_i)$ of generating observation o_j when in state q_i (we write B_{iy_t} for B_{ij} such that $y_t = o_j$). The initial state distribution $\pi_i = p(s_0 = q_i)$.

3.1 The Repaired MAP Path Estimation Problem

We now formulate the problem of MAP path estimation given an observation sequence if we are allowed to first make a limited number of edits or "repairs" to the sequence. We begin by formalizing the notion of an edit. We then state the repaired MAP path estimation problem and present a variation of the Viterbi algorithm to solve it.

Let Y^N be the set of length-N strings of observations over some finite alphabet Y. Then $e^{k,N} = ((b_1, s_1), ...(b_N, s_N))$ is a *length-N k-edit vector on* Y^N, with b_i boolean, s_i strings over Y, and $k = \sum_{1 \leq i \leq N}(b_i + |s_i|)$. For instance, $\hat{y}_1$ = "cat" is a string in Y^3; $e_1^{4,3} = ((\text{false}, \text{"BB"}), (\text{false}, \text{""}), (\text{true}, \text{"R"}))$ is an edit vector on Y^3. *Applying* an edit vector e to string $\hat{y}_n = y_1 \ldots y_n$, written $e(\hat{y})$ results in a new string $\hat{y}'$ obtained as follows. For $1 \leq i \leq n$, let if $e.b_i$ is true, then replace y_i with $\hat{y}'_i$, else replace y_i with $y_i e.s_i$ ($e.s_i$ appended to y_i). For example,

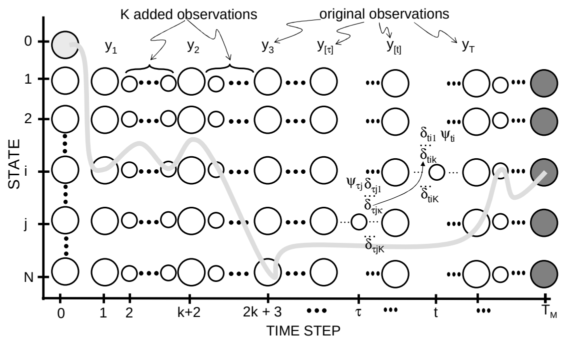

Figure 1: Trellis for k-Edit Viterbi on HMMs

$e_1^{4,3}(\hat{y}_1)$ = "cBBaR". A string $\hat{y}'$ is a *k-edit* of another $\hat{y}$ if there exists edit vector $e^{k,N}$ such that $\hat{y}' = e^{k,N}(\hat{y})$.

We are now ready to specify the problem of MAP estimation with repairs:

Definition 1. (Repaired MAP Path Estimation Problem (RMAP)) *Given observation sequence $\hat{y}_T$, HMM $\lambda = (A, B, \pi)$ and edit distance K find observation sequence $\hat{y}'_{T'} = \dot{y}_1 \dots \dot{y}_{T'}$ that is a K-edit of $\hat{y}_T$ and path $\hat{s}_{T'} = \dot{s}_1, \dots, \dot{s}_{T'}$ maximizing $p(\hat{s}_{T'}, \hat{y}'_{T'})$ over all T', $\hat{s}_{T'}$ and $\hat{y}'_{T'}$.*

Before discussing our solution, we define string $\hat{y}'$ as the *(k,a)-edit* of string $\hat{y}_n$ if $\hat{y}' = e^{k,n}(\hat{y})$ for some $e^{k,n}$, $|e^{k,n}.s_n| \leq a$, and additionally, $e^{k,n}.b_n$ if $a = 0$ and $|e^{k,n}.s_n| > 0$ if $a > 0$. The (k, a)-edit of a string requires its last character to be either preserved or replaced by at least one character, with at most a characters added. Edits compose as follows ($d_{\nu n} = n - \nu$; $a' = 1$ if $a \neq 0$, 0 otherwise; $(\nu, \alpha) <_k (n, a)$ if $\nu < n$ and $\alpha \leq k$, or if $\nu = n$ and $\alpha < a$):

Lemma 1. (Edit Composition) *Let $Y_n^{ka\hat{y}}$ be the set of all (k, a)-edits of the n-prefix of string $\hat{y}$ over Y. Then $Y_n^{ka\hat{y}} = \{\hat{y}'y | y \in Y, \hat{y}' \in Y_\nu^{\kappa\alpha\hat{y}}, (\nu, \alpha) <_k (n, a), \kappa + d_{\nu n} + a' = k\}$.*

Table 1 specifies an algorithm (the k-Edit Viterbi (KEV) algorithm) to solve RMAP. KEV iterates over the T *original observations* in the incoming observation string. For each original observation, it iterates over possibilities for the K *added observations* at that position, for a total of $TK + T$ iterations. At each iteration t corresponding to original obervation $[t] = t$ div $(K + 1) + 1$ and added observation $\#t = t$ mod $(K+1)$, KEV computes the likelihood δ_{tik} of the most likely path ending in state i given an observation string that is a $(k, \#t)$-edit of $y_1 \dots y_{[t]}$ over all such edit vectors; KEV also records as ψ_{tik} the penultimate state and edit in this path. Following the chain of ψ_{tik}'s back to the start state iteration gives the MAP repaired path.

The *trellis* of figure 1 illustrates KEV. Columns of the trellis represent edits considered for inclusion into the final string. Large circles represent original observations and small ones represent adds. For technical reasons (to allow skipping the first original observation), we add a distinguished start state q_0 with new start probabilities $\pi'_0 = 1$, $A_{0i} = \pi_i$ and $A_{i0} = 0$ and add a column ($t = 0$) processed in the initialization step. To allow skipping the last original observation with

Initialization: $t = 0, k = 1 \dots K, 1 \leq i \leq N$

$$\delta_{t0k} = 1 \qquad \delta_{tik} = 0 \qquad \psi_{tik} = -1$$

Iteration: $1 \leq t \leq TK + T, k = a_t, a_t + 1, \dots, K$

$$(\psi_{tik})\delta_{tik} = \underset{\tau, j, \kappa \text{ s.t. } \kappa + a_t + d_{\tau t} = k}{\text{(arg)max}} \delta_{\tau j \kappa} B_{i y_{it}} A_{ji}$$

Termination: $t = T_M = TK + T + 1$

$$(\psi_{tik})\delta_{tik} = \underset{\tau, j, \kappa \text{ s.t. } \kappa + a_t + d_{\tau t} = k}{\text{(arg)max}} \delta_{\tau i \kappa} \; ; i_M = \underset{1 \leq i \leq N}{\text{argmax}} \, \delta_{tiK}$$

Backtracking: $(t, i, k) = \psi_{T_M i_M K}$; while $t > 0$,

$$1) (\dot{s}_t, \dot{y}_t) = (q_i, y_{it}) \qquad 2) (t, i, k) \leftarrow \psi_{tik}$$

Table 1: The k-Edit Viterbi Algorithm for HMMs

no adds, we add a column ($t = T_M = TK + T + 1$). Rows represent possible hidden states.

The end result of the algorithm is a forward path (shown in light grey in figure 1) through the trellis that, unlike in the conventional Viterbi algorithm, may jump between nodes in non-adjacent time slices. If the path jumps over the slice for an original observation y_i (where i is the position of the observation in the input string $\hat{y}$), we conclude that y_i was deleted from $\hat{y}$, otherwise not. Further, if the path passes through a sequence of added nodes with no intervening original node such that y_i is the first original observation to the left of the sequence, and the observations at these nodes are $y_{i_1}, \dots, y_{i_n}$, we conclude that the string $y_{i_1} \dots y_{i_n}$ was added at the i'th spot in the incoming string. The forward path is the required solution $\hat{s}^{T'}$, and the string of observations along the path is the edited string $\hat{y}^{T'}$.

The algorithm uses three intermediate variables, a_t, $d_{t\tau}$ and y_{it}. Variable $a_t = 1$ if $\#t \neq 0$ and 0 otherwise; $d_{\tau t} = [t] - [\tau]$, represents the number of deletes skipping original observations between τ and t; y_{it} is the observation considered when processing state i at slice t. Note that we only process original observations at time slices $1, K + 1, 2K + 1, \dots$. In all other "added" slices, we need to propose the observed value to be added. A simple but inefficient approach would be to consider for each state, k-value and iteration t, every possible observable $o \in O_B$ as a candidate. In fact, we can consider a single observation instead of all $|O_B|$. The key insight is that, when processing state i in an added slice, it is sufficient to consider adding as observable the most likely observable in that state. Let S_N and Y_N be the sets of all length-N sequences of states and observables. Let $\dot{y}_i = \underset{1 \leq j \leq M}{\text{argmax}} \, B_{ij}$. Let $\hat{s}q$ be the result of appending state q to sequence $\hat{s}$, and similarly for $\hat{y}y$. Then, for all $q_i \in Q_A$:

Lemma 2. $\underset{\hat{s}, \hat{y}' \in S_N, Y_{N+1}}{max} p(\hat{s}q_i, \hat{y}') = \underset{\hat{s}, \hat{y} \in S_N, Y_N}{max} p(\hat{s}q_i, \hat{y}\dot{y}_i)$

This follows from the fact that $\max_{y_i} p(\hat{s}q_i, \hat{y}y_i) = \max_{y_i} \pi_{s1} B_{s_1 y_1} (\prod_{1 \leq i \leq N, s_i q_j \in \hat{s}_N} A_{ij} B_{j y_j})(A_{s_N i} B_{i y_i}) = \pi_{s1} \dots A_{s_N i} \max_{y_i} B_{i y_i} = \pi_{s1} \dots A_{s_N i} \dot{y}_i$. Given this identity for the optimal observable to be added in state q_i, we set y_{it}

to $\dot{y}_i$ if t is an "added" timeslice, and to $y_{[t]}$ otherwise.

We are now ready to establish the soundness of the KEV algorithm. Let S_n^i be the set of length-n sequence of states ending in state q_i. Let Y_n^{tik} be the set of length-n strings of observables that are $(k, \#t)$-edits of $y_1 \dots y_{[t]}$.

Lemma 3. $(\psi_{tik})\delta_{tik} = \underset{n, \hat{s} \in S_n^i, \hat{y} \in Y_n^{tik}}{\textit{(arg)max}} \; p(\hat{s}, \hat{y})$

Proof sketch. Proof is by induction on t. We focus on the inductive case for δ. For ψ, replace "max" with "argmax".

$$\begin{aligned}
\delta_{tik} &= \max_{\tau,j,\kappa} A_{ji} B_{iy_{it}} \delta_{\tau j \kappa} \forall_{j,\tau,\kappa} \text{ s.t. } (\kappa + a_t + d_{\tau t}) = k \\
&= \text{(by the inductive hypothesis)} \\
&\quad \max_{\tau,j,\kappa} (A_{ji} B_{iy_{it}} \max_{n, \hat{s} \in S_n^j, \hat{y} \in Y_n^{\tau j \kappa}} p(\hat{s}, \hat{y}))
\end{aligned}$$

Given s_n, s_{n+1} is independent of $\hat{s}^{n-1}, \hat{y}^n$:

$$\begin{aligned}
A_{ji} &= p(s_{n+1} = q_i | \hat{s}, s_n = q_j, \hat{y}) \; \forall_{n, \hat{s} \in S_{n-1}, \hat{y} \in Y_n^{\tau j \kappa}} \\
&= p(s_{n+1} = q_i | \hat{s}, \hat{y}) \; \forall_{n, \hat{s} \in S_n^j, \hat{y} \in Y_n^{\tau j \kappa}}
\end{aligned}$$

Similarly, for y_{n+1} given s_{n+1}:

$$B_{iy_{it}} = p(y_{n+1} = y_{it} | \hat{s}, s_{n+1} = q_i, \hat{y}) \, \forall_{n, \hat{s} \in S_n^j, \hat{y} \in Y_n^{\tau j \kappa}}$$

Substituting for A_{ji} and $B_{iy_{it}}$ above, and using $p(A, B) = p(A|B)p(B)$ twice, we have, with $(\kappa + a_t + d_{\tau t}) = k$:

$$\delta_{tik} = \max_{\tau,j,\kappa,n,\hat{s} \in S_n^j, \hat{y} \in Y_n^{\tau j \kappa}} p(s_{n+1} = q_i, \hat{s}, y_{n+1} = y_{it}, \hat{y})$$

Lemma 2 ensures that maximizing over $\hat{y}y_{it}$ maximizes over all strings $\hat{y}y$. By lemma 1 maximizing over all $\hat{y}y$ with $\hat{y} \in Y_n^{\tau j \kappa}$ maximizes over $\hat{y} \in Y_{n+1}^{tik}$. Finally, $\forall_{1 \le j \le N, \hat{s} \in S_n^j q_i} \hat{s} = \forall_{1 \le i \le N, \hat{s} \in S_{n+1}^i} \hat{s}$. Modifying the previous equation to reflect these insights:

$$\delta_{tik} = \max_{n, \hat{s} \in S_{n+1}^i, \hat{y} \in Y_{n+1}^{tik}} p(\hat{s}, \hat{y}) = \max_{n, \hat{s} \in S_n^i, \hat{y} \in Y_n^{tik}} p(\hat{s}, \hat{y})$$

□

The soundness of KEV follows in a straightforward way from the above lemma. Further, given that the trellis has $O(TKN)$ nodes, that at each node we compute $O(K)$ δ and ψ values, and we consult $O(NK)$ preceding data values to do so, the complexity of KEV as a whole is $O(TN^2K^3)$.

4 Trace Repair for HSMMs

A Hidden Semi-Markov Model (HSMM) [3] $\lambda = (A, B, D, \pi)$ is identical to an HMM except for the *duration distribution* D. Where an HMM generates a single observation according to B on each visit to a state s, the HSMM generates l independent observations from B on each visit, where l is drawn according to $D_{sl} = p(l|s)$. The added flexibility is useful when modeling human activities, since the duration of stay in a state is restricted to be geometric (and therefore biased to small values) in HMMs. In what follows, we assume that D is over a finite set (of size $|D|$) of durations, where the longest duration is L steps.

The RMAP problem is: given HSMM (A, B, D, π), observations $\hat{y}^T$ and limit K, find $\underset{t, \hat{s} \in Q_A^t, \hat{y} \in O_B^t, \hat{y} \text{ k-edit of } \hat{y}^T}{\text{argmax}} \; p(\hat{s}, \hat{y})$.

Init., Term., Bactracking: See table 1.
Iteration: $1 \le t \le TK + T, \;\; k = a_t, a_t + 1, \dots, t$

$$(\psi_{tik})\delta_{tik} = \underset{1 \le \tau \le t, j, \kappa, y_{\kappa k \tau t i}}{\text{(arg)max}} \; \delta_{\tau j \kappa} p(y_{\kappa k \tau t i}) A_{ji} D_{i|y_{\kappa k \tau t i}|}$$

Table 2: The k-Edit Viterbi Algorithm for HSMMs

Table 2 specifies a variant of KEV to solve the problem, and figure 2 shows a trellis for this algorithm. The trellis is identical to that used by KEV (we represent k added nodes with a single small circle), only its use is different. We focus on how δ_{tik} is calculated. At each timestep t, state i and edit distance k, as with KEV, we iterate over previous timesteps, states and edit distances τ, j and κ. However, this time instead of discarding the observations in the intervening timesteps, we seek their sub-sequence $y_{\kappa k \tau t i}$. We assume that step t only ends a stay in state q_i that begins immediately after the stay in q_j that ended in step τ. If $e_{\tau t}$ is the number of edits in $y_{\kappa k \tau t i}$ (added nodes included + original nodes ignored), we require $\kappa + a_t + e_{\tau t} = k$. The problem of maximizing the likelihood of the path ending at (i, t) then reduces to the problem of finding $y_{\kappa k \tau t i}$ maximizing $p(y_{\kappa k \tau t i}) D_{i|y_{\kappa k \tau t i}|}$.

We find this maximum by iterating through durations l in D_i; for each l, we iterate through predecessors (τ, j, κ) of (t, i), finding a sequence $y_{\kappa k \tau t i}$ of length l with the highest probability; we keep a running tally of the maximum $p(y_{\kappa k \tau t i}) D_{il}$. Finding $y_{\kappa k \tau t i}$ reduces to identifying N_A added nodes (to include in $y_{\kappa k \tau t i}$) and N_O original nodes (to ignore), such that $N_A + N_O = k - \kappa - a_t$ (to satisfy the k-edit criterion), and $N_A + (([t] - [\tau]) - N_O) = l$ (to satisfy the duration constraint). The two equations fix N_A and N_O. Since all the added nodes have the same probability $\dot{y}_i = p(\hat{o}_i | q_i)$, it doesn't matter which particular N_A we pick. On the other hand, we pick the N_O original nodes with lowest probability of observation for exclusion; this can be done by sorting the original nodes in $O(T \log T)$ time *offline*, with $O(L)$ access during execution. Once the sequence of nodes is picked to get $y_{\kappa k \tau t i}$, we simply multiply their observation and transition probabilties together to get $p(y_{\kappa k \tau t i})$, a process that takes l operations, since $|y_{\kappa k \tau t i}| = l = O(L)$.

Given $O(TNK)$ trellis nodes, computing $O(K)$ δ and ψ values at each node, consulting $O(|D|NK)$ preceding values for each value, and spending $O(L)$ for each preceding value considered, the entire algorithm takes $O(TN^2|D|LK^3)$ steps. Note that in the (fairly) common case that D and L are unbounded, this running time becomes $O(T^3N^2K^3)$.

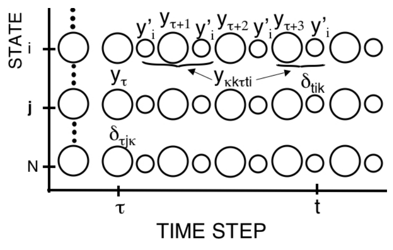

Figure 2: Trellis for KEV on HSMMs

5 Trace Repair for Constrained HMMs

We define a *temporally constrained HMM (TCHMM)* as $\lambda = (A, B, \mathcal{C}, \pi)$, where $\mathcal{C}$ is a *temporal constraint* of the form $\phi_1 \mathcal{E} \phi_2 \mathcal{E} \ldots \mathcal{E} \phi_{|\mathcal{C}|}$. The ϕ_i are propositional boolean formulas over state labels l and observations y: $\phi ::= \text{state}(l) | \text{obs}(y) | \phi \wedge \phi | \neg\phi$. Path suffix $s_i \ldots s_T$ and observations $y_i \ldots y_T$ satisfy the constraint suffix $C_j = \phi_j \ldots \phi_W$ if for any $k \geq j$, $\phi_j(s_k, y_k)$ implies that $(s_{k+1} \ldots s_T, y_{k+1} \ldots y_T)$ satisfy C_{j+1}, written $(s_{k+1} \ldots s_T, y_{k+1} \ldots y_T) \vdash C_j$. Intuitively if one formula in the constraint sequence is true w.r.t. the head of the state/observation sequences, then the formulas that follow must also eventually be true in their specified order later in the sequences. The constraint $(\text{state(COOK)} \wedge \text{obs(oil)}) \mathcal{E} (\text{state(WASH)} \wedge \text{obs(soap)})$ could, for instance capture the constraint that if oil is used in the cooking step of making dinner, soap should be used in the eventual required washing step.

The RMAP problem may now be reformulated as given TCHMM with constraints $\mathcal{C}$, observations $\hat{y}^T$ and limit K, find $SY = \underset{t, \hat{s} \in Q_A^t, \hat{y} \in O_B^t, \hat{y} \text{ k-edit of } \hat{y}^T}{\text{argmax}} \phi(\hat{s}, \hat{y})$ such that $SY \vdash \mathcal{C}$.

Our solution for RMAP estimation is restricted to formulas of the form $\phi ::= \text{state}(l) | \phi \wedge \phi | \neg\phi$ (we disallow dependences on observables). A small modification to the KEV algorithm enables polynomial time solution of this problem. We use the same trellis as in KEV. For each timestep t, state i and edit distance k, we also now maintain an additional $|\mathcal{C}|$-vector. An element δ_{tikm} with $0 \leq m < |\mathcal{C}|$ represents the likelihood of the MAP path ending at state i in time slice t with $(k, \#t)$ edits that still requires constraint suffix C_{m+1} to be satisfied (except δ_{tik0}, which has no outstanding constraints to be satisfied). This likelihood can be computed compositionally from $\delta_{\tau j \kappa \mu}$, with $\tau < t$ and (κ, μ) pointwise $\leq (k, m)$ in $O(TN^2k^3|C|^2P)$ steps, where formulas ϕ_i can be evaluated in $O(P)$ steps (where P is the size of the formulas).

Even MAP estimation (without trace perturbation) for TCHMM's has apparently neither been formulated nor solved previously, although it is potentially quite powerful. For instance, the constrained inference work of Culotta *et. al.* [2] is a special case of TCHMM k-edit MAP estimation (with $k = 0$, and $\mathcal{C} = \text{state}(q_0)\mathcal{E}\text{state}(q_i)$). MAP estimation is a special case of RMAP estimation with $k = 0$. Our variant of KEV above therefore performs MAP estimation. Interestingly with $k = 0$, we can allow the more general version of formulas ϕ and still retain the fast running time. It is open how general $\mathcal{C}$ can be while remaining tractable. For instance, our constraints can be viewed as a fragment of Linear Temporal Logic (LTL) [1]. It is interesting to consider larger fragments as candidates.

6 Evaluation

How does model choice affect advice? The k-edit Viterbi algorithm dispenses advice based on the parameters of its activity models. The credibility of this advice will suffer from any differences between these models and the reality they represent. In order to illustrate this point, we conducted two experiments over three different activity models. The two experiments compare a regular HMM and a time-sensitive HSMM, and a regular HSMM with an HSMM that has temporal logic constraints, respectively.

First, we compare the output of HMMs versus HSMMs on three activity traces from different activity models (see the top row of Figure 3). Each activity trace was intentionally made incorrect: for making tea, the preparation step was hurried; for making a sandwich, not enough ingredients were collected; and for grooming, brushing teeth and combing hair were performed too rapidly. In the top row of Figure 3, we plotted the maximum likelihood values at each step of the activity traces (where a "step" is considered to be a state transition). HMMs fail to detect any problem, exhibiting high likelihood. However, HSMM likelihoods plummet, due to sensitivity to the amount of time spent in each state. The KEDIT trace correctly adds the proper number of observations to each state, resulting in a high likelihood.

Second, we compare the output of HSMMs with and without temporal logic constraints (TLCs) (see the bottom row of Figure 3). Again, we intentionally chose incorrect sequences for the three activities: for making tea, the stove is turned on but never turned off; for preparing a sandwich, the refrigerator door is opened and never closed; and for grooming, the sink water is turned on and never turned back off. In the top row of Figure 3, we plotted the maximum likelhood values at each step of the activity traces. Regular HSMMs fail to detect any problem, reporting high likelihood. HSMMs with TLCs report low likelihood, because they are only allowed to consider state-transitions which satisfy all constraints. In these traces, constraints are broken and alternate, low-likelihood, paths must be considered. The KEDIT trace correctly adds the necessary steps (i.e., turn off stove, shut refrigerator, and turn off sink), resulting in high likelihood.

How does the rating change as k increases? The k-edits Viterbi algorithm provides advice for up to K edits. Ideally, we desire a trace that is above the likelihood threshold with the minimum number of edits. One method is to incrementally increase k until the threshold is exceeded. For this reason, we are interested in how the likelihood changes as k increases.

In this experiment we ran k-edits Viterbi for HSMMs on an empty trace of the "making tea" activity. In Figure 4 we plotted the overall likelihood of each trace as the number of possible edits was increased. The dashed line is a threshold showing the likelihood of an acceptable "good" trace. Obviously, the original empty sequence had low likelihood. As k was increased from one to three, the algorithm was forced to assemble partially complete activity traces which had even lower overall likelihood. When $k = 4$ the algorithm formed a complete trace and met the threshold. As k increased further, the algorithm tweaked the sequence for a slightly higher likelihood. The most likely possible path was reached at $k = 6$. Afterwards, we see an "odd-even" effect as the algorithm is forced to add new (less likely) observations, and then opportunistically delete other observations. For $k \geq 9$, the likelihood drops as the algorithm performs too many modifications to the trace and is unable to reach the optimal solution.

How intuitive is the advice? We now examine the advice dispensed by k-edits Viterbi in several scenarios. We ran the

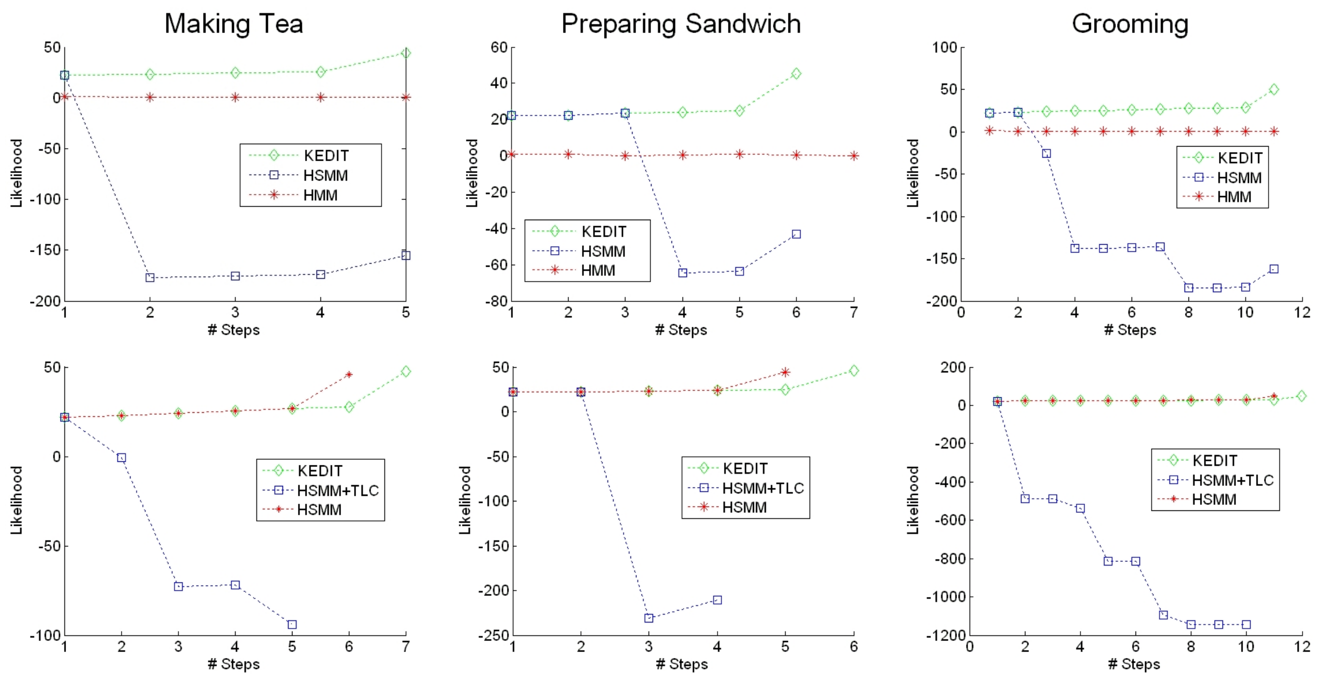

Figure 3: One activity per column; top row compares HMMs to HSMMs, bottom to TCHMMs

algorithm on activity traces that had the following problems: restarted the activity, got two steps out of order, performed a step too quickly, and missed or sped through several non-consecutive steps. All traces are from the "making tea" activity and likelihoods are reported using the optimal number of edits (i.e., k value).

The beginning steps of the next trace were performed twice (i.e., a "start" and a "restart"). The algorithm finds the maximum likelihood solution by deleting extra observations. However, the algorithm did not delete the entire start or restart, but decided to "pick and choose" among the two, keeping the best observations of both. In contrast, our intuition would be to advise the user to keep either the start or the restart. Similarly, in a trace in which two steps were performed out of order, the algorithm deletes one of the misordered steps and inserts new steps in the correct position. We found this to be less intuitive than simply telling the user to switch the two steps to the correct order.

In the next trace, one step was performed too quickly: the "preparation step" only generated one observation, when it should have generated at least two. The algorithm suggested new observations that corrected the amount of time is spent in the state. However, the algorithm will always suggest the most likely observation from the state, because this maximizes the overall likelihood. At first glance, we found this suggestion strategy to be non-intuitive (although mathematically optimal), however, it became a non-issue for models in which observations were spread across multiple states.

In the last trace several non-contiguous steps were missed entirely or performed too quickly. As k increased, the algorithm first chose to insert states that had been missed entirely, and then to add more observations to states that had been visited too briefly. In other words, the algorithm advises the user to at least visit each step of the activity before it advises how to perfect each step. This "top-down" approach fits with our intuition of how advice should be given.

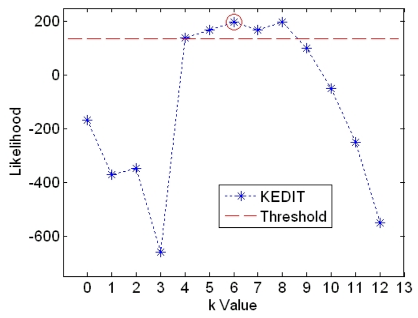

Figure 4: The likelihood of KEDIT traces as k increases.

7 Conclusions

In this paper, we describe the credible activity rating problem. We introduced the k-edits Viterbi algorithm and showed that given model parameters and an activity trace it can provide optimally repaired traces with from zero to k edits. We improved the algorithm by incorporating high-level temporal logic constraints. Finally, we evaluated the strengths and limitations of the algorithm on data from three activity models.

References

[1] E.M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, Boston, 2000.

[2] A. Culotta and A. McCallum. Confidence estimation for information extraction. In *Proc. of Human Lang. Tech. Conf. (HLT-NAACL)*, 2004.

[3] M. Ostendorf, V. V. Digalakis, and O. A. Kimball. From hmms to segment models: A unified view of stochastic modeling for speech recognition. *IEEE Trans. Speech Audio Process*, 4(5):360–378, 1996.

[4] M. Philipose, K.P. Fishkin, M. Perkowitz, D.J. Patterson, H. Kautz, and D. Hahnel. Inferring activities from interactions with objects. *IEEE Pervasive Computing Magazine*, 3(4):50–57, 2004.

[5] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.