# Learning Scripts as Hidden Markov Models

**J. Walker Orr, Prasad Tadepalli, Janardhan Rao Doppa, Xiaoli Fern, Thomas G. Dietterich**
{orr,tadepall,dopa,xfern}@eecs.oregonstate.edu, tgd@oregonstate.edu
School of EECS, Oregon State Univserity, Corvallis OR 97331

## Abstract

Scripts have been proposed to model the stereotypical event sequences found in narratives. They can be applied to make a variety of inferences including filling gaps in the narratives and resolving ambiguous references. This paper proposes the first formal framework for scripts based on Hidden Markov Models (HMMs). Our framework supports robust inference and learning algorithms, which are lacking in previous clustering models. We develop an algorithm for structure and parameter learning based on Expectation Maximization and evaluate it on a number of natural datasets. The results show that our algorithm is superior to several informed baselines for predicting missing events in partial observation sequences.

## 1 Introduction

Scripts were developed as a means of representing stereotypical event sequences and interactions in narratives. The benefits of scripts for encoding common sense knowledge, filling in gaps in a story, resolving ambiguous references, and answering comprehension questions have been amply demonstrated in the early work in natural language understanding (Schank and Abelson 1977). The earliest attempts to learn scripts were based on explanation-based learning, which can be characterized as example-guided deduction from first principles (DeJong 1981; DeJong and Mooney 1986). While this approach is successful in generalizing from a small number of examples, it requires a strong domain theory, which limits its applicability.

More recently, some new graph-based algorithms for inducing script-like structures from text have emerged. "Narrative Chains" is a narrative model similar to Scripts (Chambers and Jurafsky 2008). Each Narrative Chain is a directed graph indicating the most frequent temporal relationship between the events in the chain. Narrative Chains are learned by a novel application of pairwise mutual information and temporal relation learning. Another graph learning approach employs Multiple Sequence Alignment in conjunction with a semantic similarity function to cluster sequences of event descriptions into a directed graph (Regneri, Koller, and Pinkal 2010). More recently still, graphical models have been proposed for representing script-like knowledge, but these lack the temporal component that is central to this paper and to the early script work. These models instead focus on learning bags of related events (Chambers 2013; Kit Cheung, Poon, and Vanderwende 2013).

While the above approaches demonstrate the learnability of script-like knowledge, they do not offer a probabilistic framework to reason robustly under uncertainty taking into account the temporal order of events. In this paper we present the first formal representation of scripts as Hidden Markov Models (HMMs), which support robust inference and effective learning algorithms. The states of the HMM correspond to event types in scripts, such as entering a restaurant or opening a door. Observations correspond to natural language sentences that describe the event instances that occur in the story, e.g., "John went to Starbucks. He came back after ten minutes." The standard inference algorithms, such as the Forward-Backward algorithm, are able to answer questions about the hidden states given the observed sentences, for example, "What did John do in Starbucks?"

There are two complications that need to be dealt with to adapt HMMs to model narrative scripts. First, both the set of states, i.e., event types, and the set of observations are not pre-specified but are to be learned from data. We assume that the set of possible observations and the set of event types to be bounded but unknown. We employ the clustering algorithm proposed in (Regneri, Koller, and Pinkal 2010) to reduce the natural language sentences, i.e., event descriptions, to a small set of observations and states based on their Wordnet similarity.

The second complication of narrative texts is that many events may be omitted either in the narration or by the event extraction process. More importantly, there is no indication of a time lapse or a gap in the story, so the standard forward-backward algorithm does not apply. To account for this, we allow the states to skip generating observations with some probability. This kind of HMMs, with insertions and gaps, have been considered previously in speech processing (Bahl, Jelinek, and Mercer 1983) and in computational biology (Krogh et al. 1994). We refine these models by allowing state-dependent missingness, without introducing additional

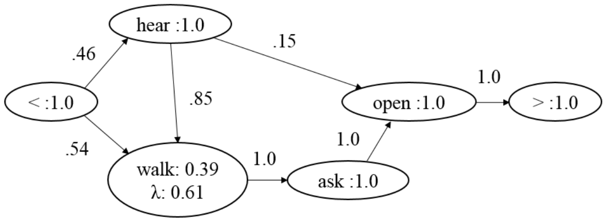

Figure 1: A portion of a learned "Answer the Doorbell" script

"insert states" or "delete states" as in (Krogh et al. 1994). In this paper, we restrict our attention to the so-called "Left-to-Right HMMs" which have acyclic graphical structure with possible self-loops, as they support more efficient inference algorithms than general HMMs and suffice to model most of the natural scripts.

We consider the problem of learning the structure and parameters of scripts in the form of HMMs from sequences of natural language sentences. Our solution to script learning is a novel bottom-up method for structure learning, called *SEM-HMM*, which is inspired by Bayesian Model Merging (BMM) (Stolcke and Omohundro 1994) and Structural Expectation Maximization (SEM) (Friedman 1998). It starts with a fully enumerated HMM representation of the event sequences and incrementally merges states and deletes edges to improve the posterior probability of the structure and the parameters given the data. We compare our approach to several informed baselines on many natural datasets and show its superior performance. We believe our work represents the first formalization of scripts that supports probabilistic inference, and paves the way for robust understanding of natural language texts.

## 2 Problem Setup

Consider an activity such as answering the doorbell. An example HMM representation of this activity is illustrated in Figure 1. Each box represents a state, and the text within is a set of possible event descriptions (i.e., observations). Each event description is also marked with its conditional probability. Each edge represents a transition from one state to another and is annotated with its conditional probability.

In this paper, we consider a special class of HMMs with the following properties. First, we allow some observations to be missing. This is a natural phenomenon in text, where not all events are mentioned or extracted. We call these null observations and represent them with a special symbol $\lambda$. Second, we assume that the states of the HMM can be ordered such that all transitions take place only in that order. These are called Left-to-Right HMMs in the literature (Rabiner 1989; Bahl, Jelinek, and Mercer 1983). Self-transitions of states are permitted and represent "spurious" observations or events with multi-time step durations. While our work can be generalized to arbitrary HMMs, we find that the Left-to-Right HMMs suffice to model scripts in our corpora.

Formally, an HMM is a 4-tuple $(Q, T, O, \Omega)$, where $Q$ is a set of states, $T(q'|q)$ is the probability of transition from $q$ to $q'$, $O$ is a set of possible non-null observations, and $\Omega(o|q)$ is the probability of observing $o$ when in state $q$[1], where $o \in O \cup \{\lambda\}$, and $q_n$ is the terminal state. An HMM is Left-to-Right if the states of the HMM can be ordered from $q_0$ thru $q_n$ such that $T(q_j|q_i)$ is non-zero only if $i \leq j$. We assume that our target HMM is Left-to-Right. We index its states according to a topological ordering of the transition graph. An HMM is a generative model of a distribution over sequences of observations. For convenience w.l.o.g. we assume that each time it is "run" to generate a sample, the HMM starts in the same initial state $q_0$, and goes through a sequence of transitions according to $T$ until it reaches the same final state $q_n$, while emitting an observation in $O \cup \{\lambda\}$ in each state according to $\Omega$. The initial state $q_0$ and the final state $q_n$ respectively emit the distinguished observation symbols, "$<$" and "$>$" in $O$, which are emitted by no other state. The concatenation of observations in successive states constitutes a sample of the distribution represented by the HMM. Because the null observations are removed from the generated observations, the length of the output string may be smaller than the number of state transitions. It could also be larger than the number of *distinct* state transitions, since we allow observations to be generated on the self transitions. Thus spurious and missing observations model insertions and deletions in the outputs of HMMs without introducing special states as in profile HMMs (Krogh et al. 1994).

In this paper we address the following problem. Given a set of narrative texts, each of which describes a stereotypical event sequence drawn from a fixed but unknown distribution, learn the structure and parameters of a Left-to-Right HMM model that best captures the distribution of the event sequences. We evaluate the algorithm on natural datasets by how well the learned HMM can predict observations removed from the test sequences.

## 3 HMM-Script Learning

At the top level, the algorithm is input a set of documents $D$, where each document is a sequence of natural language sentences that describes the same stereotypical activity. The output of the algorithm is a Left-to-Right HMM that represents that activity.

Our approach has four main components, which are described in the next four subsections: Event Extraction, Parameter Estimation, Structure Learning, and Structure Scoring. The event extraction step clusters the input sentences into event types and replaces the sentences with the corresponding cluster labels. After extraction, the event sequences are iteratively merged with the current HMM in batches of size $r$ starting with an empty HMM. Structure Learning then merges pairs of states (nodes) and removes state transitions (edges) by greedy hill climbing guided by the improvement in approximate posterior probability of the

[1] $\Omega$ can be straightforwardly generalized to depend on both of the states in a state transition.

HMM. Once the hill climbing converges to a local optimum, the maxmimum likelihood HMM parameters are re-estimated using the EM procedure based on all the data seen so far. Then the next batch of $r$ sequences are processed. We will now describe these steps in more detail.

### 3.1 Event Extraction

Given a set of sequences of sentences, the event extraction algorithm clusters them into events and arranges them into a tree structured HMM. For this step, we assume that each sentence has a simple structure that consists of a single verb and an object. We make the further simplifying assumption that the sequences of sentences in all documents describe the events in temporal order. Although this assumption is often violated in natural documents, we ignore this problem to focus on script learning. There have been some approaches in previous work that specifically address the problem of inferring temporal order of events from texts, e.g., see (Raghavan, Fosler-Lussier, and Lai 2012).

Given the above assumptions, following (Regneri, Koller, and Pinkal 2010), we apply a simple agglomerative clustering algorithm that uses a semantic similarity function over sentence pairs $Sim(S_1, S_2)$ given by $w_1 PS(V_1, V_2) + w_2 PS(O_1, O_2)$, where $V_i$ is the verb and $O_i$ is the object in the sentence $S_i$. Here $PS(w, v)$ is the path similarity metric from Wordnet (Miller 1995). It is applied to the first verb (preferring verbs that are not stop words) and to the objects from each pair of sentences. The constants $w_1$ and $w_2$ are tuning parameters that adjust the relative importance of each component. Like (Regneri, Koller, and Pinkal 2010), we found that a high weight on the verb similarity was important to finding meaningful clusters of events. The most frequent verb in each cluster is extracted to name the event type that corresponds to that cluster.

The initial configuration of the HMM is a Prefix Tree Acceptor, which is constructed by starting with a single event sequence and then adding sequences by branching the tree at the first place the new sequence differs from it (Dupont, Miclet, and Vidal 1994; Seymore, McCallum, and Rosenfeld 1999). By repeating this process, an HMM that fully enumerates the data is constructed.

### 3.2 Parameter Estimation with EM

In this section we describe our parameter estimation methods. While parameter estimation in this kind of HMM was treated earlier in the literature (Rabiner 1989; Bahl, Jelinek, and Mercer 1983), we provide a more principled approach to estimate the state-dependent probability of $\lambda$ transitions from data without introducing special insert and delete states (Krogh et al. 1994). We assume that the structure of the Left-to-Right HMM is fixed based on the preceding structure learning step, which is described in Section 3.3.

The main difficulty in HMM parameter estimation is that the states of the HMM are not observed. The Expectation-Maximization (EM) procedure (also called the Baum-Welch algorithm in HMMs) alternates between estimating the hidden states in the event sequences by running the Forward-Backward algorithm (the Expectation step) and finding the maximum likelihood estimates (the Maximization step) of the transition and observation parameters of the HMM (Baum et al. 1970). Unfortunately, because of the $\lambda$-transitions the state transitions of our HMM are not necessarily aligned with the observations. Hence we explicitly maintain two indices, the time index $t$ and the observation index $i$. We define $\alpha_{q_j}(t, i)$ to be the joint probability that the HMM is in state $q_j$ at time $t$ and has made the observations $\vec{o}_{0,i}$. This is computed by the forward pass of the algorithm using the following recursion. Equations 1 and 2 represent the base case of the recursion, while equation 3 represents the case for null observations. Note that the observation index $i$ of the recursive call is not advanced unlike in the second half of equation 3 where it is advanced for a normal observation. We exploit the fact that the HMM is Left-to-Right and only consider transitions to $j$ from states with indices $k \leq j$. The time index $t$ is incremented starting 0, and the observation index $i$ varies from 0 thru $m$.

$$\alpha_{q_0}(0, 0) = 1 \tag{1}$$

$$\forall j > 0, \alpha_{q_j}(0, 0) = 0 \tag{2}$$

$$\alpha_{q_j}(t, i) = \sum_{0 \leq k \leq j} T(q_j|q_k)\{\Omega(\lambda|q_j)\alpha_{q_k}(t-1, i) + \Omega(o_i|q_j)\alpha_{q_k}(t-1, i-1)\} \tag{3}$$

The backward part of the standard Forward-Backward algorithm starts from the last time step $\tau$ and reasons backwards. Unfortunately in our setting, we do not know $\tau$—the true number of state transitions—as some of the observations are missing. Hence, we define $\beta_{q_j}(t, i)$ as the conditional probability of observing $\vec{o}_{i+1,m}$ in the remaining $t$ steps given that the current state is $q_j$. This allows us to increment $t$ starting from 0 as recursion proceeds, rather than decrementing it from $\tau$.

$$\beta_{q_n}(0, m) = 1 \tag{4}$$

$$\forall j < n, \beta_{q_j}(0, m) = 0 \tag{5}$$

$$\beta_{q_j}(t, i) = \sum_{j \leq k} T(q_k|q_j)\{\Omega(\lambda|q_k)\beta_{q_k}(t-1, i) + \Omega(o_{i+1}|q_k)\beta_{q_k}(t-1, i+1)\} \tag{6}$$

Equation 7 calculates the probability of the observation sequence $z = P(\vec{o})$, which is computed by marginalizing $\alpha_q(t, m)$ over time $t$ and state $q$ and setting the second index $i$ to the length of the observation sequence $m$. The quantity $z$ serves as the normalizing factor for the last three equations.

$$z = P(\vec{o}) = \sum_{q \in Q} \sum_{t} \alpha_q(t, m) \tag{7}$$

$$\gamma_q(t, i) = P(q|\vec{o}) = z^{-1} \sum_{\tau} \alpha_q(t, i)\beta_q(\tau - t, i) \tag{8}$$

$$\delta_{q,q'\uparrow\lambda}(t) = P(q \to q', \lambda|\vec{o}) = z^{-1} T(q'|q)\Omega(\lambda|q') \sum_{\tau} \sum_{i} \{\alpha_q(t, i)\beta_{q'}(\tau - t - 1, i)\} \tag{9}$$

$$\forall o \in \Omega, \delta_{q,q'\uparrow o}(t) = P(q \to q', o|\vec{o}) = z^{-1} T(q'|q)\Omega(o|q') \sum_{\tau} \sum_{i} \{\alpha_q(t, i) I(o_{i+1} = o)\beta_{q'}(\tau - t - 1, i + 1)\} \tag{10}$$

Equation 8, the joint distribution of the state and observation index $\gamma$ at time $t$ is computed by convolution, i.e., multiplying the $\alpha$ and $\beta$ that correspond to the same time step and the same state and marginalizing out the length of the state-sequence $\tau$. Convolution is necessary, as the length of the state-sequence $\tau$ is a random variable equal to the sum of the corresponding time indices of $\alpha$ and $\beta$.

Equation 9 computes the joint probability of a state-transition associated with a null observation by first multiplying the state transition probability by the null observation probability given the state transition and the appropriate $\alpha$ and $\beta$ values. It then marginalizes out the observation index $i$. Again we need to compute a convolution with respect to $\tau$ to take into account the variation over the total number of state transitions. Equation 10 calculates the same probability for a non-null observation $o$. This equation is similar to equation 9 with two differences. First, we ensure that the observation is consistent with $o$ by multiplying the product with the indicator function $I(o_{i+1} = o)$ which is 1 if $o_{i+1} = o$ and 0 otherwise. Second, we advance the observation index $i$ in the $\beta$ function.

Since the equations above are applied to each individual observation sequence, $\alpha$, $\beta$, $\gamma$, and $\delta$ all have an implicit index $s$ which denotes the observation sequence and has been omitted in the above equations. We will make it explicit below and calculate the expected counts of state visits, state transitions, and state transition observation triples.

$$\forall q \in Q, C(q) = \sum_{s,t,i} \gamma_q(s,t,i) \tag{11}$$

$$\forall q, q' \in Q, C(q \to q') = \sum_{s,t,o \in \Omega \bigcup \{\lambda\}} \delta_{q,q' \uparrow o}(s,t) \tag{12}$$

$$\forall q, q' \in Q, o \in \Omega \bigcup \{\lambda\}, \qquad C(q, q' \uparrow o) = \sum_{s,t} \delta_{q,q' \uparrow o}(s,t) \tag{13}$$

Equation 11 counts the total expected number of visits of each state in the data. Also, Equation 12 estimates the expected number of transitions between each state pair. Finally, Equation 13 computes the expected number of observations and state-transitions including null transitions. This concludes the E-step of the EM procedure.

The M-step of the EM procedure consists of Maximum Aposteriori (MAP) estimation of the transition and observation distributions is done assuming an uninformative Dirichlet prior. This amounts to adding a pseudo-count of 1 to each of the next states and observation symbols. The observation distributions for the initial and final states $q_0$ and $q_n$ are fixed to be the Kronecker delta distributions at their true values.

$$\hat{T}(q'|q) = \frac{C(q \to q') + 1}{[C(q) + \sum_{p' \in Q} 1]} \tag{14}$$

$$\hat{\Omega}(o|q') = \frac{\sum_q C(q, q' \uparrow o) + 1}{\sum_{o'} \{\sum_q C(q, q' \uparrow o')\} + 1} \tag{15}$$

The E-step and the M-step are repeated until convergence of the parameter estimates.

### 3.3 Structure Learning

We now describe our structure learning algorithm, SEM-HMM. Our algorithm is inspired by Bayesian Model Merging (BMM) (Stolcke and Omohundro 1994) and Structural EM (SEM) (Friedman 1998) and adapts them to learning HMMs with missing observations. SEM-HMM performs a greedy hill climbing search through the space of acyclic HMM structures. It iteratively proposes changes to the structure either by merging states or by deleting edges. It evaluates each change and makes the one with the best score. An exact implementation of this method is expensive, because, each time a structure change is considered, the MAP parameters of the structure given the data must be re-estimated. One of the key insights of both SEM and BMM is that this expensive re-estimation can be avoided in factored models by incrementally computing the changes to various expected counts using only local information. While this calculation is only approximate, it is highly efficient.

During the structure search, the algorithm considers every possible structure change, i.e., merging of pairs of states and deletion of state-transitions, checks that the change does not create cycles, evaluates it according to the scoring function and selects the best scoring structure. This is repeated until the structure can no longer be improved (see Algorithm 1).

**Algorithm 1**

```
procedure LEARN(Model $M$, Data $D$, Changes $S$)
    while $NotConverged$ do
        $\mathcal{M}$ = AcyclicityFilter $(S(M))$
        $M^* = argmax_{M' \in \mathcal{M}} P(M'|D)$
        if $P(M^*|D) \leq P(M|D)$ then
            return $M$
        else
            $M = M^*$
        end if
    end while
end procedure
```

The Merge States operator creates a new state from the union of a state pair's transition and observation distributions. It must assign transition and observation distributions to the new merged state. To be exact, we need to redo the parameter estimation for the changed structure. To compute the impact of several proposed changes efficiently, we assume that all probabilistic state transitions and trajectories for the observed sequences remain the same as before except in the changed parts of the structure. We call this "locality of change" assumption, which allows us to add the corresponding expected counts from the states being merged as shown below.

$$\begin{aligned}
C(r) &= C(p) + C(q) \\
C(r \to s) &= C(p \to s) + C(q \to s) \\
C(s \to r) &= C(s \to p) + C(s \to q) \\
C(r, s \uparrow o) &= C(p, s \uparrow o) + C(q, s \uparrow o) \\
C(s, r \uparrow o) &= C(s, p \uparrow o) + C(s, q \uparrow o)
\end{aligned}$$

The second kind of structure change we consider is edge deletion and consists of removing a transition between two states and redistributing its evidence along the other paths between the same states. Again, making the locality of change assumption, we only recompute the parameters of the affected transition and observation distributions. Hence, we re-estimate the parameters due to deleting an edge $(q_s, q_e)$, by effectively redistributing the expected transitions from $q_s$ to $q_e$, $C(q_s \rightarrow q_e)$, among other edges between $q_s$ and $q_e$ based on the parameters of the current model.

This is done efficiently using a procedure similar to the Forward-Backward algorithm under the null observation sequence. Algorithm 3.3 takes the current model $M$, an edge $(q_s \rightarrow q_e)$, and the expected count of the number of transitions from $q_s$ to $q_e$, $N = C(q_s \rightarrow q_e)$, as inputs. It updates the counts of the other transitions to compensate for removing the edge between $q_s$ and $q_e$. It initializes the $\alpha$ of $q_s$ and the $\beta$ of $q_e$ with 1 and the rest of the $\alpha$s and $\beta$s to 0. It makes two passes through the HMM, first in the topological order of the nodes in the graph and the second in the reverse topological order. In the first, "forward" pass from $q_s$ to $q_e$, it calculates the $\alpha$ value of each node $q_i$ that represents the probability that a sequence that passes through $q_s$ also passes through $q_i$ while emitting no observation. In the second, "backward" pass, it computes the $\beta$ value of a node $q_i$ that represents the probability that a sequence that passes through $q_i$ emits no observation and later passes through $q_e$. The product of $\alpha(q_i)$ and $\beta(q_i)$ gives the probability that $q_i$ is passed through when going from $q_s$ to $q_t$ and emits no observation. Multiplying it by the expected number of transitions $N$ gives the expected number of additional counts which are added to $C(q_i)$ to compensate for the deleted transition $(q_s \rightarrow q_e)$. After the distribution of the evidence, all the transition and observation probabilities are re-estimated for the nodes and edges affected by the edge deletion

**Algorithm 2** Forward-Backward algorithm to delete an edge and re-distribute the expected counts.

```
procedure DELETEEDGE(Model $M$, edge $(q_s \rightarrow q_e)$, count $N$)
    $\forall i s.t. s \leq i \leq e, \alpha(q_i) = \beta(q_i) = 0$
    $\alpha(q_s) = \beta(q_e) = 1$
    for $i = s + 1$ to $e$ do
        for all $q_p \in Parents(q_i)$ do
            $\alpha(q_p \rightarrow q_i) = \alpha(q_p)T(q_i|q_p)\Omega(\lambda|q_i)$
            $\alpha(q_i) = \alpha(q_i) + \alpha(q_p \rightarrow q_i)$
        end for
    end for
    for $i = e - 1$ downto $s$ do
        for all $q_c \in Children(q_i)$ do
            $\beta(q_i \rightarrow q_c) = \beta(q_c)T(q_c|q_i)\Omega(\lambda|q_c)$
            $C(q_i \rightarrow q_c)$ += $\alpha(q_i \rightarrow q_c)\beta(q_i \rightarrow q_c)N$
            $C(q_i)$ += $C(q_i \rightarrow q_c)$
            $\beta(q_i) = \beta(q_i) + \beta(q_i \rightarrow q_c)$
        end for
    end for
end procedure
```

In principle, one could continue making incremental structural changes and parameter updates and never run EM again. This is exactly what is done in Bayesian Model Merging (BMM) (Stolcke and Omohundro 1994). However, a series of structural changes followed by approximate incremental parameter updates could lead to bad local optima. Hence, after merging each batch of $r$ sequences into the HMM, we re-run EM for parameter estimation on all sequences seen thus far.

### 3.4 Structure Scoring

We now describe how we score the structures produced by our algorithm to select the best structure. We employ a Bayesian scoring function, which is the posterior probability of the model given the data, denoted $P(M|D)$. The score is decomposed via Bayes Rule (i.e., $P(M|D) \propto P(M)P(D|M)$)), and the denominator is omitted since it is invariant with regards to the model.

Since each observation sequence is independent of the others, the data likelihood $P(D|M) = \Pi_{\vec{o} \in D} P(\vec{o})$ is calculated using the Forward-Backward algorithm and Equation 7 in Section 3.2. Because the initial model fully enumerates the data, any merge can only reduce the data likelihood. Hence, the model prior $P(M)$ must be designed to encourage generalization via state merges and edge deletions (described in Section 3.3). We employed a prior with three components: the first two components are syntactic and penalize the number of states $|Q|$ and the number of non-zero transitions $|T|$ respectively. The third component penalizes the number of frequently-observed semantic constraint violations $|C|$. In particular, the prior probabilty of the model $P(M) = \frac{1}{Z} \exp(-(\kappa_q |Q| + \kappa_t |T| + \kappa_c |C|))$. The $\kappa$ parameters assign weights to each component in the prior.

The semantic constraints are learned from the event sequences for use in the model prior. The constraints take the simple form "$X$ never follows $Y$." They are learned by generating all possible such rules using pairwise permutations of event types, and evaluating them on the training data. In particular, the number of times each rule is violated is counted and a $z$-test is performed to determine if the violation rate is lower than a predetermined error rate. Those rules that pass the hypothesis test with a threshold of 0.01 are included. When evaluating a model, these constraints are considered violated if the model could generate a sequence of observations that violates the constraint.

Also, in addition to incrementally computing the transition and observation counts, $C(r \rightarrow s)$ and $C(r, s \uparrow o)$, the likelihood, $P(D|M)$ can be incrementally updated with structure changes as well. Note that the likelihood can be expressed as $P(D|M) = \prod_{q,r \in Q} \prod_{o \in O} T(r|q)^{C(q \rightarrow r)} \Omega(o|r)^{C(q,r \uparrow o)}$ when the state transitions are observed. Since the state transitions are not actually observed, we approximate the above expression by replacing the observed counts with expected counts. Further, the locality of change assumption allows us to easily calculate the effect of changed expected counts and parameters on the likelihood by dividing it by the old products and multiplying by the new products. We call this version of our algorithm SEM-HMM-Approx.

| Batch Size $r$ | 2 | 5 | 10 |
|---|---|---|---|
| SEM-HMM | 42.2% | 45.1% | 46.0% |
| SEM-HMM Approx. | 43.3% | 43.5% | 44.3% |
| BMM + EM | 41.1% | 41.2% | 42.1% |
| BMM | 41.0% | 39.5% | 39.1% |
| Conditional | | | 36.2% |
| Frequency | | | 27.3% |

Table 1: The average accuracy on the OMICS domains

| Example 1 | Example 2 | Example 3 |
|---|---|---|
| Hear the doorbell. | Listen for the doorbell. | Hear doorbell. |
| Walk to the door. | Go towards the door. | Walk to door. |
| Open the door. | Open the door. | Peek through the hole. |
| Allow the people in. | Greet the visitor. | Open the door. |
| Close the door. | See what the visitor wants. | |
| | Say goodbye to the visitor. | |
| | Close the door. | |

Table 2: Examples from the OMICS "Answer the Doorbell" task with event triggers underlined

## 4 Experiments and Results

We now present our experimental results on SEM-HMM and SEM-HMM-Approx. The evaluation task is to predict missing events from an observed sequence of events. For comparison, four baselines were also evaluated. The "Frequency" baseline predicts the most frequent event in the training set that is not found in the observed test sequence. The "Conditional" baseline predicts the next event based on what most frequently follows the prior event. A third baseline, referred to as "BMM," is a version of our algorithm that does not use EM for parameter estimation and instead only incrementally updates the parameters starting from the raw document counts. Further, it learns a standard HMM, that is, with no $\lambda$ transitions. This is very similar to the Bayesian Model Merging approach for HMMs (Stolcke and Omohundro 1994). The fourth baseline is the same as above, but uses our EM algorithm for parameter estimation without $\lambda$ transitions. It is referred to as "BMM + EM."

The Open Minds Indoor Common Sense (OMICS) corpus was developed by the Honda Research Institute and is based upon the Open Mind Common Sense project (Gupta and Kochenderfer 2004). It describes 175 common household tasks with each task having 14 to 122 narratives describing, in short sentences, the necessary steps to complete it. Each narrative consists of temporally ordered, simple sentences from a single author that describe a plan to accomplish a task. Examples from the "Answer the Doorbell" task can be found in Table 2. The OMICS corpus has 9044 individual narratives and its short and relatively consistent language lends itself to relatively easy event extraction.

The 84 domains with at least 50 narratives and 3 event types were used for evaluation. For each domain, forty percent of the narratives were withheld for testing, each with one randomly-chosen event omitted. The model was evaluated on the proportion of correctly predicted events given the remaining sequence. On average each domain has 21.7 event types with a standard deviation of 4.6. Further, the average narrative length across domains is 3.8 with standard deviation of 1.7. This implies that only a fraction of the event types are present in any given narrative. There is a high degree of omission of events and many different ways of accomplishing each task. Hence, the prediction task is reasonably difficult, as evidenced by the simple baselines. Neither the frequency of events nor simple temporal structure is enough to accurately fill in the gaps which indicates that most sophisticated modeling such as SEM HMM is needed.

The average accuracy across the 84 domains for each method is found in Table 1. On average our method significantly out-performed all the baselines, with the average improvement in accuracy across OMICS tasks between SEM-HMM and each baseline being statistically significant at a .01 level across all pairs and on sizes of $r = 5$ and $r = 10$ using one-sided paired t-tests. For $r = 2$ improvement was not statistically greater than zero. We see that the results improve with batch size $r$ until $r = 10$ for SEM-HMM and BMM+EM, but they decrease with batch size for BMM without EM. Both of the methods which use EM depend on statistics to be robust and hence need a larger $r$ value to be accurate. However for BMM, a smaller $r$ size means it reconciles a couple of documents with the current model in each iteration which ultimately helps guide the structure search. The accuracy for "SEM-HMM Approx." is close to the exact version at each batch level, while only taking half the time on average.

Also, SEM-HMM can be used to get a general estimate of the level of missing data. By averaging the value of $P(\lambda)$, we obtain a rough estimate of the proportion missing events in each OMICS task with the mean being 14.6% with a standard deviation of .1 with the maximum being 61.1%. We compared the accuracy of each predictor against the average estimated missing-level and the trend of all the predictors is down as the level of missing data increases. This result is intuitive as the task of prediction should be become increasinly difficult as information decreases. However, SEM-HMM, while more accurate a lower levels of missing data, eventually fails at high levels of incompleteness just like the other predictors as information becomes increasingly sparse.

## 5 Conclusions

In this paper, we have given the first formal treatment of scripts as HMMs with missing observations. We adapted the HMM inference and parameter estimation procedures to scripts and developed a new structure learning algorithm, SEM-HMM, based on the EM procedure. It improves upon BMM by allowing for $\lambda$ transitions and by incorporating maximum likelihood parameter estimation via EM. We showed that our algorithm is effective in learning scripts from documents and performs better than other baselines on sequence prediction tasks. Thanks to the assumption of missing observations, the graphical structure of the scripts is usually sparse and intuitive. Future work includes learning from more natural text such as newspaper articles, enriching the representations to include objects and relations, and integrating HMM inference into text understanding.

**Acknowledgements and Thanks.** We would like to thank Nate Chambers, Frank Ferraro, and Ben Van Durme for their helpful comments, critism, and feedback. Also we would like to thank the SCALE 2013 workshop. This work was supported by the DARPA and AFRL under contract No. FA8750-13-2-0033. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA, the AFRL, or the US government.

# References

Bahl, L. R.; Jelinek, F.; and Mercer, R. L. 1983. A maximum likelihood approach to continuos speech recognition. *IEEE Transactions in Pattern Analysis and Machine Intelligence* 5(2):179–190.

Baum, L. E.; Petrie, T.; Soules, G.; and Weiss, N. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics* 41(1):164–171.

Chambers, N., and Jurafsky, D. 2008. Unsupervised learning of narrative event chains. *Proceedings of ACL-08: HLT* 789–797.

Chambers, N. 2013. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1797–1807.

DeJong, G., and Mooney, R. 1986. Explanation-based learning: An alternative view. *Machine learning* 1(2):145–176.

DeJong, G. 1981. Generalizations based on explanations. *Urbana* 51:61801.

Dupont, P.; Miclet, L.; and Vidal, E. 1994. What is the search space of the regular inference? In *Grammatical Inference and Applications*. Springer. 25–37.

Friedman, N. 1998. The Bayesian structural EM algorithm. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 129–138. Morgan Kaufmann Publishers Inc.

Gupta, R., and Kochenderfer, M. J. 2004. Common sense data acquisition for indoor mobile robots. In *AAAI*, 605–610.

Kit Cheung, J. C.; Poon, H.; and Vanderwende, L. 2013. Probabilistic frame induction. In *Proceedings of NAACL-HLT 2013*, 837–846.

Krogh, A.; Brown, M.; Mian, I. S.; Sjolander, K.; and Haussler, D. 1994. Hidden markov models in computational biology. *Journal of Molecular Biology* 1501–1531.

Miller, G. A. 1995. WordNet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

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

Raghavan, P.; Fosler-Lussier, E.; and Lai, A. M. 2012. Learning to temporally order medical events in clinical text. In *Proceedings of the 50th Annual Meeting of the ACL*, 70–74. Stroudsburg, PA, USA: Association for Computational Linguistics.

Regneri, M.; Koller, A.; and Pinkal, M. 2010. Learning script knowledge with web experiments. In *Proceedings of* *the 48th Annual Meeting of the Association for Computational Linguistics*, 979–988. Association for Computational Linguistics.

Schank, R., and Abelson, R. 1977. *Scripts, plans, goals and understanding: An inquiry into human knowledge structures.* Lawrence Erlbaum.

Seymore, K.; McCallum, A.; and Rosenfeld, R. 1999. Learning hidden Markov model structure for information extraction. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, 37–42.

Stolcke, A., and Omohundro, S. M. 1994. Best-first model merging for hidden Markov model induction. *arXiv preprint cmp-lg/9405017*.