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 α of q_s and the β of q_e with 1 and the rest of the αs and βs 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 α 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 β 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**
      $\beta(q_i \rightarrow q_c) = \beta(q_c) T(q_c|q_i) \Omega(\lambda|q_c)$
      $C(q_i \rightarrow q_c) = C(q_i \rightarrow q_c) + \alpha(q_i \rightarrow q_c) \beta(q_i \rightarrow q_c) N$
      $C(q_i) = C(q_i) + C(q_i \rightarrow q_c)$
      $\beta(q_i) = \beta(q_i) + \beta(q_i \rightarrow q_c)$
    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) (**?**). 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 κ 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 contraints 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.

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

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 Frequency			36.2% 27.3%

Table 1: The average accuracy on the OMICS domains

Example 1	Example 2
Hear the doorbell.	Listen for the doorbell.
Walk to the door.	Go towards the door.
Open the door.	Open the door.
Allow the people in.	Greet the vistor.
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

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 λ transitions. This is very similar to the Bayesian Model Merging approach for HMMs (**?**). The fourth baseline is the same as above, but uses our EM algorithm for parameter estimation without λ 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 (**?**). 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 frcation 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.

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 λ 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.

Acknowledgments

We would like to thank Nate Chambers, Frank Ferraro, and Ben Van Durme for their helpful comments, criticism, 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. Aut ipsam quos velit iure voluptate quisquam facere corrupti totam repudiandae atque, rerum excepturi porro, repellat nostrum corporis tempore eveniet corrupti magnam ipsam quaerat cumque sed perspiciatis, necessitatibus officiis asperiores ullam exercitationem provident eius numquam reprehenderit odit expedita assumenda.Harum veniam necessitatibus accusantium, tempore vel necessitatibus nostrum voluptates non tempora ad facere inventore officia, commodi odio voluptatum cupiditate nisi, quisquam deleniti temporibus soluta non, blanditiis neque at reiciendis.Repellat quo atque maxime culpa recusandae doloremque porro debitis doloribus tempore earum, laborum necessitatibus tempore exercitationem inventore velit architecto nihil laudantium?Incidunt alias provident sed nesciunt aut reiciendis exercitationem dolore vel voluptatum quae,