'Say EM' for Selecting Probabilistic Models for Logical Sequences

Kristian Kersting
Machine Learning Laboratory
University of Freiburg
Georges-Koehler-Allee 079, 79112 Freiburg, Germany

Tapani Raiko
Laboratory of Computer and Information Science
Helsinki University of Technology
P.O. Box 5400, 02015 HUT, Finland

Abstract

Many real world sequences such as protein secondary structures or shell logs exhibit a rich internal structures. Traditional probabilistic models of sequences, however, consider sequences of flat symbols only. *Logical hidden Markov models* have been proposed as one solution. They deal with logical sequences, i.e., sequences over an alphabet of logical atoms. This comes at the expense of a more complex model selection problem. Indeed, different abstraction levels have to be explored. In this paper, we propose a novel method for selecting logical hidden Markov models from data called SAGEM. SAGEM combines *generalized expectation maximization*, which optimizes parameters, with structure search for model selection using *inductive logic programming* refinement operators. We provide convergence and experimental results that show SAGEM's effectiveness.

1 Introduction

Hidden Markov models [21] (HMMs) are extremely popular for analyzing sequential data. Areas of application include computational biology, user modeling, and robotics. Despite their successes, HMMs have a major weakness: they handle only sequences of flat, i.e., unstructured symbols. In many applications the symbols occurring in sequences are structured. Consider, e.g., the sequence of UNIX commands `emacs lohmms.tex`, `ls`, `latex lohmms.tex`, ... Such data have been used to train HMMs for anomaly detection [15]. However, as the above command sequence shows, UNIX commands may have parameters. Thus, commands are essentially *structured* symbols. HMMs cannot easily deal with this type of structured sequences. Typically, the application of HMMs requires either 1) ignoring the structure of the commands (i.e., the parameters), or 2) taking all possible parameters explicitly into account. The former approach results in serious information loss; the latter in a combinatorial explosion in the number of parameters and, as a consequence, inhibits generalization.

The above sketched problem with HMMs is akin to the problem of dealing with structured examples in traditional machine learning algorithms as studied in the field of inductive logic programming (ILP) [17]. Recently, Kersting et al. [12] proposed *logical HMMs* (LOHMMs) as an *probabilistic* ILP [4] framework that upgrades HMMs to deal with structure. The key idea is to employ logical atoms. Using logical atoms, the above UNIX command sequence can be represented as `emacs(lohmms.tex)`, `ls`, `latex(lohmms.tex)`, ... LOHMMs have been proven to be useful within bioinformatics domains. For instance in [12], the LOHMMs used to discover structural signatures of protein folds were simpler but more effective compared to corresponding HMMs (120 vs. > 62000 parameters). The compactness and comprehensibility, however, comes at the expense of a more complex model selection problem. So far, model selection for LOHMMs has not been investigated. Our main contribution is SAGEM, German for 'say EM', a novel method for selecting LOHMM structures from data. Selecting a structure is a significant problem for many reasons. First, eliciting LOHMMs from experts can be a laborious and expensive process. Second, HMMs are commonly learned by estimating the maximum likelihood parameters of a fixed, fully connected model. Such an approach is not feasible for LOHMMs as different abstraction levels have to be explored. Third, LOHMMs are strictly more expressive than HMMs. In [11], LOHMMs are used to classify tree-structured mRNA data. Finally, the parameter estimation of a LOHMM is a costly nonlinear optimization problem, so the naïve search is infeasible.

SAGEM adapts Friedman's *structural EM* [6]. It combines a *generalized expectation maximization* (GEM) algorithm, which optimizes parameters, with structure search for model selection using ILP refinement operators. Thus, SAGEM explores different abstraction levels due to ILP re-

finement operators, and, due to a GEM approach, it reduces the selection problem to a more efficiently solvable one.

The outline of the paper is as follows. Section 2 reviews LOHMMs and their underlying logical concepts; Section 3 formalizes the model selection problem; in Section 4, we present a naïve learning algorithm; in Section 5, we introduce a structural, generalized EM – called SAGEM – for learning LOHMMs. SAGEM is experimentally evaluated in Section 6. Before concluding we discuss related work.

2 Probabilistic Models for Logical Sequences

We will briefly review *logical Hidden Markov models* (LOHMMs) [12, 13, 11]. The logical component of HMMs corresponds to a *Mealy machine*, i.e., to a finite state machine where the output symbols are associated with transitions. The key idea to develop probabilistic models for structured sequences is to replace these flat symbols by abstract symbols, more precisely logical atoms.

First-Order Predicate Logic: A *first-order logic alphabet* Σ is a set of relation symbols $\mathtt{r}$ with arity $m \geq 0$, written $\mathtt{r/m}$, and a set of function symbols $\mathtt{f}$ with arity $n \geq 0$, written $\mathtt{f/n}$. An *atom* $\mathtt{r}(\mathtt{t}_1, \ldots, \mathtt{t}_m)$ is a relation symbol $\mathtt{r}$ followed by a bracketed m-tuple of terms $\mathtt{t}_i$. A *term* is a variable $\mathtt{V}$ or a function symbol $\mathtt{f}$ of arity n immediately followed by a bracketed n-tuple of terms $\mathtt{s}_j$, i.e., $\mathtt{f}(\mathtt{s}_1, \ldots, \mathtt{s}_n)$. A *definite clause* $\mathtt{A} \leftarrow \mathtt{B}$ consists of atoms A and B and can be read as $\mathtt{A}$ *is true if* $\mathtt{B}$ *is true*. A substitution $\theta = \{\mathtt{V}_1/\mathtt{t}_1, \ldots, \mathtt{V}_k/\mathtt{t}_k\}$, e.g. $\{\mathtt{X}/\mathtt{tex}\}$, is an assignment of terms $\mathtt{t}_i$ to variables $\mathtt{V}_i$. Applying a substitution θ to a term, atom or clause e yields the instantiated term, atom, or clause $e\theta$ where all occurrences of the variables $\mathtt{V}_i$ are simultaneously replaced by the term $\mathtt{t}_i$, e.g. $\mathtt{ls}(\mathtt{X}) \leftarrow \mathtt{emacs}(\mathtt{F}, \mathtt{X})\{\mathtt{X}/\mathtt{tex}\}$ yields $\mathtt{ls}(\mathtt{tex}) \leftarrow \mathtt{emacs}(\mathtt{F}, \mathtt{tex})$. A term, atom or clause e is called *ground* when it contains no variables, i.e., $vars(e) = \emptyset$. The *Herbrand base* of Σ, denoted as hb_Σ, is the set of all ground atoms constructed with the predicate and function symbols in Σ. The set $G_\Sigma(\mathtt{A})$ of an atom $\mathtt{A}$ consists of all ground atoms $\mathtt{A}\theta$ belonging to hb_Σ.

Our running example will be user modeling. For example, $\mathtt{emacs}(\mathtt{readme}, \mathtt{other})$ means that the user of type $\mathtt{other}$ writes a command $\mathtt{emacs}$ $\mathtt{readme}$ to a shell.

Logical Hidden Markov Models (LOHMMs): The sequences generated by LOHMMs are sequences of ground atoms rather than flat symbols. Within LOHMMs, the flat symbols employed in traditional HMMs are replaced by logical atoms such as $\mathtt{emacs}(\mathtt{F}, \mathtt{tex})$. Each atom $\mathtt{emacs}(\mathtt{F}, \mathtt{tex})$ there represents the set of ground atoms $G_\Sigma(\mathtt{emacs}(\mathtt{F}, \mathtt{tex}))$, e.g. $\mathtt{emacs}(\mathtt{readme}, \mathtt{tex}) \in G_\Sigma(\mathtt{emacs}(\mathtt{F}, \mathtt{tex}))$.

Additionally, we assume that the alphabet is typed which in our case means that there is a function mapping every predicate $\mathtt{r/m}$ and number $1 \leq i \leq m$ to the set of ground terms allowed as the i-th argument of predicate $\mathtt{r/m}$. This set is called the domain of the i-th argument of predicate $\mathtt{r/m}$.

Figure 1 shows a LOHMM graphically. The states, observations, and transitions of LOHMMs are **abstract** in the sense that every abstract state or observation $\mathtt{A}$ represents all possible concrete states in $G_\Sigma(\mathtt{A})$. In Figure 1 *solid edges* encode **abstract transitions**. Let $\mathtt{H}$ and $\mathtt{B}$ be logical atoms representing abstract states, let $\mathtt{O}$ be a logical atom representing an abstract output symbol. An abstract transition from state $\mathtt{B}$ with probability p to state $\mathtt{H}$ and omitting $\mathtt{O}$ is denoted by $p : \mathtt{H} \xleftarrow{\mathtt{O}} \mathtt{B}$. If $\mathtt{H}$, $\mathtt{B}$, and $\mathtt{O}$ are all ground, there is no difference to 'normal' transitions. Otherwise, if $\mathtt{H}$, $\mathtt{B}$, and $\mathtt{O}$ have no variables in common, the only difference to 'normal' transitions is that for each abstract state (resp. observation) we have to sample which concrete state (resp. observation) we are in. Otherwise, we have to remember the variable bindings. More formally, let $\mathtt{B}\theta_\mathtt{B} \in G_\Sigma(\mathtt{B})$, $\mathtt{H}\theta_\mathtt{B}\theta_\mathtt{H} \in G_\Sigma(\mathtt{H}\theta_\mathtt{B})$ $\mathtt{O}\theta_\mathtt{B}\theta_\mathtt{H}\theta_\mathtt{O} \in G_\Sigma(\mathtt{O}\theta_\mathtt{B}\theta_\mathtt{H})$, and let μ be a **selection distribution**. Then with probability $p \cdot \mu(\mathtt{H}\theta_\mathtt{B}\theta_\mathtt{H} \mid \mathtt{H}\theta_\mathtt{B}) \cdot \mu(\mathtt{O}\theta_\mathtt{B}\theta_\mathtt{H}\theta_\mathtt{O} \mid \mathtt{O}\theta_\mathtt{B}\theta_\mathtt{H})$, the model makes a transition from state $\mathtt{B}\theta_\mathtt{B}$ to $\mathtt{H}\theta_\mathtt{B}\theta_\mathtt{H}$ and emits symbol $\mathtt{O}\theta_\mathtt{B}\theta_\mathtt{H}\theta_\mathtt{O}$.

A selection distribution specifies for each abstract state (respectively observation) $\mathtt{A}$ over the alphabet Σ a distribution $\mu(\cdot \mid \mathtt{A})$ over $G_\Sigma(\mathtt{A})$. Consider, for example, the abstract transition $0.5 : \mathtt{s}(\mathtt{f}(\mathtt{Z})) \xleftarrow{\mathtt{o}(\mathtt{X},\mathtt{Y},\mathtt{Z})} \mathtt{s}(\mathtt{X})$. Suppose, $\mathtt{B}\theta_\mathtt{B} = \mathtt{s}(1)$, $\mu(\mathtt{s}(\mathtt{f}(3)) \mid \mathtt{s}(\mathtt{f}(\mathtt{Z}))) = 0.2$ and $\mu(\mathtt{o}(1,2,3) \mid \mathtt{o}(1,\mathtt{Y},3)) = 0.05$. Then, from state $\mathtt{s(1)}$ with probability $0.5 \times 0.2 \times 0.05 = 0.005$ the output symbol is $\mathtt{o(1,2,3)}$ and the next state is $\mathtt{s(f(3))}$. To reduce the model complexity, we employ a naïve Bayes approach in which – at the expense of a lower expressivity – functors are neglected and variables are treated independently. More precisely, for each domain D_i there is a probability distribution P_{D_i}. Let $\mathrm{vars}(\mathtt{A}) = \{\mathtt{V}_1, \ldots, \mathtt{V}_l\}$ be the variables occurring in $\mathtt{A}$, and let $\theta = \{\mathtt{t}_1/\mathtt{V}_1, \ldots \mathtt{t}_1/\mathtt{V}_1\}$ be a substitution grounding $\mathtt{A}$. Each $\mathtt{V}_j$ is then considered a random variable over the domain of the first argument of $\mathtt{r/m}$ it appears in, denoted by $D_{\mathtt{V}_j}$. Then, $\mu(\mathtt{A}\theta \mid \mathtt{A}) = \prod_{j=1}^{l} P_{D_{\mathtt{V}_j}}(\mathtt{V}_j = \mathtt{t}_j)$. For instance, $\mu(\mathtt{s}(\mathtt{f}(3)) \mid \mathtt{s}(\mathtt{f}(\mathtt{Z})))$ equals $P_1^{\mathtt{s}/1}(3)$.

Indeed, multiple abstract transitions can match a given ground state. Consider the abstract states $\mathtt{B}_1 = \mathtt{emacs}(\mathtt{File}, \mathtt{tex})$ and $\mathtt{B}_2 = \mathtt{emacs}(\mathtt{File}, \mathtt{User})$ in Fig. 1 **(a)**. The abstract state $\mathtt{B}_1$ is more specific than $\mathtt{B}_2$ because there exists a substitution $\theta = \{\mathtt{User}/\mathtt{tex}\}$ such that $\mathtt{B}_2\theta = \mathtt{B}_1$, i.e., $\mathtt{B}_2$ subsumes $\mathtt{B}_1$. Therefore $G_\Sigma(\mathtt{B}_1) \subseteq G_\Sigma(\mathtt{B}_2)$ and the first transition can be regarded as more informative than the second one. It should therefore be preferred over the second one when starting

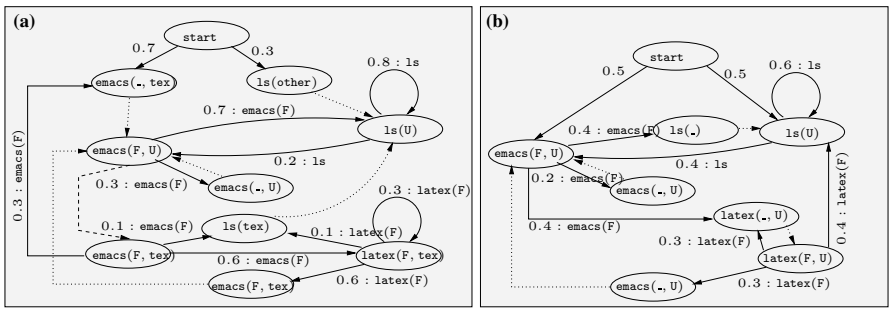

Figure 1: Logical hidden Markov models. The vertices represent abstract (hidden) states. Solid edges encode abstract transitions. Dotted edges indicate that two abstract states behave in exactly the same way. Dashed edge denote the *more general than* relation. The LOHMMs are described in the text.

e.g. from $\texttt{emacs}(\texttt{hmm1}, \texttt{tex})$. We will also say that the *transitions* of the first abstract state are *more specific* than the second ones; encoded by *dashed edges*. These considerations lead to the **conflict resolution strategy** [1] of only considering the maximally specific transitions that apply to a state in order to determine the successor states. This implements a kind of exception handling or default reasoning and is akin to Katz's *back-off* n-gram models [10]. In back-off n-gram models, the most detailed model that is deemed to provide sufficiently reliable information about the current context is used. That is, if one encounters an n-gram that is not sufficiently reliable, then back-off to use an $(n-1)$-gram; if that is not reliable either then back-off to level $n-2$, etc.

Finally, *dotted edges* denote that two abstract states behave in exactly the same way. If we follow a transition to an abstract state with an outgoing dotted edge, we will automatically follow that edge making appropriate unifications.

Definition 1 *A* logical hidden Markov model *(LOHMM) is a tuple* $M = (\Sigma, \mu, \Delta)$ *where* Σ *is a logical alphabet,* μ *a selection probability over* Σ *and* Δ *is a set of abstract transitions. Let* $\mathbf{B}$ *be the set of all atoms that occur as the body part of transitions in* Δ*. We require*

$$\forall \mathtt{B} \in \mathbf{B} : \sum\nolimits_{p:\mathtt{H} \xleftarrow{0} \mathtt{B} \in \Delta} p = 1. \qquad (1)$$

In [11] it is proven that LOHMMs specify a unique probability measure over hb_Σ. Here, we would like to exemplify that LOHMMs are generative models. Consider the model in Fig. 1**(a)**. Starting from `start`, it chooses an initial abstract state, say $\texttt{emacs}(_, \texttt{tex})$ with probability 0.7. Here, `_` denotes an anonymous variable which is read and treated as distinct, new variables each time it is encountered. Forced to follow the dotted edge, it enters the abstract state $\texttt{emacs}(\texttt{F}, \texttt{U})$. In each abstract state, the model samples values for all variables that are not instantiated yet according to the *selection distribution* μ. Since the value of `U` was already instantiated in the previous abstract state $\texttt{emacs}(\texttt{F}, \texttt{tex})$, the model has only to sample a value for `F`, say `f1`, using μ. Now, it selects a transition, say, to $\texttt{latex}(\texttt{F}, \texttt{tex})$ with probability 0.6. Since `F` is shared among the head and the body, the state $\texttt{latex}(\texttt{f1}, \texttt{tex})$ is selected with probability 1.0. The observation $\texttt{emacs}(\texttt{f1})$ is emitted from $\texttt{emacs}(\texttt{F})$ with probability 1.0 using μ. Now, the model goes over to, say $\texttt{ls}(\texttt{tex})$, emitting $\texttt{latex}(\texttt{f1})$ which in turn was sampled from $\texttt{latex}(\texttt{F})$. The dotted edge brings us to $\texttt{ls}(\texttt{U})$ and automatically unifies `U` with `tex`. Emitting `ls`, we return to $\texttt{emacs}(\texttt{F}, \texttt{tex})$ where `F` now denotes a new filename.

3 The Learning Setting

For traditional HMMs, the learning problem basically collapses to parameter estimation (i.e., estimating the transition probabilities) because HMMs can be considered to be fully connected. For LOHMMs, however, we have to account for different abstraction levels. The model selection problem can formally be defined as:

Given a set $\mathbf{O} = \{O_1, \ldots, O_m\}$ of data cases independently sampled from the same distribution, a set $\mathcal{M}$ of LOHMMs, and a scoring function $score_{\mathbf{O}} : \mathcal{M} \mapsto \mathbb{R}$, **find** a hypothesis $M^* \in \mathcal{M}$ that maximizes $score_{\mathbf{O}}$.

Each *data case* $O_i \in \mathbf{O}$ is a sequence $O_i = \mathtt{o}_{\mathtt{i},1}\mathtt{o}_{\mathtt{i},2}\ldots\mathtt{o}_{\mathtt{i},\mathtt{T}}$ of ground atoms and describes the observations evolving over time. For instance in the user modeling domain a data case could be $\texttt{emacs}(\texttt{lohmms}), \texttt{ls}, \texttt{emacs}(\texttt{lohmms})$. The corresponding evolution of the system's state over time $H_i = \mathtt{h}_{\mathtt{i},0}\mathtt{h}_{\mathtt{i},1}\ldots\mathtt{h}_{\mathtt{i},\mathtt{T}_\mathtt{i}+1}$ is hidden, i.e. , not specified in O_i. For instance, we do not know whether $\texttt{emacs}(\texttt{lohmms})$ has been generated by $\texttt{emacs}(\texttt{lohmms}, \texttt{prog})$ or $\texttt{emacs}(\texttt{lohmms}, \texttt{tex})$.

The *hypothesis space* $\mathcal{M}$ consists of all candidate LOHMMs to be considered during search. We assume Σ

[1] Another conflict resolution strategy would be *smoothing*, i.e. , considering all matching abstract states. We chose not to use smoothing to keep the LOHMM locally interpretable, i.e. to have a single abstract body for each ground state.

to be given. Thus, the possible constants which can be selected by μ are apriori known. Each $M \in \mathcal{M}$ is parameterized by a vector $\boldsymbol{\lambda}_M$. Each (legal) choice of $\boldsymbol{\lambda}_M$ defines a probability distribution $P(\cdot \mid M, \boldsymbol{\lambda}_M)$ over $\mathrm{hb}(\Sigma)$. For the sake of simplicity, we will denote the underlying logic program (i.e., the set of abstract transitions without associated probability values) by M and abbreviate $\boldsymbol{\lambda}_M$ by $\boldsymbol{\lambda}$ as long as no ambiguities will arise. Furthermore, a syntactic bias on the transitions to be induced is a parameter of our framework, as usual in ILP [18]. For instance in the experiments, we only consider transitions which obey the type constraints induced by the predicates.

As score, we employ $\mathit{score}_{\mathbf{O}}(M, \boldsymbol{\lambda}) = \log P(\mathbf{O} \mid M, \boldsymbol{\lambda}) - \mathit{Pen}(M, \boldsymbol{\lambda}, \mathbf{O})$. Here, $\log P(\mathbf{O} \mid M, \boldsymbol{\lambda})$ is the *log-likelihood* of the current of model $(M, \boldsymbol{\lambda})$. It holds that the higher the log-likelihood, the closer $(M, \boldsymbol{\lambda})$ models the probability distribution induced by the data. The second term, $\mathit{Pen}(M, \boldsymbol{\lambda}, \mathbf{O})$, is a penalty function that biases the scoring function to prefer simpler models. Motivated by the *minimum description length* score for Bayesian networks, we use the simple penalty $\mathit{Pen}(M, \boldsymbol{\lambda}, \mathbf{O}) = |\Delta| \log(m)/2$. It is independent of the model parameters and therefore it can be neglected when estimating parameters. We assume that each M covers all possible observation sequences (over the given language Σ). This guarantees that all new data cases will get a positive likelihood.

4 A Naïve Learning Algorithm

A simple way of selecting a model structure is the following greedy approach:

1: Let $\boldsymbol{\lambda}^0 = \mathrm{argmax}_{\boldsymbol{\lambda}}\, \mathit{score}_{\mathbf{O}}(M^0, \boldsymbol{\lambda})$
2: **Loop** for $k = 0, 1, 2, \ldots$
3: **Find** model $M^{k+1} \in \rho(M^k)$ that maximizes $\max_{\boldsymbol{\lambda}} \mathit{score}_{\mathbf{O}}(M^{k+1}, \boldsymbol{\lambda})$
4: Let $\boldsymbol{\lambda}^{k+1} = \mathrm{argmax}_{\boldsymbol{\lambda}}\, \mathit{score}_{\mathbf{O}}(M^{k+1}, \boldsymbol{\lambda})$
5: **Until** convergence, i.e., no improvement in score

It takes as input an initial model M^0 and the data $\mathbf{O}$. At each stage k we choose a model structure and parameters among the current best model M^k and its neighbors $\rho(M^k)$ (see below) that have the highest score. It stops, when there is no improvement in score. In practice, we initialize the parameters of each model on lines 1 and 3 randomly.

We will now show how to traverse the hypotheses space and how to estimate parameters for a hypothesis in order to score it. That is, we will make line 3 more concrete.

Traversing the Hypotheses Space: An obvious candidate for the initial hypothesis M^0 (which we also used in our experiments) is the fully connected LOHMM built over all maximally general atoms over Σ, i.e., expressions of the form $\mathtt{r}(\mathtt{X}_1, ..., \mathtt{X}_m)$, where the $\mathtt{X}_i$ are different variables.

Now, to traverse the hypothesis space $\mathcal{M}$, we have to compute all neighbors of the currently best hypothesis M^k. To do so, we employ refinement operators traditionally used in ILP. More precisely, for the language bias considered and the experiments conducted in the present paper, we used the refinement operator $\rho : \mathcal{M} \mapsto 2^{\mathcal{M}}$ which selects a single clause $\mathrm{cl} \equiv p : \mathtt{H} \xleftarrow{0} \mathtt{B} \in \mathcal{M}$ and adds a minimal specialization $\mathrm{cl}' \equiv p : \mathtt{H}' \xleftarrow{0'} \mathtt{B}'$ of cl to $\mathcal{M}$ (w.r.t. to θ-subsumption). Specializing a single abstract transition means instantiating or unifying variables, i.e., $\mathrm{cl}' \equiv \mathrm{cl}\,\theta$ for some substitution θ. When adding cl' to M^k, we have to ensure that (1) the same observation and hidden state sequences are still covered and (2) the list of bodies $\mathbf{B}'$ after applying $\rho(M)$ should remain well-founded, that is, for each ground state, there is a unique maximally specific body in $\mathbf{B}'$. Both conditions together guarantee that the most specific body corresponding to a state always exists and is unique. Condition (1) can only be violated if $\mathtt{B}' \notin \mathbf{B}$. In this case, we add transitions with $\mathtt{B}'$ and maximally general heads and observations. Condition (2) is established analogously. We complete the keep the list of bodies well-founded by adding new bodies (and therefore abstract transitions) in a similar way as described above.

Consider refining the LOHMM in Fig. 1 **(b)**. When adding $\mathtt{ls(U)} \xleftarrow{\mathtt{latex(lohmm)}} \mathtt{latex(lohmm, U)}$, hence introducing the more specific abstract state $\mathtt{latex(lohmm, U)}$, further variants of the same abstract transition but with different heads have to be added. Otherwise condition (1) would be violated as the resulting LOHMM does not cover the same sequences as the original one; the state $\mathtt{latex(lohmm, U)}$ can only be left via $\mathtt{ls(U)}$ and not e.g. via $\mathtt{emacs(_, U)}$. On the other hand, we have to be careful when subsequently adding abstract transitions for the body $\mathtt{latex(F, tex)}$. The problem is that we do not know which abstract body to select in state $\mathtt{latex(lohmm, tex)}$. To fulfill condition (2), you need to add abstract transitions for an additional, third abstract state $\mathtt{latex(lohmm, tex)}$, too.

Parameter Estimation: In the presence of hidden variables maximum log-likelihood (ML) parameter estimation is a numerical optimization problem, and all known algorithms involve nonlinear, iterative optimization and multiple calls to an inference algorithm. The most common approach for HMMs is the Baum-Welch algorithm, an instance of the EM algorithm [5]. In each iteration $l + 1$ it performs two steps:

(E-step) *Compute the expectation of the log-likelihood given the old model* $(M^k, \boldsymbol{\lambda}^{k,l})$ *and the observed data* $\mathbf{O}$*, i.e.,* $Q(M^k, \boldsymbol{\lambda} \mid M^k, \boldsymbol{\lambda}^{k,l}) = E\left[\log P(\mathbf{O}, \mathbf{H} \mid M^k, \boldsymbol{\lambda}) \mid M^k, \boldsymbol{\lambda}^{k,l}\right]$.

Here, $\mathbf{O}, \mathbf{H}$ denotes the completion of $\mathbf{O}$ where the evolution $\mathbf{H}$ of the system's state over time is made explicit. The current model $(M^k, \boldsymbol{\lambda}^{k,l})$ and the observed data $\mathbf{O}$ give

us the conditional distribution governing $\mathbf{H}$, and $E[\cdot|\cdot]$ denotes the expectation over it. The function Q is called the *expected score*.

(M-step) *Maximize the expected score* $Q(M^k, \boldsymbol{\lambda} \mid M^k, \boldsymbol{\lambda}^{k,l})$ *w.r.t.* $\boldsymbol{\lambda}$*, i.e.,* $\boldsymbol{\lambda}^{k,l+1} = \mathit{argmax}_{\boldsymbol{\lambda}}\, Q(M^k, \boldsymbol{\lambda} | M^k, \boldsymbol{\lambda}^{k,l})$.

The naïve greedy algorithm can easily be instantiated using the EM. The problem, however, is its huge computational costs. To evaluate a single neighbor, the EM has to run for a reasonable number of iterations in order to get reliable ML estimates of $\boldsymbol{\lambda}^{k'}$. Each EM iteration requires a full LOHMM inference on all data cases. In total, the running time per neighbor evaluation is at least $\mathcal{O}(\#\textit{EM iterations} \cdot \textit{size of data})$.

5 SAGEM: Structural Generalized EM

To reduce the computational costs, SAGEM (German for *'say EM'*) adapts Friedman's *structural EM* (SEM) [6]. That is, we take our current model $(M^k, \boldsymbol{\lambda}^k)$ and run the EM algorithm for a while to get reasonably completed data. We then fix the completed data cases and use them to compute the ML parameters $\boldsymbol{\lambda}^{k'}$ of each neighbor $M^{k'}$. We choose the neighbor with the best improvement of the score as $(M^{k+1}, \boldsymbol{\lambda}^{k+1})$ and iterate. More formally, we have

1: Initialize $\boldsymbol{\lambda}^{0,0}$ randomly
2: **Loop** for $k = 0, 1, 2, \ldots$
3: **Loop** for $l = 0, 1, 2, \ldots$
4: Let $\boldsymbol{\lambda}^{k,l+1} = \text{argmax}_{\boldsymbol{\lambda}}\, Q(M^k, \boldsymbol{\lambda} \mid M^k, \boldsymbol{\lambda}^{k,l})$
5: **Until** convergence **or** $l = l_{\max}$
6: **Find** model $M^{k+1} \in \rho(M^k)$ that maximizes $\max_{\boldsymbol{\lambda}} Q(M^{k+1}, \boldsymbol{\lambda} \mid M^k, \boldsymbol{\lambda}^{k,l})$
7: Let $\boldsymbol{\lambda}^{k+1,0} = \text{argmax}_{\boldsymbol{\lambda}}\, Q(M^{k+1}, \boldsymbol{\lambda} \mid M^k, \boldsymbol{\lambda}^{k,l})$
8: **Until** convergence

The hypotheses space is traversed as described in Section 4, and again we stop if there is no improvement in score. The following theorem shows that even when the structure changes in between, improving the expected score Q always improves the log-likelihood as well.

Theorem 1 *If* $Q(M, \boldsymbol{\lambda} \mid M^k, \boldsymbol{\lambda}^{k,l}) > Q(M^k, \boldsymbol{\lambda}^{k,l} \mid M^k, \boldsymbol{\lambda}^{k,l})$ *holds, then* $\log P(\mathbf{O} \mid M, \boldsymbol{\lambda}) > \log P(\mathbf{O} \mid M^k, \boldsymbol{\lambda}^{k,l})$ *holds.*

The proof is a simple extension of the argumentation by [16, Section 3.2 ff.]. To apply the algorithm to selecting LOHMMs, we will now show how to choose the best neighbour [2] in line 6.

[2] In the following, we will omit some derivation steps due to space restriction. They can be found in [13]. Furthermore, for the sake of simplicity, we will not explicitly check that a transition is *maximally specific* for ground states.

Let $c(\mathtt{b}, \mathtt{h}, \mathtt{o})$ denote the number of times the systems proceeds from ground state $\mathtt{b}$ to ground state $\mathtt{h}$ emitting ground observation $\mathtt{o}$. The expected score in line 6 simplifies to

$$Q(M, \boldsymbol{\lambda} | M^k, \boldsymbol{\lambda}^{k,l}) \tag{2}$$
$$= \sum\nolimits_{\mathtt{b},\mathtt{h},\mathtt{o}} \underbrace{E\left[c(\mathtt{b},\mathtt{h},\mathtt{o}) \middle| M^k, \boldsymbol{\lambda}^{k,l}\right]}_{=: ec(\mathtt{b},\mathtt{h},\mathtt{o})} \cdot \log P(\mathtt{h}, \mathtt{o} | \mathtt{b}, M, \boldsymbol{\lambda}) \ .$$

The term $ec(\mathtt{b}, \mathtt{h}, \mathtt{o})$ in (2) denotes the expected counts of making a transition from ground state $\mathtt{b}$ to ground state $\mathtt{h}$ emitting ground observation $\mathtt{o}$. The expectation is taken according to $(M^k, \boldsymbol{\lambda}^{k,l})$.

An analytical solution, however, of the M-step in line 7 seems to be difficult. In HMMs, the updated transition probabilities are simply directly proportional to the expected number of times they are used. In LOHMMs, however, there is an ambiguity: multiple abstract transitions (with the same body), can match the same ground transition $(\mathtt{b}, \mathtt{h}, \mathtt{o})$. Using *ec* as sufficient statistics makes the M step nontrivial. The solution is to improve (2) instead of maximizing it. Such an approach is called *generalized EM* [16]. To do so, we follow a gradient-based optimization technique. We iteratively compute the *gradient* $\nabla_{\boldsymbol{\lambda}}$ of (2) w.r.t. the parameters of a LOHMM, and, then, take a step in the direction of the gradient to the point $\boldsymbol{\lambda} + \delta \nabla_{\boldsymbol{\lambda}}$ where δ is the step-size.

For LOHMMs, the gradient w.r.t. (2) consists of partial derivatives w.r.t. abstract transition probabilities and to selection probabilities. Assume that λ is the transition probability associated with some abstract transition cl. Now, the partial derivative of (2) w.r.t. some parameter λ is

$$\frac{\partial Q(M, \boldsymbol{\lambda} \mid M^k, \boldsymbol{\lambda}^{k,l})}{\partial \lambda}$$
$$= \sum\nolimits_{\mathtt{b},\mathtt{h},\mathtt{o}} ec(\mathtt{b},\mathtt{h},\mathtt{o}) \cdot \frac{\partial \log P(\mathtt{h}, \mathtt{o} \mid \mathtt{b}, M, \boldsymbol{\lambda})}{\partial \lambda}$$
$$= \sum\nolimits_{\mathtt{b},\mathtt{h},\mathtt{o}} \frac{ec(\mathtt{b},\mathtt{h},\mathtt{o})}{P(\mathtt{h}, \mathtt{o} \mid \mathtt{b}, M, \boldsymbol{\lambda})} \cdot \frac{\partial P(\mathtt{h}, \mathtt{o} \mid \mathtt{b}, M, \boldsymbol{\lambda})}{\partial \lambda} \tag{3}$$

The partial derivative of $P(\mathtt{h}, \mathtt{o} \mid \mathtt{b}, M, \boldsymbol{\lambda})$ w.r.t. λ can be computed as

$$\frac{P(\mathtt{h}, \mathtt{o} \mid \mathtt{b}, M, \boldsymbol{\lambda})}{\partial \lambda}$$
$$= \mu(\mathtt{h} \mid \mathrm{head}(\mathrm{cl})\theta_{\mathtt{H}}, M) \cdot \mu(\mathtt{o} \mid \mathrm{obs}(\mathrm{cl})\theta_{\mathtt{H}}\theta_{\mathtt{0}}, M) \tag{4}$$

Substituting (4) back into (3) yields

$$\frac{\partial Q(M, \boldsymbol{\lambda} \mid M^k, \boldsymbol{\lambda}^{k,l})}{\partial \lambda}$$
$$= \sum_{\mathtt{b},\mathtt{h},\mathtt{o}} \left(\frac{ec(\mathtt{b},\mathtt{h},\mathtt{o})}{P(\mathtt{h}, \mathtt{o} \mid \mathtt{b}, M, \boldsymbol{\lambda})} \cdot \mu(\mathtt{h} \mid \mathrm{head}(\mathrm{cl})\theta_{\mathtt{H}}, M) \right.$$
$$\left. \cdot \mu(\mathtt{o} \mid \mathrm{obs}(\mathrm{cl})\theta_{\mathtt{H}}\theta_{\mathtt{0}}, M) \right) \tag{5}$$

The selection probability follows a naïve Bayes approach. Therefore, one can show in a similar way as for transition probabilities that

$$\frac{\partial Q(M, \boldsymbol{\lambda} \mid M^k, \boldsymbol{\lambda}^{k,l})}{\partial \lambda} = \sum_{\mathtt{b},\mathtt{h},\mathtt{o}} \Big(\frac{ec(\mathtt{b},\mathtt{h},\mathtt{o})}{P(\mathtt{h},\mathtt{o} \mid \mathtt{b}, M, \boldsymbol{\lambda})} \cdot \sum_{\mathrm{cl}} c(\lambda, \mathrm{cl}, \mathtt{b}, \mathtt{h}, \mathtt{o}) \cdot \cdot P(\mathrm{cl} \mid M, \boldsymbol{\lambda}) \cdot \mu(\mathtt{h} \mid \mathrm{head}(\mathrm{cl})\theta_{\mathtt{H}}, M) \cdot \cdot \mu(\mathtt{o} \mid \mathrm{obs}(\mathrm{cl})\theta_{\mathtt{H}}\theta_{\mathtt{O}}, M) \Big) \quad (6)$$

where $c(\lambda, \mathrm{cl}, \mathtt{b}, \mathtt{h}, \mathtt{o})$ is the number of times that the domain element associated with λ is selected to ground cl w.r.t. $\mathtt{h}$ and $\mathtt{o}$.

In the problem at hand, the described method has to be modified to take into account that $\lambda \in [0, 1]$ and that corresponding λ's sum to 1.0. A general solution, which we used in our experiments, is to reparameterize the problem so that the new parameters automatically respect the constraints no matter what their values are. To do so, we define the parameters $\boldsymbol{\beta}$ where $\beta_{ij} \in \mathbb{R}$ such that $\lambda_{ij} = \exp(\beta_{ij}) / (\sum_l \exp(\beta_{il}))$. This enforces the constraints given above, and a local maximum w.r.t. $\boldsymbol{\beta}$ is also a local maximum w.r.t. $\boldsymbol{\lambda}$, and vice versa. The gradient w.r.t the β_{ij}'s can be found by computing the gradient w.r.t. the λ_{ij}'s and then deriving the gradient w.r.t. $\boldsymbol{\beta}$ using the chain rule.

Discussion on SAGEM: What do we gain from SAGEM over the naïve approach? The expected ground counts $ec(\mathtt{b}, \mathtt{h}, \mathtt{o})$ are used as the sufficient statistics to evaluate all the neighbors. Evaluating neighbors is thus now independent of the number and length of the data cases—a feature which is important for scaling up. More precisely, the running time per neighbor evaluation is basically $\mathcal{O}(\#\textit{Gradient iterations} \cdot \#\textit{Ground transitions})$ because SAGEM's gradient approach does not perform LOHMM inferences.

The greedy approach is not always enough. For instance, if two hidden states are equivalent, to make them effectively differ from each other, one needs to make them differ both in visiting probabilities of the state and in behavior in the state, possibly requiring two steps for any positive effect. Fixing the expected counts in SAGEM worsens the problem, since changes in visiting probabilities of states do not show up before a LOHMM inference is made. To overcome this, different search strategies, such as beam search, can be used: Instead of a current hypothesis, a fixed-size set of current hypotheses is considered, and their common neighborhood is searched for the next set.

To summarize, SAGEM explores different abstraction levels due to ILP refinement operators, and, due to a GEM approach, it reduces the neighborhood evaluation problem to one that is solvable more efficiently.

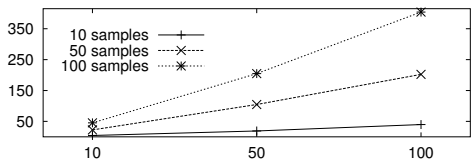

Figure 2: Speed-up (y axis), i.e., the ratio of time per EM iteration (in sec.) and time per SAGEM's gradient approach to evaluate neighbors. The speed-up is shown for different numbers of sequence lengths (x axis) and for different numbers of data cases (curves).

6 Experimental Evaluation

Our intentions here are to investigate whether SAGEM can be applied to real world domains. More precisely, we will investigate whether SAGEM

H1 speeds-up neighbor evaluation considerably (compared to the naïve learning algorithm);
H2 finds a comprehensible model;
H3 works in the presence of transition ambiguity;
H4 can be applied to real-world domains and is competitive with standard machine learning algorithms such as nearest-neighbor and decision-tree learners.

To this aim, we implemented the SAGEM using the Prolog system YAP-4.4.4. The experiments were run on a Pentium-III-2.3 GHz machine. For the improvement of expected score, we adapted the scaled conjugate gradient as implemented in Bishop and Nabney's Netlab library (`http://www.ncrg.aston.ac.uk/netlab/`) with a maximum number of 10 iterations and 5 random restarts.

Experiments with Synthetic Data: We sampled independently $10, 50, 100$ sequences of length $10, 50, 100$ (100 to 10000 ground atoms in total) from the LOHMM shown in Fig. 1**(a)** and computed their ground counts w.r.t. the samples. We measured the averaged running time in seconds per iteration for both, the naïve algorithm and SAGEM's gradient approach to evaluate neighbors when applied to the LOHMM shown in Fig. 1**(b)**. The times were measured using YAP's built-in `statistics/2`. The results are summarized in Fig. 2 showing the ratio of running times of naïve over SAGEM's gradient approach. In some cases the speed-up was more than 400. EM's lowest running time was 0.075 seconds (for 10 sequences of length 10). In contrast, SAGEM was constantly below 0.017 seconds. This suggests that **H1** holds.

We sampled 2000 sequences of length 15 (30000 ground atoms) from the LOHMM in Fig. 1**(a)**. There were 4 filenames, 2 users types. The initial hypothesis was the LOHMM in Fig. 1**(b)** with randomly initialized parameters. We run SAGEM on the sampled data. [3] Averaged

[3] The naïve algorithm was no longer used for comparison due

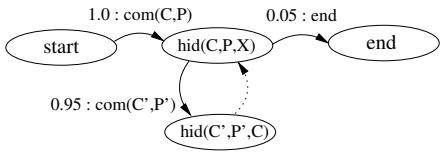

Figure 3: The initial hypothesis for the experiments with real-world data is a minimal structure, implying learning from scratch. C stands for command and P for parameters. The hidden state hid contains the new command, parameters and the latest old command.

over 5 runs, estimating the parameters for the initial hypothesis achieved a score of -47203. In contrast, the score of SAGEM's selected model was -26974 which was even slightly above the score of the original LOHMM (-30521). This suggests that **H3** holds. Moreover, in all runs, SAGEM included e.g. $\texttt{latex}(\texttt{A},\texttt{B}) \xleftarrow{0.61:\texttt{emacs(A)}} \texttt{emacs}(\texttt{A},\texttt{B})$ and $\texttt{emacs}(\texttt{A},\texttt{B}) \xleftarrow{0.48:\texttt{emacs(A)}} \texttt{latex}(\texttt{A},\texttt{B})$ which were not present in the initial model. This suggests that **H2** holds.

Experiments with Real-World Data: Finally, we applied SAGEM to the data set collected by Greenberg [8]. The data consists of 168 users of four groups: computer scientists, non-programmers, novices and others. About 300000 commands have been logged in on average 110 sessions per user. We present here results for two classes: novice-1(NV) consisting of 2512 ground atoms and non-programmers-4 (NP) consisting of 5183 ground atoms. We randomly selected 35 training sessions (about 1500 commands) for each class. On this data, we let SAGEM select a model for each class independently, starting from the initial hypothesis described in Fig. 3. To evaluate, we computed the plug-in estimates of each model for the remaining sessions corrected by the class priors. Averaged over five runs, the precision (0.94 ± 0.06 NV, 0.91 ± 0.02 NP) and recall values (0.67 ± 0.03 NV, 0.89 ± 0.05 NP) were balanced and the overall predictive accuracy was 0.92 ± 0.01. Jacobs and Blockeel [9] report that a kNN approach achieved a precision of 0.91 and J48 (WEKA's implementation of Quinlan's C4.5 decision tree learner) of 0.86 averaged over ten runs on 50 randomly sampled training examples. This suggests that **H4** holds. The used kNN and J48 methods, however, do not yield generative models and lack comprehensibility. SAGEM's selected models encoded e.g. "*non-programmers are very likely to type in* `cd`*.. after performing* `ls` *in some directory*". This pattern was not present in the NV model. This suggests that **H2** holds.

7 Related Work

Statistical relational learning (SRL) can be viewed as combining ILP principles (such as refinement operators) with statistical learning, see [3] for an overview and references. Most attention, however, has been devoted to developing highly expressive formalisms. LOHMMs can be seen as an attempt towards *downgrading* such highly expressive frameworks. They retain most of the essential logical features but are easier to understand, adapt and learn. For the same reasons, simple statistical techniques (such as logistic regression or naïve Bayes) have been combined with ILP refinement oprators for traversing the search space, see e.g. [20, 14]. They, however, do not select dynamic models.

LOHMMs are related to Anderson *et al.*'s *relational Markov models* (RMMs) [1]. Here, states can be of different types, with each type described by a different set of variables. The domain of each variable is hierarchically structured. The main differences are that neither variable bindings, unification nor hidden states are supported. RMMs do not select the most-specific transition to resolve conflicting transitions. Instead, they interpolate between conflicting ones. This is an interesting option for LOHMMs because it makes parameter estimation more robust. On the other hand, it also seems to make it more difficult to adhere one of our design principles: locally interpretable transitions. Structure learning has been addressed based on *probability estimation trees*.

Logical sequences can be converted into binary trees by putting each instance of a relation symbol into a node. The left subtree represents the first argument and the right subtree represents the next atom in the list (of observations or arguments). Methods for learning tree languages [2] can thus be used for learning probabilistic models for logical sequences, too. The main differences, though, is that variable bindings are not supported.

LOHMMs are related to several extensions of HMMs such as factorial HMMs [7]. Here, state variables are decomposed into smaller units. The key difference to LOHMMs is that these approaches do not employ logical concepts.

Finally, SAGEM is related to more advanced HMM model selection methods. *Model merging* [22] starts with the most specific model consistent with the training data and generalizes by successively merging states. Abstract transitions, however, aim at good generalization, and the most general clauses can be considered to be the most informative ones. Therefore, *successive state splitting* [19] refines hidden states by splitting them into new states. In both cases, the authors are not aware of adaptions of Friedman's SEM.

8 Conclusions

A novel model selection method for logical hidden Markov models called SAGEM has been introduced. SAGEM combines *generalized EM*, which optimizes parameters, with structure search for model selection using ILP refinement operators. Experiments show SAGEM's effectiveness.

Future work should address other scores; other refinement

to unreasonable running times.

operators e.g. handling functors, deleting transitions, and generalizing hypotheses; logical pruning criteria for hypotheses; and efficient storing of ground counts. Moreover, the authors hope that the presented work will inspire further research at the intersection of ILP and HMMs.

Acknowledgments: The authors thank the three anonymous reviewers for helpful comments, Luc De Raedt for discussions on the topic, Nico Jacobs for discussions on user modeling, and Saul Greenberg for providing the data. The research was partially supported by the EU IST programme, contract no. FP6-508861, *APrIL II*, and by the Finnish Centre of Excellence Programme (2000-2005) under the project *New Information Processing Principles*. Tapani Raiko was partially supported by a Marie Curie fellowship at DAISY, HPMT-CT-2001-00251.

References

[1] C. R. Anderson, P. Domingos, and D. S. Weld. Relational Markov Models and their Application to Adaptive Web Navigation. In *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining (KDD-02)*, 2002.

[2] R.C. Carrasco, J. Oncina, and J. Calera-Rubio. Stochastic inference of regular tree languages. *Machine Learning*, 44(1/2):185–197, 2001.

[3] L. De Raedt and K. Kersting. Probabilistic Logic Learning. *ACM-SIGKDD Explorations*, 5(1):31–48, 2003.

[4] L. De Raedt and K. Kersting. Probabilistic Inductive Logic Programming. In *Proceedings of the 15th International Conference on Algorithmic Learning Theory (ALT-2004)*, volume 3244 of *LNCS*, pages 19–36, Padova, Italy, October 2–5 2004. Springer.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc.*, B 39:1–39, 1977.

[6] N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML-1997)*, 1997.

[7] Z. Ghahramani and M. Jordan. Factorial hidden Markov models. *Machine Learning*, 29:245–273, 1997.

[8] S. Greenberg. Using Unix: collected traces of 168 users. Technical report, Dept. of Computer Science, University of Calgary, Alberta, 1988.

[9] N. Jacobs and H. Blockeel. User modeling with sequential data. In *Proceedings of 10th International Conference on Human - Computer Interaction*, volume 4, pages 557–561, 2003.

[10] S. M. Katz. Estimation of probabilities from sparse data for hte language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing (ASSP)*, 35:400–401, 1987.

[11] K. Kersting, L. De Raedt, and T. Raiko. Logical hidden markov models. 2005. (submitted).

[12] K. Kersting, T. Raiko, S. Kramer, and L. De Raedt. Towards discovering structural signatures of protein folds based on logical hidden markov models. In *Proceedings of the Pacific Symposium on Biocomputing (PSB-03)*, 2003.

[13] K. Kersting, T. Raiko, and L. De Raedt. A Structural GEM for Learning Logical Hidden Markov Models. In *Working Notes of the Second KDD-Workshop on Multi-Relational Data Mining (MRDM-03)*, 2003.

[14] N. Landwehr, K. Kersting, and L. De Raedt. nFOIL: Integrating Naïve Bayes and Foil. In *Proceedings of AAAI-05*, 2005. To appear.

[15] T. Lane. Hidden Markov Models for Human/Computer Interface Modeling. In *Proceedings of the IJCAI-99 Workshop on Learning about Users*, pages 35–44, Stockholm, Sweden, July 1999.

[16] G. J. McKachlan and T. Krishnan. *The EM Algorithm and Extensions*. John Eiley & Sons, Inc., 1997.

[17] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19(20):629–679, 1994.

[18] C. Nédellec, C. Rouveirol, H. Adé, F. Bergadano, and B. Tausend. Declarative Bias in ILP. In L. De Raedt, editor, *Advances in Inductive Logic Programming*. IOS Press, 1996.

[19] M. Ostendorf and H. Singer. HMM topology design using maximum likelihood successive state splitting. *Computer Speech and Language*, 11(1):17–41, 1997.

[20] A. Popescul, L. H. Ungar, S. Lawrence, and D. M. Pennock. Statistical Relational Learning for Document Mining. In *Proceedings of ICDM-03*, pages 275–282, 2003.

[21] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 4–15, January 1986.

[22] A. Stolcke and S. Omohundro. Hidden Markov model induction by Bayesian model merging. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Adv. in Neural Information Processing Systems*, volume 5, pages 11–18, 1993.