

Taming the Unstructured: Creating Structured Content from Partially Labeled Schematic Text Sequences

Saikat Mukherjee and I.V. Ramakrishnan

Department of Computer Science,
Stony Brook University,
Stony Brook, NY 11794, U.S.A.
{saikat,ram}@cs.sunysb.edu

Abstract. Numerous data sources such as classified ads in online newspapers, electronic product catalogs and postal addresses are rife with unstructured text content. Typically such content is characterized by attribute value sequences having a common schema. In addition each sequence is unstructured free text without any separators between the attribute values. Hidden Markov Models (HMMs) have been used for creating structured content from such text sequences by identifying and extracting attribute values occurring in them. Extant approaches to creating “structured content from text sequences” based on HMMs use either completely labeled or completely unlabeled training data. The HMMs resulting from these two dominant approaches present contrasting trade offs w.r.t. labeling effort and recall/precision of the extracted attribute values. In this paper we propose a HMM based algorithm that uses partially labeled training data for creating structured content from text sequences. By exploiting the observation that partially labeled sequences give rise to independent subsequences we *compose* the HMMs corresponding to these subsequences to create structured content from the complete sequence. An interesting aspect of our approach is that it gives rise to a family of HMMs spanning the trade off spectrum. We present experimental evidence of the effectiveness of our algorithm on real-life data sets and demonstrate that it is indeed possible to bootstrap structured content creation from schematic text data sources using HMMs that require limited labeling effort and do so without compromising on the recall/precision performance metrics.

1 Introduction

It is quite common to find data sources especially on the Web, whose content is in free text. Examples include electronic postal addresses, classified ads in online newspapers, online product data catalogs, and so on. Figure 1(a) shows examples of free text descriptions of binder products at www.staples.com. In the “Item” column of Figure 1(a) binder products are uniquely described by six attributes: *Thickness*, *Manufacturer*, *Model*, *Ring Type*, *Category*, and *Quantity*.

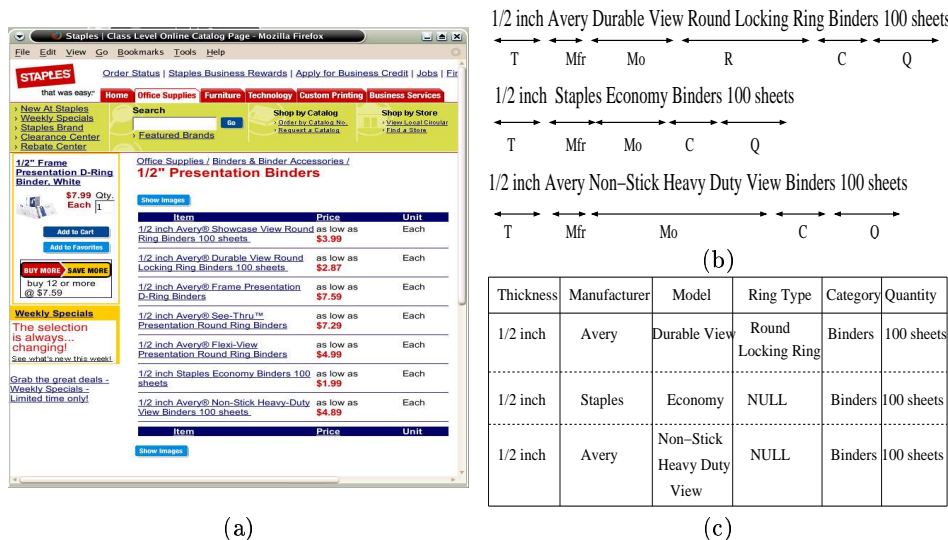


Fig. 1. (a) Staples Binders Web Page, (b) “Item” columns of a few Staples Binders Sequences from (a) having the attributes *Thickness* (*T*), *Manufacturer* (*Mfr*), *Model* (*Mo*), *Ring Type* (*R*), *Category* (*C*) and *Quantity* (*Q*), (c) Structured Content for the Sequences in (b).

These descriptions for three such binder products are shown in Figure 1(b). Each text fragment in Figure 1(b) describes a unique binder item in terms of its attribute values. For instance, the 1st text fragment describes a binder that has 1/2 inch thickness, manufactured by Avery, belonging to the durable view model, with round locking ring type and comes in quantities of 100 sheets. It is instructive to examine the salient characteristics of these descriptions: the text data are in fact attribute value sequences; there are no separators between the attribute values in any of the sequences; some attribute values may be missing.

Creating structured content from such text sequences corresponds to identifying and extracting, w.r.t. a schema, the attribute values in them. In the binder example above the schema consists of the attributes: *Thickness*, *Manufacturer*, *Model*, *Ring Type*, *Category*, and *Quantity*. Extracting the attribute values w.r.t. this schema from the sample sequences in Figure 1(b) will result in the structured content shown in Figure 1(c).

Automatic methods to structure text data is an important aspect of web-based electronic commerce. In particular empowering “shopbots” with such methods facilitates comparison shopping of similar product and service offerings on the Web.

Highly effective techniques using Hidden Markov Models (HMMs) for creating structured content from such text sequences have been recently reported [4, 17, 14, 15, 5]. A HMM is a probabilistic automaton with a finite set of states, each associated with transition, emission, and initial probability distributions.

The first is used to assign probability values to transitions between states while the second determines the probability of an outcome or observation (which is a string over some alphabet) at a given state, and the third determines the probability of initiating a sequence in a state. The automaton structure together with the three probability distributions constitute a *model*. Figure 2(a) is an example of a HMM structure. Given a HMM whose structure, transition, emission, and initial probabilities are known and an observation sequence (such as one of the product descriptions in Figure 1(b)) as its input, Viterbi’s algorithm [24] finds the best sequence of states corresponding to the input. In a model with one state per attribute, each state denotes a distinct attribute and the strings emitted by a state in the Viterbi sequence are the extracted values corresponding to the attribute associated with the state.

Creating structured content from text sequences using HMMs is now reduced to the problem of learning the model. This learning is concerned with synthesizing the automaton structure and parameter estimation, i.e. determining the probability distributions underlying state transitions, emissions of symbols from states, and initiating sequences from states. Most work in HMM assumes the model structure and estimates the parameters using a set of training sequences and so structured content creation is in essence the problem of estimating the model parameters.

The two dominant approaches to estimating the parameters differ mainly in how the training sequences are utilized. At one extreme is the *complete labeling* approach where each and every attribute value in all the training sequences are manually labeled. In contrast, the *unlabeled* approach at the other extreme (as exemplified by Baum-Welch [3]) uses no such labels. Not surprisingly parameters learned via the complete labeling approach, (which we will refer to as CL-HMMs), have been shown to yield high degrees of precision in extracting attribute values from unknown text sequences when compared to those learned without any labeling information (which we will refer to as BW-HMMs). On the other hand observe that training is a labor-intensive operation. Thus the contrasting trade-offs present at the two ends of the learning spectrum begs the question: Can we develop techniques for estimating HMM parameters that continuously trade precision for training effort? More interestingly can they be tuned so that with little training effort HMMs with very high precision characteristics can be learned? Using the notion of *partially labeled* training sequences in which only a subset of the attribute values are labeled, we can formulate this problem as one of estimating HMM parameters from partially labeled sequences. This is a topic that is relatively less explored in the research literature.

Note that the partial labels can be encoded as knowledge within a domain-specific ontology. Thus they need be constructed just once for a specific application domain. Hence the use of domain-specific ontologies for (partially) labeling sequences becomes a principal means to a scalable solution to structured content creation from text sequences.

Observe in Figure 1(b) that the order of attribute occurrences in all the binder sequences is fixed. For instance the *Manufacturer* attribute instances always

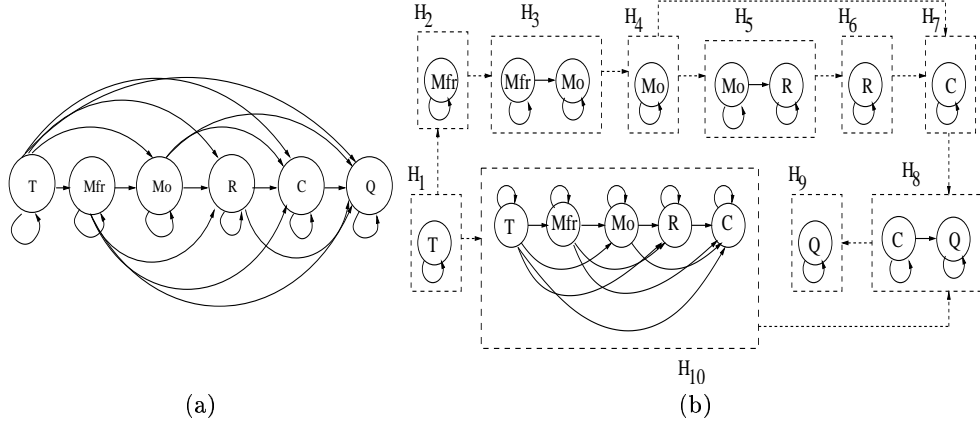


Fig. 2. (a) Hidden Markov Model Structure for the Staples Binders Sequences (b) Hierarchical Model Structure for Staples Binders Sequences

precede the *Quantity* attribute instances in every sequence. Such sequences have an implicit *common template*. We refer to such data sources as *schematic*.

Recall that we had reduced the problem of structured content creation to estimating the model parameters, namely the transition, emission, and initial probabilities. In this paper we focus on the problem of estimating HMM parameters from partially labeled sequences having an implicit common template (which we will refer to as PL-HMMs). In Section 2 we provide a brief introduction to hidden markov models. In Section 3 we propose an algorithm for estimating the parameters of PL-HMM by a compositional process. In particular we use the partial labels to *decompose* the (global) model structure of a PL-HMM into a set of local structures, estimate the parameters of these local models and then *compose* them to obtain the parameters for the global model. Our algorithm is parameterized in the sense that by varying the partial label set we can estimate the parameters of HMMs spanning the spectrum from CL-HMM at one end to BW-HMM at the other end. Preliminary experimental results, presented in Section 4, seem to suggest that with a small partial label set the performance of PL-HMM (in terms of precision and recall metrics) is superior to BW-HMM and comparable to CL-HMM. Related work and discussions appear in Sections 5 and 6 respectively.

2 Overview of HMM

A hidden markov model is a probabilistic finite state automaton with a set S of n states, a set V of m observation symbols, an array of initial probabilities where each i_j is the probability of starting a sequence from state s_j , a matrix of transition probabilities where each a_{ij} is the probability of a transition from s_i to s_j , and a matrix of emission probabilities where each b_{jk} is the probability

of emitting the observation symbol v_k in s_j . See [19] for an excellent tutorial on HMMs.

Given a model structure, (n , V , and state transitions) the model probabilities are estimated from training data which are either completely *labeled* or *unlabeled* observation sequences. When observation sequences are labeled, estimating the model parameters reduces to frequency counting. The Baum-Welch algorithm, an instantiation of the Expectation-Maximization technique [12], is widely used for estimation from unlabeled observation sequences. The idea behind Baum-Welch (or EM in general) is to start with initial values of parameters and in successive iterations refine them till a stationary point is achieved (see [19] for details). Central to this re-estimation is the value $\xi_t(i, j) = P(q_t = s_i, q_{t+1} = s_j | O, \lambda)$, which is the probability of being in state s_i at time t and in state s_j during time $t + 1$ given an observation sequence O and the model parameters λ . The value $\xi_t(i, j)$ is used for computing the following expectations given an observation sequence of length T and parameters of a model with n states:

- I_i : the expected number of times the sequence starts in state s_i . This is $\sum_{j=1}^{j=n} \xi_1(i, j)$.
- E_i : the expected number of times the sequence ends in s_i . This is $\sum_{j=1}^{j=n} \xi_{T-1}(j, i)$.
- A_{ij} : the expected number of transitions from s_i to s_j . This is $\sum_{t=1}^{t=T-1} \xi_t(i, j)$.
- B_{jk} : the expected number of times of being in state s_j and emitting the symbol v_k . This is $\sum_{t=1}^{t=T} \sum_{l=1}^{l=n} \xi_t(j, l)$, where the outer summation is over only those t where O_t is v_k .

3 Structuring Unstructured Sequences

Templates: The structure of our HMM associates a single state with every attribute. In sequences having a common template, the order in which attribute values appear in every sequence is consistent. For instance, in Figure 1(b), the consistent order is: *Thickness* \prec *Manufacturer* \prec *Model* \prec *Ring Type* \prec *Category* \prec *Quantity*. Given such a total order on attributes the structure of the HMM is an ordered sequence of states, one for each attribute, mirroring the order of attributes. In such a structure a state makes a transition to itself and to all higher states in the order. Figure 2(a) shows the model structure for the sequences in Figure 1(b). Given a template, we will assume this model structure for a PL-HMM and proceed to estimate its parameters using partially labeled observation sequences following that template.

Partial Labeling: In observation sequences that are *partially labeled* only a subset of symbols are labeled with their corresponding states. These symbols can be identified in the sequences with a domain-specific *ontology*. For our work, this ontology is assumed to have a set of attributes and a labeling function for each attribute that *unambiguously* identifies a subset of its occurrences in the observation sequences. Given a set of sequences, we assume that at least one value from the domain of values for an attribute can be identified by its labeling function.

In general, a partially labeled observation sequence PO can be represented as $\gamma_1 \cdot \alpha_{1,s_{i_1}} \cdots \gamma_n \cdot \alpha_{n,s_{i_n}} \cdot \gamma_{n+1}$, where γ_i denotes a consecutive sequence of unlabeled symbols, and $\alpha_{i,s_{i_i}}$ denotes a consecutive sequence of symbols labeled with the state s_{i_i} . When all the γ_i 's are null then PO collapses to a completely labeled sequence and when all the $\alpha_{i,s_{i_i}}$ values are null PO collapses to a completely unlabeled sequence.

Figure 3 shows ontologies for the Binders and US Addresses domains. Applying the Binders ontology on the first sequence in Figure 1(b) will identify the symbols *inch* (*Thickness*), *Avery* (*Manufacturer*), *View* (*Model*), *Ring* (*Ring Type*), *Binders* (*Category*), and *sheets* (*Quantity*). In the representation of this partially labeled observation sequence γ_1 corresponds to $1/2$, $\alpha_{1,s_{i_1}}$ corresponds to *inch* where s_{i_1} is the state for the attribute *Thickness*, γ_2 is null, $\alpha_{2,s_{i_2}}$ is *Avery* with s_{i_2} being the state for the attribute *Manufacturer*, γ_3 is *Durable*, $\alpha_{3,s_{i_3}}$ is *View* where s_{i_3} is the state for the attribute *Model*, γ_4 is *Round Locking*, $\alpha_{4,s_{i_4}}$ is *Ring* with s_{i_4} being the state for *Ring Type*, γ_5 is null, $\alpha_{5,s_{i_5}}$ is *Binders* with s_{i_5} being the state for the attribute *Category*, γ_6 is *100*, $\alpha_{6,s_{i_6}}$ being *sheets* and s_{i_6} being the state for the attribute *Quantity*, and finally γ_7 is null.

Parameter Estimation: Note that in sequences with a common template, a subsequence of consecutive unlabeled observation symbols γ_i between two neighbouring labeled phrases $\alpha_{i-1,s_{i_{i-1}}}$ and $\alpha_{i,s_{i_i}}$ can only be generated from the sequence of states inclusive and between $s_{i_{i-1}}$ and s_{i_i} in the global model structure. For instance, in the second sequence in Figure 1(b), the symbol *inch* is labeled with the attribute *Thickness* and the symbol *Binders* is labeled with the attribute *Category*. Given the model structure in Figure 2(a), the symbols *Staples* and *Economy* can only be generated from the states corresponding to the attributes *Thickness*, *Manufacturer*, *Model*, *Ring Type* and *Category*. This observation lets us *decompose* the global HMM structure into local HMM structures where the local HMMs emit subsequences from the original observation sequences.

The idea behind decomposition is to split up the global HMM into multiple local HMMs such that each local HMM emits either labeled observation phrases or unlabeled observation phrases. A local HMM is connected by transitions to other local HMMs. For instance, the local HMM emitting a labeled phrase in an observation sequence will have a transition to the local HMM which emits the unlabeled phrase immediately following the labeled phrase in the same observation sequence. The local HMMs themselves maybe composed of single or multiple states. For instance, a local HMM emitting a consecutive sequence of observation symbols labeled by the same state consists of just that state. On the other hand, a local HMM emitting an unlabeled observation phrase between two labeled symbols consists of the sequence of states from the global model which lie between and including the states corresponding to the two labeled symbols. In particular, the order in which these states will appear in the local HMM will be the same in which they appear in the global HMM structure. Moreover, all and only the transitions between these states in the global model are carried

over into the local HMM. Intuitively, the structure of a local HMM reflects a certain contiguous segment of the global HMM structure.

The states inside a local HMM emit single observation symbols which may be labeled or unlabeled. This manner of decomposition results in a two-level hierarchical HMM with the outer level corresponding to the local HMMs that emit phrases and the inner level corresponding to states inside the local HMMs that emit single observation symbols.

This decomposition process is encoded in Lines 1-13 in *Algorithm PartialLabelTrain*. The algorithm takes as input a set of partially labeled sequences and the structure of the global HMM. The construct *localHMM* takes a consecutive sequence of observation symbols and a consecutive sequence of states as input. It creates a new local HMM if one does not exist for this input sequence of states; otherwise the existing local HMM is reused. The input observation symbol sequence becomes the emission phrase for this local HMM. The local HMM emitting the j th labeled phrase, $\alpha_{j,s_{i_j}}$, in a partially labeled sequence is denoted by $H_{j,L}$ while the local HMM emitting the unlabeled phrase preceding it, γ_j , is denoted by $H_{j,U}$. For every $\alpha_{j,s_{i_j}}$ $H_{j,L}$ is created with one inner state s_{i_j} and with $\alpha_{j,s_{i_j}}$ as its emission phrase (Line 3). If the unlabeled phrase γ_j preceding $\alpha_{j,s_{i_j}}$ exists, then $H_{j,U}$ is created with the sequence of states from $s_{i_{j-1}}$, the state corresponding to $\alpha_{j-1,s_{i_{j-1}}}$, to s_{i_j} present in the global model structure M (Line 5). Transitions are created from $H_{j,U}$ to $H_{j,L}$ and from $H_{j-1,L}$, the local HMM corresponding to the previous labeled phrase $\alpha_{j-1,s_{i_{j-1}}}$ in the sequence, to $H_{j,U}$ (Lines 6-7). If on the other hand γ_j is null, then the local HMM $H_{j-1,L}$ is directly linked to $H_{j,L}$ (Line 8). The first local HMM in every sequence, which can be either $H_{1,U}$ or $H_{1,L}$ depending upon the presence or absence of γ_1 respectively, is marked *init* which indicates the start of the sequence from that HMM (Line 11-12). Boundary conditions, such as dealing with the local HMM for γ_{n+1} and the local HMM $H_{0,L}$ at the beginning of a sequence, have been skipped in *Algorithm PartialLabelTrain* for reasons of simplicity.

Decomposition is illustrated on the sequences in Figure 1(b) using the global model in Figure 2(a) and labels from the Binders ontology in Figure 3. The resulting hierarchical HMM is shown in Figure 2(b). In the first sequence in Figure 1(b), the symbols *inch*, *Avery*, *View*, *Ring*, *Binders*, and *sheets* are labeled by the attributes *Thickness*, *Manufacturer*, *Model*, *Ring Type*, *Category* and *Quantity* respectively. These labeled symbols create the single state local models H_1 , H_2 , H_4 , H_6 , H_7 , and H_9 respectively¹. Next local HMMs are created for the unlabeled phrases between these labeled symbols. The unlabeled symbol *1/2* can only be emitted from the state for attribute *Thickness* (H_1) while the unlabeled symbol *Durable* can be emitted from the states for the attributes *Manufacturer* or *Model* (H_3), and the unlabeled phrase *Round Locking* can be emitted from the states for the attributes *Model* and *Ring* (H_5), and finally the unlabeled symbol *100* can be emitted from the states for the attributes *Category*

¹ For purposes of clarity the labeled or unlabeled subscript is dropped from the notation of a local HMM

or *Quantity* (H_8). For the second sequence in Figure 1(a), *1/2 inch* is emitted from H_1 and *Binders* from H_7 . The unlabeled phrase *Staples Economy* can be emitted from the states for the attributes *Thickness*, *Manufacturer*, *Model*, *Ring Type* and *Category* (H_{10}). Similarly, for the third sequence in Figure 1(a), the unlabeled phrase *Non-Stick Heavy Duty* can only be emitted from states for the attributes *Manufacturer* and *Model* (H_3).

Algorithm PartialLabelTrain(PO, M)

input

- PO : a set of partially labeled sequences
where each $PO_i = \gamma_1 \cdot \alpha_{1, s_{i1}} \cdots \gamma_n \cdot \alpha_{n, s_{in}} \cdot \gamma_{n+1}$
- M : the global HMM structure

output

- M : the global HMM with estimated parameters

begin

1. **for** each PO_i **do**
2. **for** $1 \leq j \leq n$ **do**
3. $H_{j,L} = \text{localHMM}(\alpha_j, \langle s_{ij} \rangle)$
4. **if** $\gamma_j \neq \epsilon$ **then**
5. $H_{j,U} = \text{localHMM}(\gamma_j, \langle s_{i_{j-1}} \cdots s_{ij} \rangle)$
6. create transition from $H_{j,U}$ to $H_{j,L}$
7. create transition from $H_{j-1,L}$ to $H_{j,U}$
8. **else** create transition from $H_{j-1,L}$ to $H_{j,L}$
9. **endif**
10. **endfor**
11. **if** $\gamma_1 \neq \epsilon$ **then** mark $H_{1,U}$ *init*
12. **else** mark $H_{1,L}$ *init* **endif**
13. **endfor**
14. train every H_i by Baum-Welch
15. estimate transition prob. between every H_i, H_j
16. **for** all states H_i marked as *init* **do**
17. **for** all states h_{ij} in H_i **do**
18. $\text{initCount}(g_i(h_{ij})) += \text{initCount}(h_{ij})$
19. **endfor**
20. **endfor**
21. **for** all H_i and for every pair of states h_{ij}, h_{ik} in it **do**
22. $\text{transCount}(g_i(h_{ij}), g_i(h_{ik})) += \text{transCount}(h_{ij}, h_{ik})$
23. **endfor**
24. **for** all H_i s.t. H_i has one state h_{i1} **do**
25. **for** all H_j which has an edge from H_i **do**
26. **for** all states h_{jk} in H_j **do**
27. $\text{transCount}(g_i(h_{i1}), g_j(h_{jk})) += p_{H_i H_j} \times \text{initCount}(h_{jk})$
28. **endfor**
29. **for** all H_j which has an edge to H_i **do**
30. **for** all states h_{jk} in H_j **do**
31. $\text{transCount}(g_j(h_{jk}), g_i(h_{i1})) += p_{H_j H_i} \times \text{endCount}(h_{jk})$
32. **endfor**
33. **endfor**
34. **for** all H_i and for all states h_{ij} in it **do**
35. **for** all observation symbols O_m in the vocabulary **do**
36. $\text{emitCount}(g_i(h_{ij}), O_m) += \text{emitCount}(h_{ij}, O_m)$
37. **endfor**
38. **endfor**
39. Compute maximum likelihood parameter values from count values

end

The parameters of the local HMMs with multiple states are estimated by Baum-Welch [3]. The emission probabilities of local HMMs with a single state are estimated using maximum-likelihood frequency counting. For every γ_i and for

every $\alpha_{i,s_{i_i}}$ in a partially labeled observation sequence, $\dots\gamma_i.\alpha_{i,s_{i_i}}\dots$, the local HMMs emitting them are known. Thus, the transition probabilities between local HMMs can be computed using maximum-likelihood frequency counting (Line 15). We denote the transition probability between local HMMs H_i and H_j by p_{H_i,H_j} .

The estimation of the probability of an event, which can be a transition between two states or emission of a symbol from a state or initiating a sequence from a state, is determined from the expected number of times the event occurs in all the sequences. To compute the parameters of the global HMM it is necessary to compute the expectations of these events. Recall from the brief review of Baum-Welch in Section 2 that computing the expectation of an event in a sequence entails summing the probability of the event over the entire sequence. However, observe that in a schematic partially labeled sequence $\dots\alpha_{j-1,s_{i_{j-1}}}\cdot\gamma_j.\alpha_{j,s_{i_j}}\dots$, transitions between the states after $s_{i_{j-1}}$ and before s_{i_j} in the global HMM can only occur in the subsequence γ_j . Similarly, the emission expectations for the states between $s_{i_{j-1}}$ and s_{i_j} is positive only for symbols in the subsequence γ_j . Also, the expectation for initiating a sequence is positive only for the sequence of states from s_1 to s_{i_1} . Based on these observations we can compute the expectations for the events separately for each subsequence and compose them into expectation values for the global sequence.

Every local HMM in the two-level hierarchical HMM emits labeled or unlabeled phrases. Thus computing the expectation of an event occurring in a labeled or unlabeled phrase in an observation sequence reduces to computing the expectation of the event in the local HMM emitting that phrase. For every local HMM these expectations, viz. *initCount* (initiating a sequence in a state), *transCount* (transitions between states), *emitCount* (emitting a symbol in a state) and *endCount* (ending a sequence in a state), are computed by training it with the Baum-Welch algorithm. These expectations from the local HMMs are aggregated to generate the expectations of the global model (see Lines 16-38 in *Algorithm PartialLabelTrain*). A local HMM is denoted by H_i and its j th inner state is denoted by h_{ij} . To relate the expectations of events in the local HMMs to the expectation values for the global HMM we will have to relate every inner states in local HMMs to their corresponding global states. We use the function $g_i(h_{ij})$ to map the inner state h_{ij} in the local HMM H_i to its corresponding global state.

Lines 16-20 in *Algorithm PartialLabelTrain* encode the aggregation of the expected values for initiating a sequence in a state. The *initCount* of any state h_{ij} in a local HMM H_i marked as *init* is added to h_{ij} 's corresponding global state's *initCount* (Line 18). Intuitively, expectation of initiating a sequence is non-zero only for states in local HMMs where the sequences begin. For instance, for the sequences in Figure 1(b) and the hierarchical HMM in Figure 2(b), H_1 is the only state marked as *init* as all the three sequences begin from it. So the state for the attribute *Thickness* is the only state in the global HMM which accrues *initCount* values.

The transition expectation A_{ij} between any two states s_i, s_j , such that $s_i \prec s_j$ in the template of states, depends on their labeling in a sequence. The pair of states s_i, s_j in a partially labeled sequence are either both labeled, or only one is labeled, or neither are labeled. Accordingly A_{ij} is computed from:

- A sequence where s_i, s_j are both identified. For any such sequence, let O_m and O_n be the observation symbols labeled with s_i, s_j respectively. A transition between s_i, s_j can only occur if there are no other symbols labeled between O_m, O_n . For such a sequence,

$$\begin{aligned} A_{ij} &= \sum_{t=m}^{t=n-1} \xi_t(i, j) \\ &= \xi_m(i, j) + \sum_{t=m+1}^{t=n-2} \xi_t(i, j) + \xi_{n-1}(i, j). \end{aligned}$$

Decomposing the subsequence $O_m \dots O_n$ creates a local HMM H_i with a single state s_i emitting the symbol O_m , a local HMM H_j with the sequence of states from s_i to s_j in the template and emitting the sequence of symbols O_{m+1}, \dots, O_{n-1} , and a local HMM H_k with the single state s_j emitting the symbol O_n . Observe that $\sum_{t=m+1}^{t=n-2} \xi_t(i, j)$ can be computed from the results of applying Baum-Welch to the local HMM H_j . $\xi_m(i, j)$ (where $q_m = s_i$ and $q_{m+1} = s_j$) can occur if a transition is made from H_i to H_j and the subsequence $O_{m+1} \dots O_{n-1}$ is emitted starting from s_j in H_j . Similarly, $\xi_{n-1}(i, j)$ can occur if the subsequence $O_{m+1} \dots O_{n-1}$ is emitted from H_j ending in s_j followed by a transition from H_j to H_k . Consequently,

$$A_{ij} = p_{H_i H_j} \times I_j^j + A_{ij}^j + E_i^j \times p_{H_j H_k},$$

where I_j^j denotes the expectation of initiating a sequence from s_j in local HMM H_j , A_{ij}^k denotes the transition expectation between s_i, s_j in H_j , and E_i^k denotes the expectation of ending a sequence in s_i in local HMM H_k .

- A sequence where only s_i is identified. In such a sequence, a transition between s_i, s_j can occur only if there is no other state s_l , $s_i \prec s_l \prec s_j$, labeled. Let O_m, O_n be the observation symbols labeled with s_i, s_q respectively where $s_i \prec s_j \prec s_q$ and there does not exist any $s_l, s_j \prec s_l \prec s_q$, which is identified. For this sequence,

$$\begin{aligned} A_{ij} &= \sum_{t=m}^{t=n-2} \xi_t(i, j) \\ &= \xi_m(i, j) + \sum_{t=m+1}^{t=n-2} \xi_t(i, j). \end{aligned}$$

Decomposition of the subsequence $O_m \dots O_n$ yields the local HMM H_i with a single state s_i and emitting the symbol O_m , the local HMM H_j with the sequence of states from s_i to s_q in the template and emitting the subsequence $O_{m+1} \dots O_{n-1}$, and the local HMM H_k with the state s_q and emitting O_n . Computing the expectations from the local HMMs yields the result:

$$A_{ij} = p_{H_i H_j} \times I_j^j + A_{ij}^j.$$

- A sequence where only s_j is identified. Similar to the case above, the expectation is composed from the local HMMs as follows:

$$\begin{aligned} A_{ij} &= \sum_{t=m}^{t=n-2} \xi_t(i, j) \\ &= \sum_{t=m+1}^{t=n-2} \xi_t(i, j) + \xi_{n-1}(i, j) \\ &= A_{ij}^j + E_i^j \times p_{H_j H_k}. \end{aligned}$$

- A sequence where neither s_i nor s_j are identified. For such a sequence, a transition between s_i, s_j can only occur if there is no other state $s_l, s_i \prec s_l \prec s_j$, identified. If O_m, O_n are the observation symbols labeled with s_p, s_q respectively where $s_p \prec s_i \prec s_j \prec s_q$ in the template, then:

$$\begin{aligned} A_{ij} &= \sum_{t=m+1}^{t=n-2} \xi_t(i, j) \\ &= A_{ij}^j, \end{aligned}$$

where A_{ij}^j is the transition expectation computed from the local HMM H_j which has the sequence of states $s_p \dots s_q$ in the template and emitting the subsequence $O_{m+1} \dots O_{n-1}$.

Based on the above observations, computing the transition expectation between a pair of states in the global HMM involves computing the values A_{ij}^j , $p_{H_i H_j} \times I_j^j$, and $E_i^j \times p_{H_j H_k}$. Lines 21-33 in *Algorithm PartialLabelTrain* illustrate the computation of the above values for all pairs of states in the global HMM. The values A_{ij}^j are computed in Lines 21-23, while $p_{H_i H_j} \times I_j^j$ and $E_i^j \times p_{H_j H_k}$ are computed in Lines 24-33. The values computed are summed up to generate the final transition expectation between pairs of states. For instance, for the sequences in Figure 1(b) and the hierarchical HMM in Figure 2(b), the expected number of times a transition occurs between the global states for the attribute *Manufacturer* and the attribute *Model* is estimated by adding the transition expectations between these two states within H_3 and H_{10} , and also between *Manufacturer*'s state in H_2 and *Model*'s state in H_3 . Similarly, the transition expectation between the global states for the attributes *Category* and *Quantity* is estimated by adding the expectation between these states within H_8 , and the expectation between *Category*'s state in H_7 and *Quantity*'s state in H_8 .

Lines 34-38 in *Algorithm PartialLabelTrain* encode the aggregation of emission expectations. For every state h_{ij} in a local HMM H_i the *emitCount* for a particular emission symbol O_m , which is emitted singly or as part of a phrase in H_i , is added to the *emitCount* of the corresponding global state of h_{ij} and for the symbol O_m (Lines 20-26). For instance, in Figure 2(b), H_{10} emits the phrase *Staples Economy*, H_7 emits the symbol *Binders* and H_8 emits the symbol *100*. Thus, the emission probabilities of the global state for the attribute *Category* are distributed over the observation symbols *Staples*, *Economy*, *Binders*, and *100* as H_{10}, H_7 , and H_8 are the only local HMMs where the state for *Category* occurs. The expected number of times this state emits these symbols are estimated from its *emitCounts* of these symbols in H_{10}, H_7 , and H_8 .

Finally, the *initCount*, *transCount*, and *emitCount* expectations computed for the states in the global HMM are used to calculate the initial, transition, and emission parameters by simple maximum-likelihood methods (Line 39). In the structured content creation process, given a set of schematic text sequences a domain ontology is used to partially label them and estimate the parameters of the HMM. The trained HMM can then be used to segment these sequences into structured attribute value pairs with the Viterbi algorithm [19].

US Addresses	
House Number	Post Office, Box, PO, P.O.
Street	Street St St. Avenue Ave Ave. Av Av. Boulevard Blvd Blvd. Drive Dr
Name	Highway Hwy Hwy. Pl Pl. Plaza Road Rd Rd. Route Rt Rt. Square Sq Turnpike Tpke Tpke. Us US Main Court Place Way Trail Lane
City	New York, Chicago, Charlotte
State	NY, NJ, MA, MD, CT, FL, IL, NC, OH
Zip Code	Regular Expression of Five Digit Number
Binders	
Manufacturer	Avery, Cardinal, Wilson Jones
Model	View, Heavy-Duty
Ring Type	Ring, Rings
Thickness	inch
Category	Binder, Binders
Quantity	Sheets

Fig. 3. Ontologies for US Addresses and Binders with Attribute name in Column 1 and the labels in Column 2

4 Experimental Results

Datasets and Ontologies: We used two datasets for our experimental evaluation, namely, sequences of Office Binders and US address data. 509 sequences in the Binders dataset were scraped from the websites of Staples (183 sequences), OfficeMax (83), and Office Depot (243). Each vendor’s binder data sequence followed a common template; however each vendor had a different template. So we built 3 different HMMs for binder data – one per vendor– estimated their parameters separately and evaluated them against several metrics. The results of these evaluations were aggregated over all the three vendors. 481 US postal address sequences were collected from www.superyellow.com, an online yellow pages repository. All these address sequences followed a single common template. Therefore we built a single HMM for the address data and evaluated it w.r.t. the same metrics as used for Binders.

Overall Performance: For evaluating performance we deployed PL-HMM to extract attribute values from schematic data sequences. We used Viterbi’s algorithm [19] to run the data sequences through PL-HMM and gathered the recall/precision numbers produced by these runs (see Figure 4).

For (partially) labeling the data sequences we used two ontologies, one for binders and the other for addresses (see Figure 3). Notice that the essential information in both the ontologies is the labeling function (one per attribute) that is used to label the attribute values in the data sequences. For the zip code attribute in the address ontology we used a regular expression for identifying 5 digit numbers as its labeling function. For all of the remaining attributes in both the ontologies keywords were used to search and label matching attribute occurrences in the data sequences.

In the tables a token corresponds to a word in a data sequence (e.g. $1/2$ *inch* has two tokens: $1/2$ and *inch*). The 2nd column in both the tables denotes

Attribute	Tokens Present	Precision (%)	Recall (%)
Manufacturer	688	87.23	94.33
Model	675	96.25	60.89
Ring Type	821	91.50	94.40
Thickness	1017	86.40	99.90
Category	514	96.96	99.42
Quantity	353	100	98.02
Overall	4068	91.15	91.15

(a)

Attribute	Tokens Present	Precision (%)	Recall (%)
House Number	499	70.70	97.19
Street Name	1120	97.06	88.48
City	612	92.49	62.42
State	481	71.96	93.35
Zip Code	481	100	93.35
Overall	3193	86.31	86.31

(b)

Fig. 4. (a) Recall and Precision for Binders Data (b) Recall and Precision for US Addresses

the total count of tokens present in all of the attribute value occurrences of an attribute in the dataset (e.g. in Figure 4(a) 688 is the total count of the tokens making up manufacturer names in binders data).

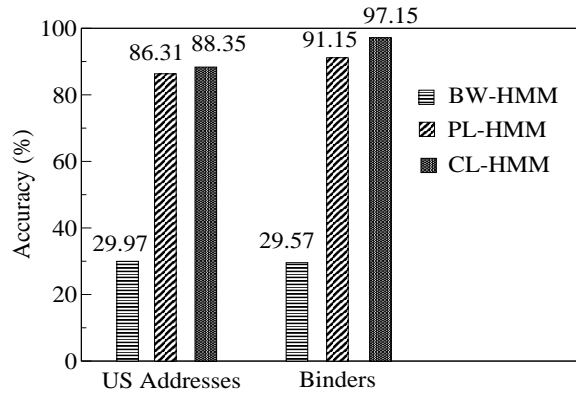


Fig. 5. Comparison of PL-HMM with CL-HMM and BW-HMM

Comparison of PL-, CL- and BW-HMMs: Figure 5 compares the extraction accuracy of PL-, CL- and BW-HMM. Accuracy is defined as the precision of extraction over all the attributes. Notice that PL-HMM outperforms BW-HMM and is comparable to CL-HMM. The interesting question is: where are the gains in PL-HMM? Below we provide experimental evidence that high-performance (in terms of recall/precision) PL-HMMs can be built with little labeling effort.

A few remarks about these results: Observe the high recall and precision numbers for *Category* and *Quantity* in Figure 4(a). In almost any sequence in the binders dataset very few tokens were present in instances of these two attributes when compared to the other attributes. Hence, even with a limited

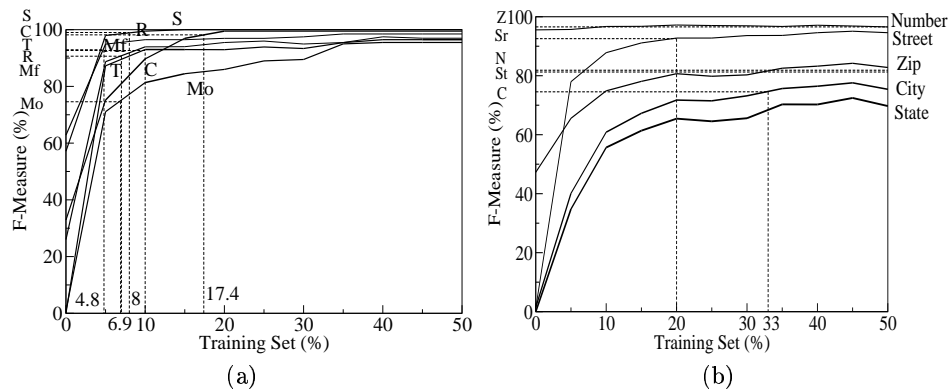


Fig. 6. (a) Training vs. F-measure for attributes in Binders domain (b) Training vs. F-measure for attributes in US Addresses

number of labels in the ontology, it is possible to achieve high recall/precision numbers with PL-HMMs. But also notice the low precision and recall numbers respectively for *Manufacturer* and *Model*. In almost all the sequences these two attribute instances are adjacent. The number of labels used in the ontology for these two attributes were very few thereby limiting the degree of discernibility between these attribute instances. So *Model* instances were identified as *Manufacturer* instances and vice versa. Similar reasoning carries over for the low recall/precision values associated with *House Number* and *Street Name* as well as *City* and *State* in Figure 4(b). Nevertheless, the high precision obtained in extracting attribute values from both the datasets validates the utility of using a very limited set of labels for estimating PL-HMM parameters.

Comparison of Training for Partial vs. Complete Labeling: Notice that in PL-HMMs the ontology used for partial labeling is built once for a domain. The use of the ontology eliminates the need for manual labeling of the data sequences. This step cannot be avoided for CL-HMMs. So an interesting question is how much savings in training effort is gained by using PL-HMMs. To answer this question we measured the degree of training needed for learning CL-HMMs to achieve comparable recall/precision of PL-HMMs (see Figure 6). We varied the size of the training set and measured the recall/precision of the resulting CL-HMM built using that training set (the solid curves in the figure). The recall/precision were combined into the f-measure, defined as the harmonic mean of recall and precision. Each solid curve corresponds to a unique attribute. To determine the degree of training needed for CL-HMMs to achieve comparable recall/precision of PL-HMMs we proceed as follows: We choose an attribute and obtain its f-measure value, say α , recorded with PL-HMMs. Next we pick the solid curve in the figure corresponding to this attribute and identify α on this curve. The training set size is the projection of α on the X-axis.

We observe that for the binders domain the training set can vary from 4.8% to over 17.4% to achieve comparable f-measure values which corresponds to

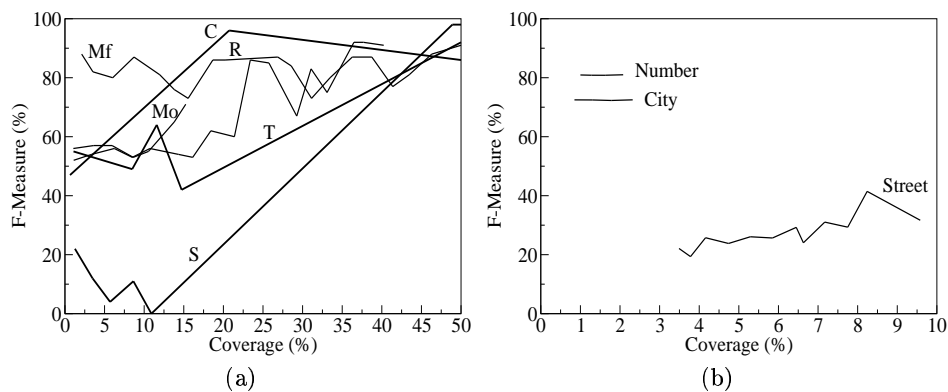


Fig. 7. (a) Coverage against F-measure in the Binders domain (b) Coverage against F-measure in the Address domain

completely labeling 89 sequences. Contrast this to constructing an ontology of 6 attributes values once.

We noticed that when attribute instances have few tokens a small training set suffices to achieve comparable f-measure values. (e.g. *Manufacturer*, *Thickness* in the binders domain). In the addresses domain, a training set of 33% (which corresponds to a completely labeling 159 sequences) was required. This is because even though instances of *City* have few tokens in them these tokens repeat infrequently across different addresses. Thus, the greater number of tokens in the vocabulary of *City* necessitates a bigger training set. A 20% training set was required for *Street Name* due to the the presence of many tokens in its instance values. Finally notice that the regular expression for *Zip Code* achieves a f-measure which cannot be matched even with a 50% training set. This is because individual zip codes rarely recur in the data set and so a high training set is necessary.

Effect of Label Selection: An interesting question concerning PL-HMMS is: to what extent does the quality of an ontology affect performance? In our case the only ontology parameter that we could experiment with were the keywords. So we fixed the number of keywords for every attribute in the ontology but used different keywords for partial labeling. Figure 7 shows the effect of label selection on the f-measure of the attribute. Coverage of the labels for an attribute is defined as the ratio of the number of times the labels occur in the data set to the total count of all the tokens present in the instances of that attribute in the data set. The higher the coverage the more frequently the labels occur in the tokens for that attribute.

Observe from Figure 7 that the f-measures of the attributes *Number*, *City*, and *Mf* (*Manufacturer*) are high (close to 80%) even at low coverage. Instances of these attributes have few tokens in them and as such label selection does not significantly affect the performance. However, f-measure is considerably affected for the attributes *S* (*Quantity*), *T* (*Thickness*), and *C* (*Category*). These attributes

have a very limited vocabulary besides having a few tokens in their instances. If these tokens are not selected as labels then the f-measure is poor while their selection creates a spike in the f-measure curve as shown in Figure 7(a). It seems empirically that while label selection is important it is particularly critical for attributes with limited vocabulary and small instances. However, one can easily pick the proper labels for these attributes because of their limited vocabulary.

5 Related Work

The primary area related to our work is information extraction from unstructured text which is an area of active research. Principal approaches to tackling this problem have involved either rule-based natural language processing [9, 11, 22, 7, 1] or statistical techniques. In the latter category there has been significant research on extraction based on HMMs [4, 17, 14, 15, 5]. In all of these works either completely labeled or completely unlabeled observation sequences were used to estimate the model parameters. In [14], a statistical technique called shrinkage was used to refine the emission probability distributions of states with limited number of labeled observation symbols. However, it still requires labeled observation sequences for estimating the transition probabilities, and the first approximation of the emission probabilities. The fundamental difference between all these and our work is the use of partial labels incorporated in a domain ontology for estimating the parameters (initial, transition, and emission) of HMMs for sequences with a common template.

In [20] a technique for estimating HMM parameters from partially labeled sequences was proposed. The technique was based on a modification to the Baum-Welch algorithm to incorporate partial label information. The fundamental difference between their work and ours is the assumption of a template structure for our hidden markov models. The consequence of this assumption is outlined in more details in Section 6.

In an upcoming paper [2], that we became aware of after submitting our work to this conference, a HMM-based technique for segmenting unstructured text sequences into attribute value pairs was proposed via the use of pre-existing “clean” attribute values in databases. In their technique they learn a single HMM for every individual attribute. However, the examples for learning these HMMs have to be completely labeled. In contrast, we do not depend on such complete labels for attributes.

The concept of hierarchical HMMs and estimation of their parameters from unlabeled sequences was proposed in [13] and used for information extraction [21] and chunking [6]. Our work uses the hierarchical HMM idea to learn the model parameters of a non-hierarchical HMM from partially labeled sequences.

Conditional markov chains [18] and fields [16], even though trained from completely labeled sequences, have been recently proposed and shown to perform better than hidden markov models for the information extraction task. It would be interesting to investigate the application of partially labeled sequences for training these models.

6 Discussions

Knowledge of any domain evolves over time. Thus, it is reasonable to expect that the set of partial labels, i.e. the ontology, is not static. Moreover, in dynamic domains new concepts and attributes continue to be incorporated in the evolving ontology. Incorporating the new domain information in techniques, as in [20] and briefly mentioned in [18], which rely on modification of Baum-Welch for parameter estimation is not easy. Essentially the algorithm has to be re-run on all the data sequences with the set of old and new labels. In contrast, our compositional technique need only re-estimate parameters of the local HMMs concerned with the new labels, their immediate neighbours, and the transition probabilities between these local HMMs. With a compositional estimation algorithm like that described in this paper, sequences with a common template lend themselves naturally to localization of such re-estimation.

Although the model structure is usually assumed to be known, encoding the structure is a manual step. An interesting problem is to learn the structure automatically. Since the model is a total order on the attributes present in the data set, it seems feasible to develop techniques for learning total orders from a set of partial orders as described in [10]. Order learning heuristics were also investigated in [2]. Another possibility is to specialize the state-merging techniques described in [15, 5, 23, 8] for learning model structures from partially labeled sequences. Such a learning algorithm coupled with our compositional technique for estimating parameters will represent a fully automatic scalable solution to creating structured content from text sequences using domain-specific ontologies consisting of partial labels.

Acknowledgments: Research partially supported by NSF grants CCR-0311512, 0205376, IIS-0072927, U.S. Army Medical Research Acquisition Activity Contract DAMD17-03-1-0520, and ONR award N000140110967.

References

1. B. Adelberg. Nodose: A tool for semi-automatically extracting structured and semi-structured data from text documents. In *ACM Conf. on Management of Data (SIGMOD)*, 1998.
2. E. Agichtein and V. Ganti. Mining reference tables for automatic text segmentation. In *ACM Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, 2004.
3. L. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process. *Inequalities*, 3:1–8, 1972.
4. Bikel, Schwartz, and Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1):211–231, 1999.
5. V. Borkar, K. Deshmukh, and S. Sarawagi. Automatic segmentation of text into structured records. In *ACM Conf. on Management of Data (SIGMOD)*, 2001.
6. T. Brants. Cascaded markov models. In *European Chapter of the Association for Computational Linguistics (EACL)*, 1999.
7. M. Califf and R. Mooney. Relational learning of pattern-match rules for information extraction. In *National Conf. on Artificial Intelligence (AAAI)*, 1999.

8. R. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In *Intl. Colloquium on Grammatical Inference and Applications (ICGI)*, 1994.
9. W. Cohen. Fast effective rule induction. In *Intl. Conf. on Machine Learning (ICML)*, 1995.
10. W. Cohen, R. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.
11. W. Cohen and Y. Singer. Simple, fast, and effective rule learner. In *National Conf. on Artificial Intelligence (AAAI)*, 1999.
12. A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(B):1–38, 1977.
13. S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, 32:41–62, 1998.
14. D. Freitag and A. McCallum. Information extrctation using hmms and shrinkage. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 31–36, 1999.
15. D. Freitag and A. McCallum. Information extraction with hmm structures learned by stochastic optimization. In *National Conf. on Artificial Intelligence (AAAI)*, 2000.
16. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Intl. Conf. on Machine Learning (ICML)*, 2001.
17. T. Leek. Information extraction using hidden markov models. In *Master's thesis UC San Diego*, 1997.
18. A. McCallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Intl. Conf. on Machine Learning (ICML)*, 2000.
19. L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2), 1989.
20. T. Scheffer and S. Wrobel. Active learning of partially hidden markov models. In *ECML/PKDD Workshop on Instance Selection*, 2001.
21. M. Skounakis, M. Craven, and S. Ray. Hierarchical hidden markov models for information extraction. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, 2003.
22. S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3), 1999.
23. A. Stolcke and S. Omohundro. Hidden markov model induction by bayesian model merging. In *Advances in Neural Information Processing Systems (NIPS)*, 1992.
24. A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–267, 1967.