Information Extraction with HMM Structures Learned by Stochastic Optimization

Dayne Freitag and **Andrew McCallum**
Just Research
4616 Henry Street
Pittsburgh, PA 15213
{dayne,mccallum}@justresearch.com

Abstract

Recent research has demonstrated the strong performance of hidden Markov models applied to *information extraction*—the task of populating database slots with corresponding phrases from text documents. A remaining problem, however, is the selection of state-transition structure for the model. This paper demonstrates that extraction accuracy strongly depends on the selection of structure, and presents an algorithm for automatically finding good structures by stochastic optimization. Our algorithm begins with a simple model and then performs hill-climbing in the space of possible structures by splitting states and gauging performance on a validation set. Experimental results show that this technique finds HMM models that almost always out-perform a fixed model, and have superior average performance across tasks.

Introduction

The Internet makes available a tremendous amount of text that has been generated for human consumption; unfortunately, this information is not easily manipulated or analyzed by computers. *Information extraction* (IE) is the process of filling fields in a database by automatically extracting fragments of human-readable text. Examples include extracting the location of a meeting from an email message, or the name of a corporate takeover target.

Recent research has demonstrated the effectiveness of hidden Markov models (HMMs) for information extraction. HMMs have been applied successfully to many sub-domains of information extraction: the named entity extraction task (Bikel *et al.* 1997); to the task of recovering the sequence of a set of entities occurring in close proximity (*dense extraction*) (Seymore *et al.* 1999); as well as the *sparse extraction* task, in which the object is to extract relevant phrases from documents containing much irrelevant text (Leek 1997; Freitag and McCallum 1999). In many cases, the accuracy of HMMs applied to these tasks is state-of-the-art and often significantly better than alternative learning approaches.

We address the sparse extraction task. We assume that for every document in a corpus there is a corresponding relational record (template), each slot of which is either empty or is filled with a fragment of text from the document. For example, an electronic seminar announcement might contain the title of the talk, the name of the speaker, the starting time, etc. These typed relevant fragments are called *fields*.

For each field we train a separate HMM. In performing extraction on a particular file, the model must account for all tokens in the document. Special states (called "target states") are trained to emit only those tokens that are part of the phrases to be extracted. Other states ("background states") are designated to emit non-target tokens. By varying the number and connection among states, we can design models that account for a wide range of text patterns, including patterns of language in the neighborhood of target phrases and the structure of the phrases themselves.

A significant problem when applying HMMs to information extraction is the selection of state-transition structure. Certain structures better capture the observed phenomena in the prefix, target and suffix sequences around certain targets. For example, if we know that we want to extract names, we might set aside a single target state for honorifics, one for first names, and one for last names, and connect these states in a way that matches our intuitions. Unfortunately the approach of building structures by hand does not scale to large corpora and is difficult to follow in practice. Furthermore, human intuitions do not always correspond to structures that make the best use of HMM potential.

This paper shows that the selection of state-transition structure effects tremendously the accuracy of the HMM extractor, and presents a stochastic-optimization approach for learning good task-specific structures automatically. Our method begins with a minimal number of states, explores various state splitting operations, selects the operation that gives best performance on a labeled validation set, and recursively explores further splitting operations. The final model is then chosen by cross-validation from those generated.

The idea of automatic structure selection for HMMs is not new (Stolcke and Omohundro 1994; Carrasco and Oncina 1994; Vasko *et al.* 1997; Lockwood and Blanchet 1993), and it has been applied to the problem of dense extraction (Seymore *et al.* 1999). Unlike this and much of the other work—which typically uses goodness of statistical fit to the data—our structure selection process at each step dicriminatively optimizes performance on the task at hand. It also shows much greater improvements due to structure learning. Whether this improvement is possible only in the context of sparse extraction is a question for future research.

We present experimental results on four different data sets. The learned structures give higher accuracy than previously attained using hand-built models, and also outperform SRV and Rapier, two state-of-the-art information extraction systems that employ ILP methods (Freitag 1998; Califf 1998).

HMMs for Information Extraction

Like many tasks involving discrete sequences, good performance on information extraction (IE) tasks relies on powerful modeling of context as well as the current observations. Finite state machines, such as hidden Markov models, offer a good balance between simplicity and expressiveness of context.

Hidden Markov Models

A HMM is a finite state automaton with stochastic state transitions and symbol emissions (Rabiner 1989). The automaton models a probabilistic generative processes whereby a sequence of symbols is produced by starting in some state, transitioning to a new state, emitting a symbol selected by that state, transitioning again, emitting another symbol—and so on until a designated final state is reached. Associated with each of a set of states, $S = \{s_1, \cdots, s_n\}$, is a probability distribution over the symbols in the emission vocabulary $V = \{w_1, w_2, \cdots w_K\}$. The probability that state s_j will emit the vocabulary item w is written $\mathrm{P}(w|s_j)$. Similarly, associated with each state is a distribution over its outgoing transitions. The probability of moving from state s_i to state s_j is written $\mathrm{P}(s_j|s_i)$. There is also a prior state distribution $P_0(s)$. Training data consists of several sequences of observed emissions, one of which would be written $\{o_1, o_2, \cdots o_m\}$.

Information Extraction with HMMs

Given a model and all its parameters, IE is performed by determining the sequence of states that was most likely to have generated the entire document, and extracting the symbols that were associated with certain designated "target" states. Determining this sequence is efficiently performed by dynamic programming with the *Viterbi* algorithm (Rabiner 1989).

The models we use for IE have the following characteristics:

- Each HMM extracts just one type of field (such as "seminar speaker"). When multiple fields are to be extracted from the same document (such as "seminar speaker" and "seminar location"), a separate HMM is constructed for each field.
- They model the entire document, and thus do not require pre-processing to segment document into sentences or other pieces. The entire text of each training document is used to train transition and emission probabilities.
- They contain two kinds of states, target states and non-target states. Target states are intended to produce the tokens we want to extract.
- They are not fully connected. The restricted transition structure captures context that helps improve extraction accuracy.

For IE, in addition to the traditional HMM parameters and training data, we have labels indicating which are the "target" states and observations. Let $L(s)$ be a binary value indicating whether state s is among the target states. Each training instance is also labeled to indicate which observations are among the target observations for this task, represented by a sequence of binary labels for each observation sequence, written $\{l_1, l_2, \cdots l_m\}$.

Parameter Estimation

Once the state-transition structure is determined, the remaining parameters of the model are the transition and emission probabilities. For IE both are estimated using labeled training data—that is, sequences of words with the target words already identified.

In some HMM structures, the labels determine a unique path through the states. If a unique path does not exist, then we use EM (in the form of Baum-Welch) to iteratively estimate parameters and fill in the missing path. In the E-step we estimate the expected path exactly as in Rabiner 1989, except that we also obey the target label constraints. Hence, for example, the iteration step of the forward procedure becomes:

$$\alpha_{t+1}(s) = \begin{cases} 0 & \text{if } l_{t+1} \neq L(s) \\ \sum_{s'} \alpha_t(s)\mathrm{P}(s|s')\mathrm{P}(o_{t+1}|s) & \text{otherwise} \end{cases} \tag{1}$$

and the backward procedure is modified analogously.

Transition probabilities are low-degree multinomials, which we estimate by maximum likelihood with ratios of counts, as is traditional. Emission probabilities on the other hand are very high-degree multinomials and require smoothing with a prior because training data is extremely sparse relative to the number of parameters.

Rather than smoothing simply against the uniform distribution, the results in this paper build on work in using shrinkage with HMMs for information extraction (Freitag and McCallum 1999). In many machine learning tasks there is a tension between constructing complex models with many states and constructing simple models with only a few states. The complex model is able to represent intricate structure of the task, but often results in poor (high variance) parameter estimation because the training data is highly fragmented. The simple model results in robust parameter estimates, but performs poorly because it is not sufficiently expressive to model the data (too much bias).

Shrinkage (a general term that includes "hierarchical Bayes" or "empirical Bayes") is a family of statistical techniques that balance these competing concerns. In our HMMs shrinkage is used to "shrink" parameter estimates from data-sparse states of the complex model toward the estimates in related data-rich states of the simpler models. The combination of the estimates is provably optimal under the appropriate conditions. We employ a form of shrinkage that combines the estimates with a weighted average, and

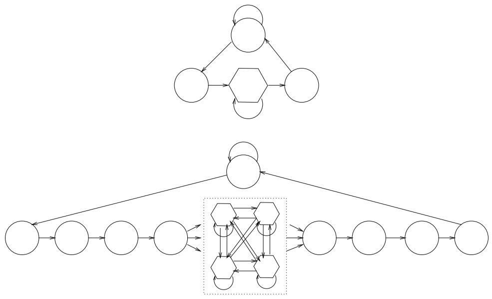

Figure 1: Two example HMM structures. Circle nodes represent non-target states; hexagon nodes represent target states.

learns the weights with Expectation-Maximization. Thus, the smoothed, shrinkage-based emission probability of word w being emitted by state s is

$$\lambda_1 \mathrm{P}(w|s) + \lambda_2 \mathrm{P}(w|a(s)) + \lambda_3 (1/K) \tag{2}$$

where the last term represents the uniform distribution, $\lambda_1 + \lambda_2 + \lambda_3 = 1$, and $a(s)$ is the "parent" of state s in the shrinkage hierachy, (*i.e.* the data-rich abstract state in the simpler model). In our implementation all target states share a parent, and all non-target state share another parent. Space limitations prevent a full description of our shrinkage implementation here; see Freitag and McCallum (Freitag and McCallum 1999) for the details.

Learning State-Transition Structure by Stochastic Optimization

The class of structures we consider for information extraction reflects our intuition that successful extraction requires a learner to model both the typical contents of a field and its context to either side. We distinguish four types of states:

- **Target** States required to model the content of target phrases.
- **Prefix** A prefix is a set of one or more states connected as a string. A prefix state transitions only to the next state in the string or, if it is the last state in the string, to one or more target states. Models are designed in such a way that, if a state sequence (such as that returned by Viterbi) passes through any target state, it must first pass through a prefix.
- **Suffix** A suffix is similar in structure to a prefix. Any state sequence must pass through a suffix upon leaving the set of target states.
- **Background** Background states model any text not modeled by other kinds of states. A background state has outgoing transitions only to itself and to the beginnings of all prefixes, and it has incoming transitions only from itself and the ends of all suffixes.

Figure 1 shows two HMM structures that meet our criteria. The bottom model has a single background state, a prefix and suffix of length four, and four fully interconnected target states. This model performs quite well on a range of information extraction tasks. The top model is the simplest we consider and serves as the starting point for our method's search in structure space.

Beginning with the simple model in Figure 1 we perform hill-climbing in the space of possible structures, at each step applying each of a set of operations to the current model and selecting one of the resulting structures as the next model. For the experiments reported here we define seven operations:

- **Lengthen a prefix** A single state is added to the end of a prefix. The penultimate state now transitions only to the new state; the new state transitions to any target states to which the penultimate state previously transitioned.
- **Split a prefix** A duplicate is made of some prefix. Transitions are duplicated so that the first and last states of the new prefix have the same connectivity to the rest of the network as the old prefix.
- **Lengthen a suffix** The dual of the prefix-lengthening operation.
- **Split a suffix** Identical to the prefix-splitting operation, except applied to a suffix.
- **Lengthen a target string** Similar to the prefix-lengthening operation, except that all target states, in contrast with prefix and suffix states, have self-transitions. The single target state in the simple model in Figure 1 is a target string of length one.
- **Split a target string** Identical to the prefix-splitting operation, except applied to a target string.
- **Add a background state** Add a new background state to the model, with the same connectivity, with respect to the non-background states, as all other background states: the new state has outgoing transitions only to prefix states and incoming transitions only from suffix states.

Note that some of these operations may be applied in several ways, resulting in distinct structures, depending on the model. The result of applying any operation is the set of all topologically distinct models it can generate.

In our experiments, for efficiency, all structures have the same shrinkage configuration, as described in the previous section. Note that the shrinkage configuration can also be determined through optimization, in at least two ways: (1) The two states created in a splitting operation might share a local shrinkage distribution, as well as the distributions created as part of previous splits. The resulting hierarchical configuration would thereby reflect the sequence of operations that led to its construction. (2) We might include an additional set of shrinkage modification operators independently in the stochastic optimization.

Table 1 presents our method for selecting structure. It consists of two loops: one in which a set of structures is generated using one-step look-ahead hill-climbing and F1 performance on a hold-out set;[1] and one in which the models from this set are re-scored using 3-fold cross-validation

[1] See the next section for a description of the F1 metric.

```
procedure LearnStructure(LabeledSet, Ops)
    ValidSet ← 1/3 of LabeledSet
    TrainSet ← LabeledSet − ValidSet
    CurModel ← the simple model
    Keepers ← {CurModel}
    I ← 0
    while I < 20 and CurModel has fewer than 25 states
        Candidates ← {M|M ∈ op(CurModel) ∧ op ∈ Ops}
        for M ∈ Candidates
            score(M) ← average of 3 runs trained on
                TrainSet and scored for F1 on ValidSet
        CurModel ← M ∈ Candidates with highest score
        Keepers ← Keepers ∪ {CurModel}
        I ← I + 1
    for M ∈ Keepers
        score(M) ← average F1 from
            3-fold cross-validation on LabeledSet
    return M ∈ Keepers with highest score
```

Table 1: The optimization procedure used to select HMM structure.

on the training set. At any given step in the first loop, LearnStructure selects as the next model the single candidate that scores the best average F1 from several training/testing runs. We average several runs in this way, seeding the model in a different way each time, because Baum-Welch settles into different local optima depending on the initial parameter settings. The model returned by LearnStructure is the one from the series of generated structures that scores the best F1 in separate runs of three-fold cross-validation on the training set.

Experimental Results

We tested our approach on eight information extraction tasks defined over 4 document collections: **(1)** *SA*: A collection of 485 seminar announcements posted electronically at a large university. Fields include *speaker*, the name of the speaker at the seminar, and *location* the location (e.g., room number) of the seminar. **(2)** *Acq*: A collection of 600 Reuters articles detailing coporate acquisitions. Fields include *acquired*, the name of the company to be purchased, and *dlramt*, the purchase or estimated price of the sale. **(3)** *Jobs*: A collection of 298 Usenet job announcements. Fields include *company*, the name of the company seeking to hire, and *title*, the job title.) **(4)** *CFP*: A collection of 363 Internet "Call for Paper" announcements in ASCII format. Fields include *conf*, the name of the conference, and *deadline*, the full-paper submission deadline. Except for the *CFP* collection, all of these corpora have been used in previously published research.

For each of our experiments we adopt the same basic procedure: The document collection is partitioned several times into a training set and a testing set. We train a learner using the training set and measure its performance using the testing set. In the case of the approach described in this paper, we use the training set both to select a model structure and to set its parameters. With the exception of the *Jobs* partitions, the training and testing sets are of roughly equal size. The *Jobs* partitions—exactly those used in Califf (Califf 1998)—contain 90% training and 10% testing. Results from experiments we ran represent average performance over three training/testing splits. Rapier's scores are those reported in Califf (Califf 1998); they represent average performance over ten training/testing splits.

In the extraction problems we consider here, there is a single correct filler (which may occur several times in a document, with slight variations) for each slot in the answer template. Given a test document, a learner must identify a target fragment or, if none is present, decline to perform an extraction. In order for an extraction to be counted as correct, the precise boundaries of a target fragment must be identified. If a learner issues multiple predictions for a document, we take only the one with highest confidence. Two metrics characterize the performance of a learner: *precision*, the number of correct extractions divided by the number of documents for which the learner issued any prediction; and *recall*, the number of correct extractions divided by the number of documents containing one or more target fragments. We report *F1*, the harmonic mean of precision and recall.

We compare structure learning with four other approaches, two rule-learning approaches previously reported in the literature—SRV (Freitag 1998) and Rapier (Califf 1998)—and two static HMM models. Both SRV and Rapier are relational rule learners that have been shown to perform well on a variety of tasks. SRV induces rules "top down," beginning with a most general rule and specializing. Rapier induces rules bottom up, successively generalizing to cover target phrases. Rows labeled "Simple HMM" and "Complex HMM" show the performance of the two static models shown in Figure 1. Note that the "Simple HMM" is the model with which structure selection begins.

Table 2 shows the F1 performance of the HMM with learned structure on eight tasks and, for each of the four competing approaches, lists the difference in F1 score between the "Grown HMM" and the respective approach. It is clear from these results that, on balance, HMMs are to be preferred over the rule learners mentioned here. They achieve superior performance on almost all tasks, sometimes by substantial margins. Even the "Simple HMM" occasionally out-performs the symbolic methods.

Of course, in order for an HMM to realize its potential, some structure selection is required, as the "vs. Simple HMM" row in Table 2 indicates. The average performance difference with the simple model constitutes an improvement of 24%. A well-designed static model (row "Complex HMM" in Table 2) can achieve good performance, but on average its performance lags behind a dynamically selected model. Note that the complex model was selected based on considerable manual interaction with the *SA* domain, particularly the *speaker* task. It is in some sense optimized for this task, so it comes as no surprise that this is one of the few tasks on which structure learning yields worse results than the static model.

Figure 2 shows parts of the structures of HMMs designed to extract seminar locations and speakers, respectively. Transitions are labeled with the probability assigned to them by Baum-Welch; only transitions with probability greather than 0.1 are shown. Each node in the *location* graph

	speaker	*location*	*acquired*	*dlramt*	*title*	*company*	*conf*	*deadline*	***Average***
Grown HMM	76.9	87.5	41.3	54.4	58.3	65.4	27.2	46.5	57.2
vs. SRV	+19.8	+16.0	+1.1	-1.6	—	—	—	—	+8.8
vs. Rapier	+23.9	+14.8	+12.5	+15.1	-11.7	+24.9	—	—	+13.3
vs. Simple HMM	+24.3	+5.6	+14.3	+5.6	+5.7	+11.1	+15.7	+6.7	+11.1
vs. Complex HMM	-2.1	+6.7	+7.5	-0.3	-0.3	+19.1	+0.0	-6.8	+3.0

Table 2: Difference in F1 performance between the HMM using a learned structure and other methods. The + numbers indicate how much better our Grown HMM did than the alternative method.

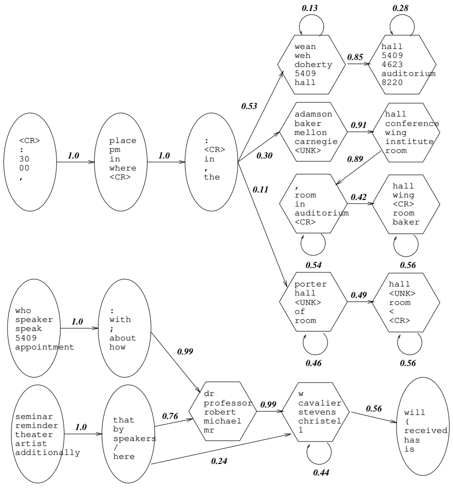

Figure 2: Part of two learned structures designed to extract *locations* (top) and *speakers* (bottom).

displays the top 5 most probable tokens emitted by that state, in order from top to bottom; the nodes in the *speaker* graph show the top 5 tokens according to an odds-ratio metric. The token `<CR>` stands for a carriage return; the token `<UNK>` stands for the "unknown" token, in our experiments any token occurring fewer than three times in the training set. Both of these models were the best found in their respective splits. The *location* model was found after 8 states of state splitting, the *speaker* model after 3 steps.

The figure suggests that, in order to extract seminar locations, a single, relatively short prefix context is needed. The fact that the model retains only a single prefix points to the unambiguity and length invariance of the kinds of language leading up to a location. Phrases like "`Place:`", "`Where:`", "`in the`", and "`00 pm,`" are encoded by the prefix. This last phrase is an indication that locations are often preceded by times.

The target states are partitioned into three parallel paths. The top path, accounting for about half of *location* phrases, captures many common *location* phrases consisting of two or three tokens, phrases such as "`Weh 5409`" and "`Wean Hall 5409`". Wean and Doherty are buildings on the CMU campus and common meeting places for the talks announced in the *SA* corpus. The middle path, of length four, appears dedicated to modeling longer *location* phrases, particularly a very popular meeting place which appears in a variety of formats: "`Adamson Wing Auditorium in Baker Hall`", "`Baker Hall, Adamson Wing`", "`Adamson Wing, Baker Hall`". Finally, the bottom path appears to be a "garbage" path, dedicated to modeling locations that are not easily modeled by the other two paths. The high probability of the unknown token in these two states supports this interpretation.

The first target state in the *speaker* model appears dedicated to emitting honorifics and first names, while the other target state emits middle initials and last names. The use of two prefixes suggests that seminar speakers occur in two contexts. The top prefix seems dedicated to accounting for the initial formal presentation, in which phrases like "`Who:`" and "`Speaker:`" are common, followed by use of the speaker's full name, including honorific. The bottom prefix seems to be used in the body of the announcement in less formal contexts. It accounts for phrases like "`reminder that`" and "`seminar by`". Interestingly, a significant fraction of contexts in which this prefix is used skip to the second target state, apparently because an honorific or first name has been omitted.

Related Work

HMMs have been applied to various versions of the information extraction problem in recent years. The approach described in Freitag and McCallum (Freitag and McCallum 1999) addresses the same problem as in this work—training one HMM per extraction task—but involves manually constructed models. Seymore *et al.* (Seymore *et al.* 1999) describe experiments in structure learning, but use HMMs that model all fields simultaneously, and address problems in which an ordering of fields is sought, rather than the location of a single field in a large body of background text. Bikel *et al.* (Bikel *et al.* 1997) applies HMMs to the named entity recognition problem, the problem of identifying text fragments that signify particular types of entities, such as people or organizations, without regard to their role in the document. They describe manually designed HMMs with one state per type of entity and use n-gram statistics, rather than HMM structure, to exploit context. The HMMs in Leek (Leek 1997) are carefully designed—both state-transition structure and emission distributions—to model the syntactic constraints of the particular extraction problem.

The problem of learning HMM structure for tasks other than information extraction has seen a fair amount of work. Stolcke and Omohundro (Stolcke and Omohundro 1994) propose a state-merging approach which begins with a large, maximally specific topology and iteratively merges pairs of states. The merging criterion is not performance on any particular task, as in this work, but a Bayesian combination of prior expectations regarding suitable topologies and goodness of fit to the data. Seymore *et al.* (Seymore *et al.* 1999) apply the same approach to their extraction problem. Closely related are the state merging algorithms that have been investigated for some years in the field of grammatical inference, particularly those involving stochastic regular grammars (Carrasco and Oncina 1994). State splitting appears better suited than state merging to the sparse extraction problem. Much of the work in state merging presupposes problems that resemble formal language modeling. The problem of dense extraction is much closer in character to formal language identification than is sparse extraction.

Alternatives to state merging/splitting exist. Vasko *et al.* (Vasko *et al.* 1997) describe a method which begins with a fully-connected structure and iteratively deletes transitions. Lockwood and Blanchet (Lockwood and Blanchet 1993) propose a method that applies incremental patches to a circuit-free model for speech processing.

Conclusions

Previous work has shown that hidden Markov models are the state-of-the-art method for information extraction. This paper has shown that task-specific state-transition structure of these models is tremendously important to their performance, and has further pushed the state-of-the-art by showing that discriminative stochastic optimization can automatically discover good structures. We hope that these initial investigations will lead to improved methods in the future.

References

Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. In *Proceedings of ANLP-97*, pages 194–201, 1997.

Mary Elaine Califf. *Relational Learning Techniques for Natural Language Information Extraction*. PhD thesis, University of Texas at Austin, August 1998.

Rafael C. Carrasco and Jose Oncina. Learning stochastic regular grammars by means of a state merging method. In Rafael C. Carrasco and Jose Oncina, editors, *Grammatical Inference and Applications: Second International Colloquium, ICGI-94*. Springer-Verlag, September 1994.

Dayne Freitag and Andrew Kachites McCallum. Information extraction using hmms and shrinkage. In *Papers from the AAAI-99 Workshop on Machine Learning for Information Extration*, pages 31–36, July 1999. AAAI Technical Report WS-99-11.

Dayne Freitag. Information extraction from HTML: Application of a general machine learning approach. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.

Timothy R. Leek. Information extraction using hidden Markov models. Master's thesis, UC San Diego, 1997.

Philip Lockwood and Marc Blanchet. An algorithm for the dynamic inference of hidden Markov models (DIHMM). In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-93)*, 1993.

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

Kristie Seymore, Andrew McCallum, and Ronald Rosenfeld. Learning hidden Markov model structure for information extraction. In *Papers from the AAAI-99 Workshop on Machine Learning for Information Extration*, pages 37–42, July 1999. AAAI Technical Report WS-99-11.

Andreas Stolcke and Stephen M. Omohundro. Best-first model merging for hidden Markov induction. Technical Report TR-94-003, International Computer Science Institute, Berkeley, California, January 1994.

Raymond C. Vasko, Jr., Amro El-Jaroudi, J.R. Boston, and Thomas E. Rudy. Hidden Markov model topology estimation to characterize the dynamic structure of repetitive lifting data. In *Proceedings of the 19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 1997.