Information Extraction with HMMs and Shrinkage

Dayne Freitag
dayne@justresearch.com

Andrew Kachites McCallum
mccallum@justresearch.com

Just Research
4616 Henry Street
Pittsburgh, PA 15213

Abstract

Hidden Markov models (HMMs) are a powerful probabilistic tool for modeling time series data, and have been applied with success to many language-related tasks such as part of speech tagging, speech recognition, text segmentation and topic detection. This paper describes the application of HMMs to another language related task—information extraction—the problem of locating textual sub-segments that answer a particular information need. In our work, the HMM state transition probabilities and word emission probabilities are learned from labeled training data. As in many machine learning problems, however, the lack of sufficient labeled training data hinders the reliability of the model. The key contribution of this paper is the use of a statistical technique called "shrinkage" that significantly improves parameter estimation of the HMM emission probabilities in the face of sparse training data. In experiments on seminar announcements and Reuters acquisitions articles, shrinkage is shown to reduce error by up to 40%, and the resulting HMM outperforms a state-of-the-art rule-learning system.

Introduction

The Internet makes available a tremendous amount of text that has been generated for human consumption; unfortunately, this vast quantity of information is not easily manipulated or analyzed by computers. *Information extraction* is the process of filling fields in a database by automatically extracting sub-sequences of human-readable text. Examples include extracting the location of a meeting from an email message, or extracting the name of the acquired company in a newswire article about a company takeover.

This paper advocates the use hidden Markov models (HMMs) for information extraction. HMMs are a type of probabilistic finite state machine, and a well-developed probabilistic tool for modeling sequences of observations. They have been applied with significant success to many language-related tasks, including as part-of-speech tagging (Kupiec 1992), speech recognition (Rabiner 1989), text segmentation and topic detection (van Mulbregt *et al.* 1998).

Because HMMs have foundations in statistical theory, there is a rich body of established techniques for learning the parameters of an HMM from training data and for classifying test data. In our work HMM state-transition probabilities and word emission probabilities are learned from labeled training data. However, as is the case in many machine learning problems, large amounts of training data are required to learn a model that generalizes well and has high accuracy. Since training data must usually be painstakingly labeled by hand, it is often difficult to obtain enough, and the small quantities of available training data is a limiting factor on performance of the learned extractor.

The key contribution of this paper is the integration of a statistical technique called *shrinkage* into information extraction by HMMs. In our system, shrinkage is used to learn more robust HMM emission probabilities in the face of limited training data. The technique works by "shrinking" parameter estimates in data-sparse individual states towards the estimates calculated for data-rich conglomerations of states, and does so in ways that are provably optimal under the appropriate conditions. Shrinkage has been widely used in statistics and language modeling, including in HMMs for acoustic modeling in speech recognition (Lee 1989).

In our approach to information extraction, the HMM forms a probabilistic generative model of an entire document from which sub-segments are to be extracted. Separate HMMs are built to extract different fields from a document. In each HMM, a subset of the states are distinguished as "target" states, and any words of the document that are determined to have been generated by those states are part of the extracted sub-sequence.

Several recent reports have described IE systems that break documents into many small fragments for the purposes of learning and prediction. Such approaches require sometimes unsatisfactory heuristic choices and can result in a huge space of possible fragments. Our approach effectively solves this fragment enumeration problem by means of the Viterbi algorithm, an efficient method for finding the most probable sequence of model states corresponding to a given document.

A few previous projects have used HMMs for information extraction, although with differing structures, and none with shrinkage over HMM states (Leek 1997; Bikel *et al.* 1997). A related project focuses on the quite different task of learning HMM state-transition structure for information extraction (Seymore, McCallum, & Rosenfeld 1999).

We describe experiments on two real-world data sets: on-line seminar announcements and Reuters newswire articles on company acquisitions. Results show that shrinkage consistently improves the performance over absolute discounting. A representative HMM outperforms a state-of-the-art rule-learning system on seven of nine extraction tasks.

HMMs for Information Extraction

Suppose we are given the task of extracting the purchasing price from each document in a collection of articles describing corporate acquisitions. We can imagine that a given article is the result of a stochastic process involving two unigram language models: a background model that typically emits tokens like 'that' and 'said'; and a model specific to prices that typically emits tokens like '$' and 'million'. To generate a document, the process emits tokens from the background model, at some point switching to the price model for a few tokens, then returning to the background model to complete the document. Slightly more realistically, there might be a number of specialized models from which the process draws, some of them devoted to the context of the purchasing price, models responsible for prefix fragments like "purchased XYZ Corp for".

HMMs represent precisely this kind of process. 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 at a designated start 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 $V = \{w_1, w_2, ...w_n\}$. 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 set of outgoing transitions. The probability of moving from state s_i to state s_j is written $\mathrm{P}(s_j|s_i)$.

Model transition and emission probabilities can be learned from training data. When the training sequences are sufficiently labeled so as to be associated with a unique path of states, the probabilities can be calculated straightforwardly with ratios of counts (maximum likelihood) or smoothed ratios of counts (maximum a posteriori).

Given a model and all its parameters, information extraction 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 designated "target" states.

To perform extraction we therefore require an algorithm for finding the most likely state sequence given a HMM model M and a sequence of symbols. Although

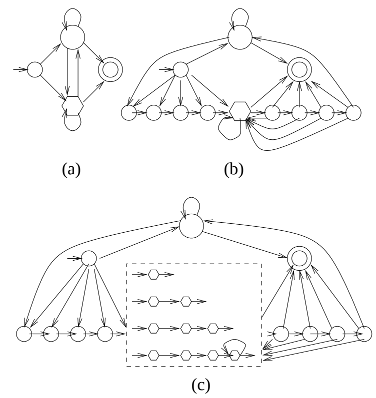

Figure 1: Three example topologies.

a naive approach to finding the most likely sequence would take time exponential in the sequence length, a dynamic programming solution called the *Viterbi algorithm* solves the problem in just $O(TN^2)$ time, where T is the length of the sequence and N is the number of states in M. Thus the Viterbi algorithm executes with time linear in the length of the sequence—much more efficiently than several other information extraction approaches that evaluate a super-linear number of sub-sequences.

The models we use for information extraction have the following characteristics:

- Each HMM extracts just one type of field (such as "purchasing price"). When multiple fields are to be extracted from the same document (such as "purchasing price" and "acquiring company"), 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, background states and 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. State-transition topology is set by hand, not learned from training data.

Figure 1 shows three example topologies. The model in Figure 1(a) is the simplest possible topology. In practice, of course, we expect that context around the target state to provide important clues in the search for target text. We can exploit some of these clues by adding prefix and suffix states, as in Figure 1(b). Similarly, target fragments can vary in length, and certain tokens may be more common at the beginning or end of the fragments. If the object is to extract the name of a company for example, the tokens "Inc" and "Corp" are almost certainly at the end of a fragment. We can attempt to capture such structure by expanding the single target state into an array of parallel paths of varying length. The final state in the longest such path includes a self-transition, in order to account for unusually long target fragments. Figure 1(c) shows a set of target paths of lengths one to three.

Figure 1 illustrates the two topological parameters we experiment with, context window size and target path count. The window size is the number of prefix and suffix states. The model in Figure 1(b) has a window size of four. Of course, nothing requires that a model have the same number of prefix and suffix states; we assume this for simplicity. The path count is the number of parallel, length-differentiated, sequential target paths. If the path count is P, then a model has P target paths, varying in length from one to P. The model in Figure 1(b) has a path count of one; the model in Figure 1(c) has a path count of three.

In order to train a model, each token in a document is labeled according to whether it is part of the target text. We require that only target states emit such tokens, and only non-target states emit non-target tokens. Given this constraint, and a particular non-empty document, only a single, unambiguous path is possible through any of the topologies in Figure 1.

Estimating Emission Probabilities

When the emission vocabulary is large with respect to the number of training examples, maximum likelihood estimation of emission probabilities will lead to poor estimates, with many words inappropriately having zero probability. The use of a well-chosen prior in conjunction with maximum a posteriori or Bayes optimal estimation will prevent zero-probability estimates and improve estimation overall.

For example, Bayes optimal parameter estimation in conjunction with a uniform Dirichlet prior results in the widely used *Laplace smoothing*, in which the count of every word in the vocabulary in incremented by a single extra "priming" occurrence. This is also known as *additive smoothing*. An alternative smoothing technique that performs better when the number of zero-count words varies widely from state to state is *absolute discounting*. This method subtracts a fixed discount, $0 < d < 1$ from all words with count greater than zero. The resulting available probability mass is then distributed over all words that have zero count according to some prior distribution (in our case uniform). Thus we have

$$\mathrm{P}(w|s) = \begin{cases} \frac{N(w,s)-d}{N(s)}, \text{if } N(w,s) > 0 \\ \frac{d \cdot (|V| - |Z_s|)}{|Z_s|}, \text{if } N(w,s) = 0. \end{cases} \quad (1)$$

where $|V|$ is the size of the vocabulary, $N(w,s)$ is the number of times w occurs in the training data for state s, $N(s)$ is the total number of word occurrences across all words in training data for state s, and $|Z_s|$ is the number of unique words with zero count in state s. There is no closed-form solution for the optimal value of d. A typical choice is $z(s,1)/(z(s,1) + 2 \cdot z(s,2))$, where $z(s,n)$ is the number of unique words in state s with count n.

Both *Laplace smoothing* and *absolute discounting* calculate the word distribution in a state using only the training data in state s itself. In the next section, we discuss *shrinkage*, a method that leverages the word distributions in several related states in order to improve parameter estimation.

Shrinkage

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 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 is a statistical technique that balances these competing concerns by "shrinking" 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. Shrinkage has been extensively studied in statistics (Carlin & Louis 1996). Versions of this technique are also used widely in statistical language modeling for speech recognition. We employ a simple form of shrinkage that combines the estimates with a weighted average, and learns the weights with Expectation Maximization. In speech recognition this form is called *deleted interpolation* (Jelinek & Mercer 1980).

Shrinkage for HMMs and Information Extraction

Shrinkage is typically defined in terms of some hierarchy that represents the expected similarity between parameter estimates, with the estimates at the leaves. To create a hierarchy from an HMM, we define subsets of states that have word emission distributions we expect to be similar, and declare them to share a common "parent" in a hierarchy of word distributions.

Figure 2 shows such a hierarchy. It depicts, for example, that all prefix states are expected to have related word distributions—reflecting also the fact that in a simpler model, all four prefix states might have been represented by a single state that allowed up to four self-transitions. Internal nodes of the hierarchy can also have parents, reflecting expectations about weaker similarity between groups of states, and representing HMM emission distributions that are yet again more simple. At the top of each hierarchy is the most unassuming of all word distributions, the uniform distribution, which gives all words in the vocabulary equal probability. Because the uniform distribution is included we no longer

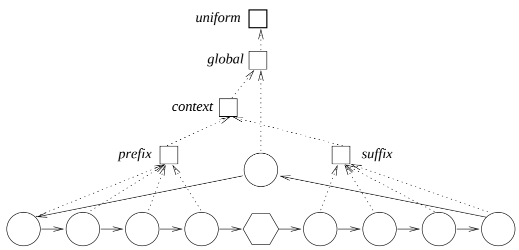

Figure 2: A shrinkage configuration that addresses data sparsity in contextual states, showing shrinkage only for *non-target* states.

need to smooth the local estimates with Laplace or absolute discounting.

We have compared several shrinkage hierarchy configurations. Given that we distinguish four classes of states: *non-target*, *target*, *prefix*, and *suffix*, the four shrinkage configurations are described as follows:

- **None.** No shrinkage; only absolute discounting is used.
- **Uniform.** Instead of absolute discounting, all single-state distributions are shrunk toward the uniform distribution.
- **Global.** The distributions of all *target* states are shrunk toward a common parent, as well as the uniform distribution; likewise for the *non-target* states with a different parent.
- **Hierarchical.** (Shown in Figure 2.) *Target* distributions are handled in the same way as in *global*. Each of the other classes of states—*non-target*, *prefix*, and *suffix*—is shrunk toward a separate, class-specific parent. The prefix and suffix parents are furthermore shrunk toward a shared "context" grandparent. Finally, all *non-target*, *prefix*, and *suffix* states are also shrunk toward a single ancestor, shared among all states that are not *target* states. Again, every state is also shrunk toward the uniform distribution.

Shrinkage-Based Word Probabilities

Our new, shrinkage-based parameter estimate in a leaf of the hierarchy (state of the HMM) is a linear interpolation of the estimates in all distributions from the leaf to its root. Local estimates are calculated from their training data by maximum likelihood (simple ratios of counts, with no additions or discounting). The training data for an internal node of the hierarchy is the union of all the data in its children.

We write the word probability estimates for the nodes on the path starting at state s_j as $\{\mathrm{P}(w|s_j^0), \mathrm{P}(w|s_j^1), \ldots \mathrm{P}(w|s_j^k)\}$, where $\mathrm{P}(w|s_j^0)$ is the estimate at the leaf, and $\mathrm{P}(w|s_j^k)$ is the uniform distribution at the root. The interpolation weights among these estimates are written $\{\lambda_j^0, \lambda_j^1, \ldots \lambda_j^k\}$, where $\sum_{i=1}^k \lambda_j^i =$ 1. The new, shrinkage-based estimate for the probability of word w in state s_j is written $\hat{\mathrm{P}}(w|s_j)$, and is the weighted average:

$$\hat{\mathrm{P}}(w|s_j) = \sum_{i=1}^{k} \lambda_j^i \mathrm{P}(w|s_j^i). \tag{2}$$

Determining Mixture Weights

We derive empirically optimal weights, λ_j^i, between the ancestors of state s_j, by finding the weights that maximize the likelihood of some hitherto unseen "held-out" data, $\mathcal{H}$. This maximum can be found by a simple form of Expectation-Maximization (EM) (Dempster, Laird, & Rubin 1977), where each each word is assumed to have been generated by first choosing one of the hierarchy nodes in the path to the root, say s_j^i (with probability λ_j^i), then using that node's word distribution to generate that word. EM then maximizes the total likelihood when the choices of nodes made for the various words are unknown. EM begins by initializing the λ_j's to some initial values, say $\lambda_j^i = \frac{1}{k}$, then iterating the following two steps until the λ_j's do not change:

E-step Calculate the degree to which each node predicts the words in state s_j's held-out set, $\mathcal{H}_j$:

$$\beta_j^i = \sum_{w_t \in \mathcal{H}_j} \frac{\lambda_j^i \mathrm{P}(w_t|s_j^i}{\sum_m \lambda_j^m \mathrm{P}(w_t|s_j^m} \tag{3}$$

M-step Derive new (and guaranteed improved) weights by normalizing the β's:

$$\lambda_j^i = \frac{\beta_j^i}{\sum_m \beta_j^m} \tag{4}$$

While correct and conceptually simple, this method makes inefficient use of the available training data by carving off a held-out set. We fix this problem by evaluating the E-step with each individual word occurrence held out in turn. This method is very similar to the "leave-one-out" cross-validation commonly used in statistical estimation.

Experiments

We present experimental results on nine information extraction problems from two corpora: a collection of seminar announcements posted to local newsgroups at a large university, and a collection of articles describing corporate acquisitions taken from the Reuters dataset (Lewis 1992). Both of these datasets, as well as the IE problems defined for them, are described in detail in previously published work (Freitag 1999).

The performance of an algorithm is measured document by document. If the task is to extract the start time of a seminar from an announcement, we assume that there is a single correct answer (perhaps presented several different times in the same or superficially different ways). We ask whether a learner's single best

	speaker	*location*	*stime*	*etime*
Window = 1	0.431	0.797	0.943	0.771
Window = 4	0.460	0.653	0.960	0.716
Window = 10	0.363	0.558	0.967	0.746

Table 1: Effect on F1 performance of changing "window" size (number of context states) on four seminar announcement fields under absolute discounting.

	speaker	*location*	*stime*	*etime*
None	0.460	0.653	0.960	0.716
Uniform	0.499	0.660	0.971	0.840
Global	0.558	0.758	0.984	0.589
Hier.	0.531	0.695	0.976	0.565

Table 2: Effect on F1 performance of different shrinkage configurations on four seminar announcement fields, given a topology with a window size of four and a single target state.

Distance	*speaker*	*location*	*stime*	*etime*
1	0.84	0.84	0.92	0.95
2	0.81	0.90	0.98	0.98
3	0.73	0.80	0.85	0.95
4	0.65	0.74	0.86	0.93

Table 3: Local mixture weights along the prefix path as a function of distance from the target states.

	speaker	*location*	*stime*	*etime*
None	0.513	0.735	0.991	0.814
Uniform	0.614	0.776	0.991	0.933
Global	0.711	0.839	0.991	0.595
Hier.	0.672	0.850	0.987	0.584

Table 4: Effect on F1 performance of different shrinkage configurations on four seminar announcement fields, given a topology with a window size of four and four parallel length-differentiated target paths.

prediction exactly identifies one of the fragments representing the start time. If a learner's best prediction does not align exactly with an actual start time, as identified by the human labeler, it is counted as an error.

Let C be the number of test documents for which a learner's best prediction correctly identifies an instance of the target field. *Precision* (P) is C divided by the number of documents for which the learner issues any prediction. *Recall* (R) is C divided by the number of documents actually containing an instance of the target field. We present results in terms of *F1*, which is the harmonic mean of P and R, i.e. $2/(1/P + 1/R)$.

We assume that context is necessary for correct extraction on most IE tasks, but we do not know, given a particular problem, how much a HMM can exploit. Table 1 explores the effect on F1 performance of window sizes 1, 4, and 10 on the four extraction tasks from the seminar announcement domain when only absolute discounting is used to estimate emission probabilities. Note that between the smallest and largest window size performance actually decreases on three of the four problems, reflecting the fact that the more complex model fractures the training data, making it more difficult to obtain reliable parameter estimates.

By softening the dependence of emission estimates on relative position, shrinkage allows the model to make improved use of larger window sizes. Table 2 shows the effect of shrinkage on performance of a window-size-four topology. For all fields one or more of the shrinkage configurations out-performs absolute discounting. In some cases the improvement is dramatic; for example shrinkage reduces error on the *speaker* field by 40%. "Global" shrinkage is best for three of the four fields. We attribute the large performance differences on the *etime* task to the relative infrequency of this field. Although times in general are prevalent in the seminar announcement corpus, only about half of the documents contain a seminar end time. The decrease in F1 for the bottom two shrinkage configurations is due to a large number of spurious predictions, extractions from documents in which no end time is present. Thus, by making sparsely trained states more willing to emit tokens they haven't seen in training, shrinkage can sometimes have a deleterious effect.

It is interesting to ask how the distribution of mixture weights varies as a function of a state's role in the model. Table 3 shows, for sample runs on each of the four seminar announcement tasks, how much weight is placed on the local token distribution of each of four prefix states. The "global" shrinkage configuration is used in this case. Note how the local weight tends to decline with increasing distance from the target text, agreeing with our intuition that the most consistent patterns are the closest. Also, local weight decreases in proportion to the difficulty of the field, as reflected in F1 results. Clearly, the two time fields tend to occur in very predictable contexts.

The combination of shrinkage with multiple target paths is particularly powerful. Table 4 shows the effect of the various shrinkage configurations on the performance of the network of Table 2 in which the single target state is replaced by four parallel target paths of lengths one, two, three, and four. Compared with the earlier model, we see a universal increase in performance—a 27% increase in performance on the *location* field. On this field, the performance of the HMM is far better than other non-HMM learning methods we have seen.

The combination of shrinkage with appropriately designed topologies yields a learning algorithm that is based on sound statistical principles, runs efficiently, and performs at a level competitive with or better than the best learning algorithms with which we have experimented. Table 5 compares HMM performance on

	speaker	*location*	*stime*	*etime*	
SRV	0.703	0.723	0.988	0.839	
HMM	0.711	0.839	0.991	0.595	
	acquired	*purchaser*	*acqabr*	*dlramt*	*status*
SRV	0.343	0.429	0.351	0.527	0.380
HMM	0.309	0.481	0.401	0.553	0.467

Table 5: F1 of SRV and a representative HMM on nine fields from two domains, the seminar announcements and corporate acquisitions.

nine information extraction problems with that of SRV, a consistently strong rule-learning algorithm described elsewhere (Freitag 1999). The model in question has a context window size of four, four target paths, and uses the global shrinkage configuration. On all but *etime* and *acquired*, the HMM obtains a higher F1 score than SRV. Note, however, that on *etime* the performance of the uniform shrinkage configuration does beat SRV (see Table 4).

Related Work

HMMs are well suited to a range of language processing tasks and have previously been applied to the problem of information extraction in ways that differ from our approach in various ways.

In a related project, Seymore, McCallum, & Rosenfeld present an effort to learn HMMs state/transition structure 1999. Unlike this paper, the approach uses a single HMM to extract many fields which are densely packed in moderately structured text (such as research paper references and headers). Since many fields must be represented in a single HMM, all observed and expected relative orderings of the fields must be captured, and structure learning becomes a challenging and necessary task.

Leek applies HMMs to the problem of extracting gene locations from biomedical texts (Leek 1997). In contrast with the models we study, Leek's models are carefully engineered for the task at hand—both the general topology (which is hierarchical and complex), and the language models of individual states. Leek uses network topology to model natural language syntax and trims training examples for presentation to the model. States are unigram language models, as in our work, but it is unclear what smoothing policy is used. Unknown tokens are handled by special "gap" states.

The Nymble system (Bikel *et al.* 1997) uses HMMs to perform "named entity" extraction as defined by MUC-6. All different fields to be extracted are modeled in a single HMM, but to avoid the resulting difficult structure-learning problem, there is a single state per target and the state-transition structure is completely connected. Emission and transition probabilities are conditioned not only on the state, but also on the last emission—resulting in highly fragmented probability functions, and sparse training data. Thus they use a form of shrinkage to obtain more robust parameter estimates. However, unlike our shrinkage, which averages among different HMM states, they average among distributions that use different representations of the last emission. They also do not learn the optimal mixture weights with EM.

Conclusions

This paper has demonstrated the ability of shrinkage to improve the performance of HMMs for information extraction. The tension between the desire for complex models and the lack of training data is a constant struggle here (as in many machine learning tasks) and shrinkage provides for us a principled method of striking a balance. We have also strived to motivate the high suitability of HMMs to the information extraction task in general—for example, by pointing out that the availability of the Viterbi algorithm avoids the need to evaluate a super-linear number of sub-sequences.

References

Bikel, D. M.; Miller, S.; Schwartz, R.; and Weischedel, R. 1997. Nymble: a high-performance learning name-finder. In *Proceedings of ANLP-97*, 194–201.

Carlin, B., and Louis, T. 1996. *Bayes and Empirical Bayes Methods for Data Analysis.* Chapman and Hall.

Dempster, A.; Laird, N.; and Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* 39(B):1–38.

Freitag, D. 1999. *Machine Learning for Information Extraction in Informal Domains.* Ph.D. Dissertation, Carnegie Melon University.

Jelinek, F., and Mercer, R. 1980. Interpolated estimation of Markov source parameters from sparse data. In Gelsema, S., and Kanal, L. N., eds., *Pattern Recognition in Practice*, 381–402.

Kupiec, J. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language* 6:225–242.

Lee, K.-F. 1989. *Automatic Speech Recognition: The Development of the SPHINX System.* Kluwer Academic Publishers.

Leek, T. R. 1997. Information extraction using hidden Markov models. Master's thesis, UC San Diego.

Lewis, D. 1992. *Representation and Learning in Information Retrieval.* Ph.D. Dissertation, Univ. of Massachusetts. CS Tech. Report 91–93.

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

Seymore, K.; McCallum, A.; and Rosenfeld, R. 1999. Learning hidden markov model structure for information extraction. Submitted to the AAAI-99 Workshop on Machine Learning for Information Extraction.

van Mulbregt, P.; Carp, I.; Gillick, L.; and Yamron, J. 1998. Text segmentation and topic tracking on broadcast news via a hidden markov model approach. In *Proceedings of the International Conference on Spoken Language Processing.*