Learning Hidden Markov Model Structure for Information Extraction

Kristie Seymore[†]
kseymore@ri.cmu.edu

Andrew McCallum[‡†]
mccallum@justresearch.com

Ronald Rosenfeld[†]
roni@cs.cmu.edu

[†]School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[‡]Just Research
4616 Henry Street
Pittsburgh, PA 15213

Abstract

Statistical machine learning techniques, while well proven in fields such as speech recognition, are just beginning to be applied to the information extraction domain. We explore the use of hidden Markov models for information extraction tasks, specifically focusing on how to learn model structure from data and how to make the best use of labeled and unlabeled data. We show that a manually-constructed model that contains multiple states per extraction field outperforms a model with one state per field, and discuss strategies for learning the model structure automatically from data. We also demonstrate that the use of distantly-labeled data to set model parameters provides a significant improvement in extraction accuracy. Our models are applied to the task of extracting important fields from the headers of computer science research papers, and achieve an extraction accuracy of 92.9%.

Introduction

Hidden Markov modeling is a powerful statistical machine learning technique that is just beginning to gain use in information extraction tasks. Hidden Markov models (HMMs) offer the advantages of having strong statistical foundations that are well-suited to natural language domains, handling new data robustly, and being computationally efficient to develop and evaluate due to the existence of established training algorithms. The disadvantages of using HMMs are the need for an *a priori* notion of the model topology and, as with any statistical technique, large amounts of training data.

This paper focuses on two aspects of using HMMs for information extraction. First, we investigate learning model structure from data. Most applications of HMMs assume a fixed model structure (the number of states and the transitions between the states), which is selected by hand *a priori* according to the domain. We argue that for information extraction, the correct model topology is not apparent, and that the typical solution of using one state per class may not be optimal.

Second, we examine the role of labeled and unlabeled data in the training of HMMs. We introduce the concept of distantly-labeled data, which is labeled data from another domain whose labels partially overlap those from the target domain. We show how using distantly-labeled data consistently improves classification accuracy.

Hidden Markov models, while relatively new to information extraction, have enjoyed success in related natural language tasks. They have been widely used for part-of-speech tagging (Kupiec 1992), and have more recently been applied to topic detection and tracking (Yamron *et al.* 1998) and dialog act modeling (Stolcke, Shriberg, & others 1998). Other systems using HMMs for information extraction include those by Leek (1997), who extracts gene names and locations from scientific abstracts, and the Nymble system (Bikel *et al.* 1997) for named-entity extraction. Unlike our work, these systems do not consider automatically determining model structure from data; they either use one state per class, or use hand-built models assembled by inspecting training examples. Freitag & McCallum (1999) hand-build multiple HMMs, one for each field to be extracted, and focus on modeling the immediate prefix, suffix, and internal structure of each field; in contrast, we focus on learning the structure of one HMM to extract all the relevant fields, taking into account field sequence.

Our work on HMMs is centered around the task of extracting information from the headers of computer science research papers. The header of a research paper consists of all the words preceding the main body of the paper, and includes the title, author names, affiliations and addresses. Automatically extracting fields such as these is useful in constructing a searchable database of computer science research. The models we describe in this paper are used as part of the Cora computer science research paper search engine (McCallum *et al.* 1999), available at http://www.cora.justresearch.com.

The remainder of the paper is structured as follows: first, we review the basics of hidden Markov models. Then, we discuss how to learn model structure from data and examine how to estimate model parameters from labeled, unlabeled and distantly-labeled data. Next, we present experimental results on extracting important fields from the headers of computer science research papers. Finally, we conclude with a breakdown of the errors that the HMMs are making and a discussion of future work towards improving the models.

Information Extraction with Hidden Markov Models

Hidden Markov models provide a natural framework for modeling the production of the headers of research papers. We want to label each word of a header as belonging to a class such as title, author, date, or keyword. We do this by modeling the entire header (and all of the classes to extract) with one HMM. This task varies from the more classical extraction task of identifying a small set of target words from a large document containing mostly uninformative text.

Discrete output, first-order HMMs are composed of a set of states Q, with specified initial and final states q_I and q_F, a set of transitions between states $(q \to q')$, and a discrete vocabulary of output symbols $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_M\}$. The model generates a string $\mathbf{x} = x_1 x_2 \ldots x_l$ by beginning in the initial state, transitioning to a new state, emitting an output symbol, transitioning to another state, emitting another symbol, and so on, until a transition is made into the final state. The parameters of the model are the transition probabilities $\mathrm{P}(q \to q')$ that one state follows another and the emission probabilities $\mathrm{P}(q \uparrow \sigma)$ that a state emits a particular output symbol. The probability of a string $\mathbf{x}$ being emitted by an HMM M is computed as a sum over all possible paths by:

$$\mathrm{P}(\mathbf{x}|M) = \sum_{q_1,\ldots,q_l \in Q^l} \prod_{k=1}^{l+1} \mathrm{P}(q_{k-1} \to q_k)\mathrm{P}(q_k \uparrow x_k), \quad (1)$$

where q_0 and q_{l+1} are restricted to be q_I and q_F respectively, and x_{l+1} is an end-of-string token. The Forward algorithm can be used to calculate this probability (Rabiner 1989). The observable output of the system is the sequence of symbols that the states emit, but the underlying state sequence itself is hidden. One common goal of learning problems that use HMMs is to recover the state sequence $V(\mathbf{x}|M)$ that has the highest probability of having produced an observation sequence:

$$V(\mathbf{x}|M) = \arg\max_{q_1 \cdots q_l \in Q^l} \prod_{k=1}^{l+1} \mathrm{P}(q_{k-1} \to q_k)\mathrm{P}(q_k \uparrow x_k). \quad (2)$$

Fortunately, the Viterbi algorithm (Viterbi 1967) efficiently recovers this state sequence.

HMMs may be used for information extraction from research paper headers by formulating a model in the following way: each state is associated with a class that we want to extract, such as title, author or affiliation. Each state emits words from a class-specific unigram distribution. We can learn the class-specific unigram distributions and the state transition probabilities from training data. In order to label a new header with classes, we treat the words from the header as observations and recover the most-likely state sequence with the Viterbi algorithm. The state that produces each word is the class tag for that word. An example HMM, annotated with class labels and transition probabilities, is shown in Figure 1.

Learning Model Structure from Data

In order to build an HMM for information extraction, we must first decide how many states the model should contain, and what transitions between states should be allowed. A reasonable initial model is to use one state per class, and to allow transitions from any state to any other state (a fully-connected model.) However, this model may not be optimal in all cases. When a specific hidden sequence structure is expected in the extraction domain, we may do better by building a model with multiple states per class, with only a few transitions out of each state. Such a model can make finer distinctions about the likelihood of encountering a class at a particular location in the document, and can model specific local emission distribution differences between states of the same class.

An alternative to simply assigning one state per class is to learn the model structure from training data. Training data labeled with class information can be used to build a maximally-specific model. Each word in the training data is assigned its own state, which transitions to the state of the word that follows it. Each state is associated with the class label of its word token. A transition is placed from the start state to the first state of each training instance, as well as between the last state of each training instance and the end state.

This model can be used as the starting point of a variety of state merging techniques. We propose two simple types of merges that can be used to generalize the maximally-specific model. First, "neighbor-merging" combines all states that share a transition and have the same class label. For example, the sequence of adjacent title states from a single header are merged into a single title state. As multiple neighbor states with the same class label are merged into one, a self-transition loop is introduced, whose probability represents the expected state duration for that class.

Second, "V-merging" merges any two states that have the same label and share transitions from or to a common state. V-merging reduces the branching factor of the maximally-specific model. Here, we apply V-merging to models that have already undergone neighbor-merging. For example, instead of beginning in the start state and selecting from among many transitions into title states, the V-merged model would merge the children title states into one, so that only one transition from the start state to the title state would remain. The V-merged model can be used for extraction directly, or more state merges can be made automatically or by hand to generalize the model further.

Model structure can be learned automatically from data, starting with either a maximally-specific, neighbor-merged or V-merged model, using a technique like Bayesian model merging (Stolcke 1994). Bayesian model merging seeks to find the model structure that maximizes the probability of the model M given some training data D, by iteratively merging states until an optimal tradeoff between fit to the data and model size has been reached. This relationship is expressed using

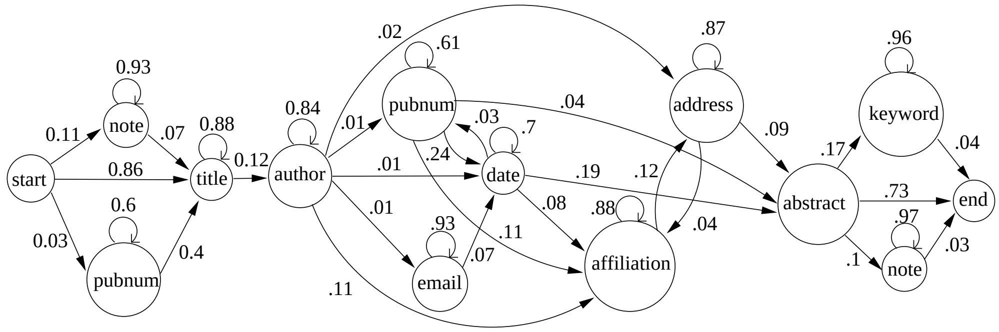

Figure 1: Example HMM. Each state emits words from a class-specific multinomial distribution.

Bayes' rule as:

$$P(M|D) \propto P(D|M)P(M). \quad (3)$$

$P(D|M)$ can be calculated with the Forward algorithm, or approximated with the probability of the Viterbi paths. The model prior can be formulated to reflect a preference for smaller models. We are implementing Bayesian model merging so that learning the appropriate model structure for extraction tasks can be accomplished automatically.

Labeled, Unlabeled, and Distantly-labeled Data

Once a model structure has been selected, the transition and emission parameters need to be estimated from training data. While obtaining unlabeled training data is generally not too difficult, acquiring labeled training data is more problematic. Labeled data is expensive and tedious to produce, since manual effort is involved. It is also valuable, since the counts of class transitions $c(q \rightarrow q')$ and the counts of a word occurring in a class $c(q \uparrow \sigma)$ can be used to derive maximum likelihood estimates for the parameters of the HMM:

$$\hat{\mathrm{P}}(q \rightarrow q') = \frac{c(q \rightarrow q')}{\sum_{s \in Q} c(q \rightarrow s)}, \quad (4)$$

$$\hat{\mathrm{P}}(q \uparrow \sigma) = \frac{c(q \uparrow \sigma)}{\sum_{\rho \in \Sigma} c(q \uparrow \rho)}. \quad (5)$$

Smoothing of the distributions is often necessary to avoid probabilities of zero for the transitions or emissions that do not occur in the training data.

Unlabeled data, on the other hand, can be used with the Baum-Welch training algorithm (Baum 1972) to train model parameters. The Baum-Welch algorithm is an iterative expectation-maximization (EM) algorithm that, given an initial parameter configuration, adjusts model parameters to locally maximize the likelihood of unlabeled data. Baum-Welch training suffers from the fact that it finds local maxima, and is thus sensitive to initial parameter settings.

A third source of valuable training data is what we refer to as *distantly-labeled* data. Sometimes it is possible to find data that is labeled for another purpose, but which can be partially applied to the domain at hand. In these cases, it may be that only a portion of the labels are relevant, but the corresponding data can still be added into the model estimation process in a helpful way. For example, BibTeX files are bibliography databases that contain labeled citation information. Several of the labels that occur in citations, such as title and author, also occur in the headers of papers, and this labeled data can be used in training emission distributions for header extraction. However, other BibTeX fields are not relevant to the header extraction task, and the data does not include any information about sequences of classes in headers.

Experiments

The goal of our information extraction experiments is to extract relevant information from the headers of computer science research papers. We define the header of a research paper to be all of the words from the beginning of the paper up to either the first section of the paper, usually the introduction, or to the end of the first page, whichever occurs first. The abstract is automatically located using regular expression matching and changed to the single token +ABSTRACT+. Likewise, a single token is added to the end of each header, either +INTRO+ or +PAGE+, to indicate the case which terminated the header. A few special classes of words are identified using simple regular expressions and converted to special tokens, such as <EMAIL>, <WEB>, <YEAR_NUMBER>, <ZIP_CODE>, <NUMBER>, and <PUBLICATION_NUMBER>. All punctuation, case and newline information is removed from the text.

The target classes we wish to identify include the following fifteen categories: title, author, affiliation, address, note, email, date, abstract, introduction (intro), phone, keywords, web, degree, publication number (pubnum), and page. The abstract, intro and page classes are each represented by a state that outputs only one token, +ABSTRACT+, +INTRO+, or +PAGE+, respectively. The degree class captures the language associated with Ph.D. or Master's theses, such as "submitted in partial fulfillment of..." and "a thesis by...". The note field commonly accounts for phrases from acknowledgements, copyright notices, and citations.

Type	Source	Word Tokens
Labeled	500 headers	23,557
Unlabeled	5,000 headers	287,770
Distantly-labeled	176 BibTeX files	2,390,637

Table 1: Sources and amounts of training data.

One thousand headers were manually tagged with class labels. Sixty-five of the headers were discarded due to poor formatting, and the rest were split into a 500-header, 23,557 word token labeled training set and a 435-header, 20,308 word token test set. Five thousand unlabeled headers, composed of 287,770 word tokens were designated as unlabeled training data. Distantly-labeled training data was acquired from 176 BibTeX files that were collected from the Web. These files consist of 2.4 million words, which contribute to the following nine header classes: address, affiliation, author, date, email, keyword, note, title, and web. The training data sources and amounts are summarized in Table 1.

Class emission distributions are trained using either the labeled training data (L), a combination of the labeled and distantly-labeled data (L+D), or a linear interpolation of the labeled and distantly-labeled data (L*D). In the L+D case, the word counts of the labeled and distantly-labeled data are added together before deriving the emission distributions. In the L*D case, separate emission distributions are trained for the labeled and distantly-labeled data, and then the two distributions are interpolated together using mixture weights derived from leave-one-out expectation-maximization of the labeled data.

For each emission distribution training case, a fixed vocabulary is derived from all of the words in the training data used. The labeled data results in a 4,914-word vocabulary, and the labeled and distantly-labeled data together contain 92,426 words. Maximum likelihood emission estimates are computed for each class, and then smoothed using absolute discounting (Ney, Essen, & Kneser 1994) to avoid probabilities of zero for the vocabulary words that are not observed for a particular class. The unknown word token `<UNK>` is added to the vocabularies to model out-of-vocabulary words. Any words in the testing data that are not in the vocabulary are mapped to this token. The probability of the unknown word is estimated separately for each class, and is assigned a portion of the discount mass proportional to the fraction of singleton words observed only in the current class.

Model Selection

We build several HMM models, varying model structures and training conditions, and test the models by finding the Viterbi paths for the test set headers. Performance is measured by word classification accuracy, which is the percentage of header words that are emitted by a state with the label of the words' true label.

Model	# states	# trans	Accuracy		
			L	L+D	L*D
full	17	255	62.8	57.4	64.5
self	17	252	85.9	83.1	89.4
ML	17	149	90.5	89.4	92.4
smooth	17	255	89.9	88.8	92.0

Table 2: Extraction accuracy (%) for models with one state per class.

The first set of models each use one state per class. Emission distributions are trained for each class on either the labeled data (L), the combination of the labeled and distantly-labeled data (L+D), or the interpolation of the labeled and distantly-labeled data (L*D). Extraction accuracy results for these models are reported in Table 2.

The full model is a fully-connected model where all transitions are assigned uniform probabilities. It relies only on the emission distributions to choose the best path through the model, and achieves a maximum accuracy of 64.5%. The self model is similar, except that the self-transition probability is set according to the maximum likelihood estimate from the labeled data, with all other transitions set uniformly. This model benefits from the additional information of the expected number of words to be emitted by each state, and its accuracy jumps to 89.4%. The ML model sets all transition parameters to their maximum likelihood estimates, and achieves the best result of 92.4% among this set of models. The smooth model adds an additional smoothing count of one to each transition, so that all transitions have non-zero probabilities, but smoothing the transition probabilities does not improve tagging accuracy. For all models, the combination of the labeled and unlabeled data (L+D) negatively affects performance relative to the labeled data results. However, the interpolation of the distantly-labeled data with the labeled data (L*D) consistently provides several percentage points improvement in accuracy over training on the labeled data alone. We will refer back to the ML model results in the next comparisons, as the best representative of the models with one state per class.

Next, we want to see if models with structures derived from data outperform the ML model. We first consider models built with a combination of automated and manual techniques. Starting from a neighbor-merged model of 805 states built from 100 randomly selected labeled training headers, states with the same class label are manually merged in an iterative manner. (We use only 100 of the 500 headers to keep the manual state selection process manageable.) Transition counts are preserved throughout the merges so that maximum likelihood transition probabilities can be estimated. Each state uses its smoothed class emission distribution estimated from the interpolation of the labeled and distantly-labeled data (L*D). Extraction per-

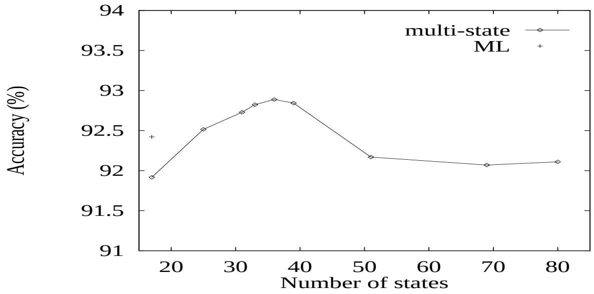

Figure 2: Extraction accuracy for multi-state models as states are merged.

Model	# states	# trans	Accuracy		
			L	L+D	L*D
ML	17	149	90.5	89.4	92.4
M-merged	36	164	91.3	90.5	92.9
V-merged	155	402	90.6	89.7	92.7

Table 3: Extraction accuracy (%) for models learned from data compared to the best model that uses one state per class.

formance, measured as the number of states decreases, is plotted in Figure 2. The performance of the ML model is indicated on the figure with a '+'. The models with multiple states per class outperform the ML model, particularly when 30 to 40 states are present. The best performance of 92.9% is obtained by the model containing 36 states. We refer to this model as the M-merged model. This result shows that more complex model structure benefits extraction performance of HMMs on the header task.

We compare this result to the performance of a 155-state V-merged model created entirely automatically from the labeled training data. A summary of the results of the ML model, the M-merged model, and the V-merged model is presented in Table 3. Both the M-merged and V-merged models outperform the ML model under all three training conditions, with the M-merged model performing the best. We expect that our future work on Bayesian model merging will result in a fully automated construction procedure that produces models performing even better than the manually created M-merged model.

Next, we investigate how to incorporate unlabeled data into our parameter training scheme. Starting with the ML and M-merged models, we run Baum-Welch training on the unlabeled data. Initial parameters are set to the maximum likelihood transition probabilities from the labeled data and the interpolated (L*D) emission distributions. Baum-Welch training produces new transition and emission parameter values which locally maximize the likelihood of the unlabeled data.

The models are tested under three different conditions; the extraction results, as well as the model perplexities on the test set, are shown in Table 4. Perplexity is a measure of how well the HMMs model the data; a lower value indicates a model that assigns a higher likelihood to the observations from the test set.

	ML		M-merged	
	Acc.	PP	Acc.	PP
initial	92.4	471	92.9	482
$\lambda = 0.5$	90.1	374	89.4	361
λ varies	89.7	364	88.8	349

Table 4: Extraction accuracy (%) and test set perplexity (PP) for the ML and M-merged models after Baum-Welch training.

The "initial" result is the performance of the models using the initial parameter estimates. These results are the same as the L*D case in Table 3. Since the vocabulary words that do not occur in the unlabeled data are given a probability of zero in the newly-estimated emission distributions resulting from Baum-Welch training, the new distributions need to be smoothed with the initial estimates. Each state's newly-estimated emission distribution is linearly interpolated with its initial distribution using a mixture weight of λ. For the "$\lambda = 0.5$" setting, both distributions for each state use a weight of 0.5. Alternatively, the Viterbi paths of the labeled training data can be computed for each model using the "$\lambda = 0.5$" emission distributions. The words emitted by each state are then used to estimate optimal mixture weights for the local and initial distributions using the EM algorithm. These mixture weights are used in the "λ varies" case.

The smoothed Baum-Welch emission estimates degrade classification performance for both the ML and M-merged models. The lack of improvement in classification accuracy can be partly explained by the fact that Baum-Welch training maximizes the likelihood of the unlabeled data, not the classification accuracy. The better modeling capabilities are pointed out through the improvement in test set perplexity. The perplexity of the test set improves over the initial settings with Baum-Welch reestimation, and improves even further with careful selection of the emission distribution mixture weights. Merialdo (1994) finds a similar effect on tagging accuracy when training part-of-speech taggers using Baum-Welch training when starting from well-estimated initial parameter estimates.

Error Breakdown

We conclude these experiments with a breakdown of the errors being made by the best performing models. Table 5 shows the errors in each class for the ML and M-merged models when using emission distributions trained on labeled (L) and interpolated (L*D) data. Classes for which there is distantly-labeled training data are indicated in bold. For several of the classes, such as title and author, there is a noticeable increase

	ML		M-merged	
Tag	L	L*D	L	L*D
All	90.5	92.4	91.3	92.9
Abstract	100	100	98.4	98.7
Address	95.8	95.5	95.2	95.1
Affiliation	87.9	91.4	88.4	90.7
Author	95.8	97.7	95.1	97.2
Date	97.6	96.9	96.9	97.2
Degree	75.8	70.8	80.3	73.2
Email	89.2	89.0	87.5	86.9
Keyword	92.2	98.1	97.3	98.9
Note	84.9	85.1	88.1	89.0
Phone	93.7	93.1	89.7	87.4
Pubnum	65.0	65.0	61.3	60.6
Title	93.4	98.4	93.2	97.8
Web	80.6	83.3	41.7	41.7

Table 5: Individual class results for the ML and M-merged models. Classes noted in bold occur in distantly-labeled data.

in accuracy when the distantly-labeled data is included. The poorest performing individual classes are the degree, publication number, and web classes. The web class has a particularly low accuracy for the M-merged model, when limited web class examples in the 100 training headers probably kept the web state from having transitions to and from as many states as necessary.

Conclusions and Future Work

Our experiments show that hidden Markov models do well at extracting important information from the headers of research papers. We achieve an accuracy of 92.9% over all classes of the headers, and class-specific accuracies of 97.8% for titles and 97.2% for authors. We have demonstrated that models that contain more than one state per class do provide increased extraction accuracy over models that use only one state per class. This improvement is due to more specific transition context modeling that is possible with more states. We expect that it is also beneficial to have localized emission distributions, which can capture distribution variations that are dependent on the position of the class in the header.

Distantly-labeled data has proven to be valuable in providing robust parameter estimates. The interpolation of distantly-labeled data provides a consistent increase in extraction accuracy for headers. In cases where little labeled training data is available, distantly-labeled data is a helpful resource.

Forthcoming experiments include using Bayesian model merging to learn model structure completely automatically from data, as well as taking advantage of additional header features such as the positions of the words on the page. We expect the inclusion of layout information to particularly improve extraction accuracy.

Finally, we also plan to model internal state structure, in order to better capture the first and last few words absorbed by each state. A possibly useful internal state structure is displayed in Figure 3. In this case, the distributions for the first and last two words are modeled explicitly, and an internal state emits all other words. We expect these improvements will contribute to the development of more accurate models for research paper header extraction.

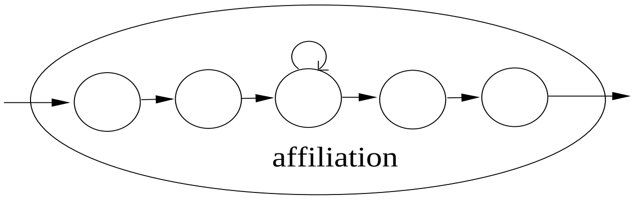

Figure 3: Proposed internal model structure for states.

References

Baum, L. 1972. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process. *Inequalities* 3:1–8.

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.

Freitag, D., and McCallum, A. 1999. Information extraction with HMMs and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction.*

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

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

McCallum, A.; Nigam, K.; Rennie, J.; and Seymore, K. 1999. A machine learning approach to building domain-specific search engines. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence.*

Merialdo, B. 1994. Tagging english text with a probabilistic model. *Computational Linguistics* 20(2):155–171.

Ney, H.; Essen, U.; and Kneser, R. 1994. On structuring probabilistic dependencies in stochastic language modeling. *Computer Speech and Language* 8(1):1–38.

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

Stolcke, A.; Shriberg, E.; et al. 1998. Dialog act modeling for conversational speech. In *Applying Machine Learning to Discourse Processing, 1998 AAAI Spring Symposium*, number SS-98-01, 98–105. Menlo Park,CA: AAAI Press.

Stolcke, A. 1994. *Bayesian Learning of Probabilistic Language Models.* Ph.D. Dissertation, University of California, Berkeley, CA.

Viterbi, A. J. 1967. Error bounds for convolutional codes and an asymtotically optimum decoding algorithm. *IEEE Transactions on Information Theory* IT-13:260–267.

Yamron, J.; Carp, I.; Gillick, L.; Lowe, S.; and van Mulbregt, P. 1998. A hidden Markov model approach to text segmentation and event tracking. In *Proceedings of the IEEE ICASSP.*