Maximum Entropy Markov Models for Information Extraction and Segmentation

Andrew McCallum MCCALLUM@JUSTRESEARCH.COM
Dayne Freitag DAYNE@JUSTRESEARCH.COM
Just Research, 4616 Henry Street, Pittsburgh, PA 15213 USA

Fernando Pereira PEREIRA@RESEARCH.ATT.COM
AT&T Labs - Research, 180 Park Ave, Florham Park, NJ 07932 USA

Abstract

Hidden Markov models (HMMs) are a powerful probabilistic tool for modeling sequential data, and have been applied with success to many text-related tasks, such as part-of-speech tagging, text segmentation and information extraction. In these cases, the observations are usually modeled as multinomial distributions over a discrete vocabulary, and the HMM parameters are set to maximize the likelihood of the observations. This paper presents a new Markovian sequence model, closely related to HMMs, that allows observations to be represented as arbitrary overlapping features (such as word, capitalization, formatting, part-of-speech), and defines the conditional probability of state sequences given observation sequences. It does this by using the maximum entropy framework to fit a set of exponential models that represent the probability of a state given an observation and the previous state. We present positive experimental results on the segmentation of FAQ's.

1. Introduction

The large volume of text available on the Internet is causing an increasing interest in algorithms that can automatically process and mine information from this text. Hidden Markov models (HMMs) are a powerful tool for representing sequential data, and have been applied with significant success to many text-related tasks, including part-of-speech tagging (Kupiec, 1992), text segmentation and event tracking (Yamron, Carp, Gillick, Lowe, & van Mulbregt, 1998), named entity recognition (Bikel, Schwartz, & Weischedel, 1999) and information extraction (Leek, 1997; Freitag & McCallum, 1999).

HMMs are probabilistic finite state models with parameters for state-transition probabilities and state-specific observation probabilities. Greatly contributing to their popularity is the availability of straightforward procedures for training by maximum likelihood (Baum-Welch) and for using the trained models to find the most likely hidden state sequence corresponding to an observation sequence (Viterbi).

In text-related tasks, the observation probabilities are typically represented as a multinomial distribution over a discrete, finite vocabulary of words, and Baum-Welch training is used to learn parameters that maximize the probability of the observation sequences in the training data.

There are two problems with this traditional approach. First, many tasks would benefit from a richer representation of observations—in particular a representation that describes observations in terms of many overlapping features, such as capitalization, word endings, part-of-speech, formatting, position on the page, and node memberships in WordNet, in addition to the traditional word identity. For example, when trying to extract previously unseen company names from a newswire article, the identity of a word alone is not very predictive; however, knowing that the word is capitalized, that is a noun, that it is used in an appositive, and that it appears near the top of the article would all be quite predictive (in conjunction with the context provided by the state-transition structure). Note that these features are not independent of each other.

Furthermore, in some applications the set of all possible observations is not reasonably enumerable. For example, it may beneficial for the observations to be whole lines of text. It would be unreasonable to build a multinomial distribution with as many dimensions as there are possible lines of text. Consider the task of segmenting the questions and answers of a frequently asked questions list (FAQ). The features that are indicative of the segmentation are not just the individual words themselves, but features of the line as a whole, such as the line length, indentation, total amount of whitespace, percentage of non-alphabetic characters, and grammatical features. We would like the observations to be parameterized with these overlapping features.

The second problem with the traditional approach is that it sets the HMM parameters to maximize the likelihood of the observation sequence; however, in most text applications, including all those listed above, the task is to predict the state sequence *given* the observation sequence. In other words, the traditional approach inappropriately uses a generative *joint* model in order to solve a *conditional* problem in which the observations are given.

This paper introduces maximum entropy Markov models (MEMMs), which address both of these concerns. To allow for non-independent, difficult to enumerate observation features, we move away from the generative, joint probability parameterization of HMMs to a conditional model that represents the probability of reaching a state given an observation and the previous state. These conditional probabilities are specified by exponential models based on arbitrary observation features.[1] The exponential models follow from a maximum entropy argument, and are trained by *generalized iterative scaling* (GIS) (Darroch & Ratcliff, 1972), which is similar in form and computational cost to the expectation-maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977). The "three classic problems" (Rabiner, 1989) of HMMs can all be straightforwardly solved in this new model with new variants of the forward-backward, Viterbi and Baum-Welch algorithms.

The remainder of the paper describes our alternative model in detail, explains how to fit the parameters using GIS, (for both known and unknown state sequences), and presents the variant of the forward-backward procedure, out of which solutions to the "classic problems" follow naturally. We also give experimental results for the problem of extracting the question-answer pairs in lists of frequently asked questions (FAQs), showing that our model increases both precision and recall, the former by a factor of two.

2. Maximum-Entropy Markov Models

A hidden Markov model (HMM) is a finite state automaton with stochastic state transitions and observations (Rabiner, 1989). The automaton models a probabilistic generative process whereby a sequence of observations is produced by starting in some state, emitting an observation selected by that state, transitioning to a new state, emitting another observation—and so on until a designated final state is reached. More formally, the HMM is given by a finite set of states S, a set of possible observations O, two conditional probability distributions: a state transition probability from s' to s, $\mathrm{P}(s|s')$ for $s, s' \in S$ and an observation probability distribution, $\mathrm{P}(o|s)$ for $o \in O, s \in S$, and an initial state distribution $\mathrm{P}_0(s)$. A run of the HMM pairs an observation sequence $o_1 \cdots o_m$ with a state sequence $s_1 \cdots s_m$. In text-based tasks, the set of possible observations is typically a finite character set or vocabulary.

[1] States as well as observations could be represented by features, but we will defer the discussion of that refinement to later.

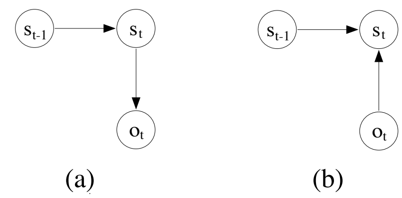

Figure 1. **(a)** The dependency graph for a traditional HMM; **(b)** for our conditional maximum entropy Markov model.

In a supervised task, such as information extraction, there is a sequence of labels $l_1 \cdots l_m$ attached to each training observation sequence $o_1 \cdots o_m$. Given a novel observation, the objective is to recover the most likely label sequence. Typically, this is done with models that associate one or more states with each possible label. If there is a one-to-one mapping between labels and states, the sequence of states is known for any training instance; otherwise, the state sequence must be estimated. To label an unlabeled observation sequence, the Viterbi path is calculated, and the labels associated with that path are returned.

2.1 The New Model

As an alternative to HMMs, we propose *maximum entropy Markov models* (MEMMs), in which the HMM transition and observation functions are replaced by a single function $\mathrm{P}(s|s', o)$ that provides the probability of the current state s given the previous state s' and the current observation o. In this model, as in most applications of HMMs, the observations are *given*—reflecting the fact that we don't actually care about their probability, only the probability of the state sequence (and hence label sequence) they induce. In contrast to HMMs, in which the current observation only depends on the current state, the current observation in an MEMM may also depend on the previous state. It can then be helpful to think of the observations as being associated with state transitions rather than with states. That is, the model is in the form of probabilistic finite-state acceptor (Paz, 1971), in which $\mathrm{P}(s|s', o)$ is the probability of the transition from state s' to state s on input o.

In what follows, we will split $\mathrm{P}(s|s', o)$ into $|S|$ separately trained *transition* functions $\mathrm{P}_{s'}(s|o) = \mathrm{P}(s|s', o)$. Each of these functions is given by an exponential model, as described later in Section 2.3.

Next we discuss how to solve the state estimation problem in the new framework.

2.2 State Estimation from Observations

Despite the differences between the new model and HMMs, there is still an efficient dynamic programming solution the classic problem of identifying the most likely state sequence given an observation sequence. The Viterbi algorithm for HMMs fills in a dynamic programming table with *forward probabilities* $\alpha_t(s)$, defined as the probability of producing the observation sequence up to time t *and* being in state s at time t. The recursive Viterbi step is $\alpha_{t+1}(s) = \sum_{s' \in S} \alpha_t(s') \cdot \mathrm{P}(s|s') \cdot \mathrm{P}(o_{t+1}|s)$.

In the new model, we redefine $\alpha_t(s)$ to be the probability of being in state s at time t *given* the observation sequence up to time t. The recursive Viterbi step is then

$$\alpha_{t+1}(s) = \sum_{s' \in S} \alpha_t(s') \cdot \mathrm{P}_{s'}(s|o_{t+1}). \tag{1}$$

The corresponding *backward probability* $\beta_t(s)$ (used for Baum-Welch, which is discussed later) is the probability of starting from state s at time t *given* the observation sequence after time t. Its recursive step is simply $\beta_t(s') = \sum_{s \in S} \mathrm{P}(s|s', o_t) \cdot \beta_{t+1}(s)$. Space limitations prevent a full description here of Viterbi and Baum-Welch; see Rabiner (1989) for an excellent tutorial.

2.3 An Exponential Model for Transitions

The use of state-observation transition functions rather than the separate transition and observation functions in HMMs allows us to model transitions in terms of multiple, non-independent features of observations, which we believe to be the most valuable contribution of the present work. To do this, we turn to exponential models fit by maximum entropy.

Maximum entropy is a framework for estimating probability distributions from data. It is based on the principle that the best model for the data is the one that is consistent with certain constraints derived from the training data, but otherwise makes the fewest possible assumptions. In our probabilistic framework, the distribution with the "fewest possible assumptions" is that which is closest to the uniform distribution, that is, the one with the highest entropy.

Each constraint expresses some characteristic of the training data that should also be present in the learned distribution. Our constraints will be based on n binary features.[2] Examples of such features might be "the observation is the word *apple*" or "the observation is a capitalized word" or, if the observations are whole lines of text at time, "the observation is a line of text that has two noun phrases." As in other conditional maximum entropy models, features do not depend only on the observation but also on the outcome predicted by the function being modeled. Here, that function is the s'-specfic transition function $\mathrm{P}_{s'}(s|o)$, and the outcome is the new current state s. Thus, each feature a gives a function $f_a(o, s)$ of two arguments, a current observation o and a possible new current state s.

[2] We use binary features in this paper, but the maximum entropy framework can in general handle real-valued features.

In this paper, each such a is a pair $a = \langle b, s \rangle$, where b is a binary feature of the observation alone and s is a destination state:

$$f_{\langle b,s \rangle}(o_t, s_t) = \begin{cases} 1 & \text{if } b(o_t) \text{ is true and } s = s_t \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The algorithm description that follows can be expressed in terms of the generic feature a without reference to this particular feature decomposition. Furthermore, we will suggest later that more general features may be useful.

The constraints we apply are that the expected value of each feature in the learned distribution be the same as its average on the training observation sequence $o_1 \cdots o_m$ (with corresponding state sequence $s_1 \cdots s_m$). Formally, for each previous state s' and feature a, the transition function $\mathrm{P}_{s'}(s|o)$ must have the property that

$$\begin{aligned} &\tfrac{1}{m_{s'}} \textstyle\sum_{k=1}^{m_{s'}} f_a(o_{t_k}, s_{t_k}) = \\ &\qquad \tfrac{1}{m_{s'}} \textstyle\sum_{k=1}^{m_{s'}} \sum_{s \in S} \mathrm{P}_{s'}(s|o_{t_k}) f_a(o_{t_k}, s), \end{aligned} \tag{3}$$

where $t_1, \ldots, t_{m_{s'}}$ are the time steps with $s_{t_k} = s'$, *i.e.*, the time steps that involve the transition function $\mathrm{P}_{s'}$.

The maximum entropy distribution that satisfies those constraints (Della Pietra, Della Pietra, & Lafferty, 1997) is unique, agrees with the maximum-likelihood distribution and has the exponential form:

$$\mathrm{P}_{s'}(s|o) = \frac{1}{Z(o, s')} \exp\left(\sum_a \lambda_a f_a(o, s) \right), \tag{4}$$

where the λ_a are parameters to be learned and $Z(o, s')$ is the normalizing factor that makes the distribution sum to one across all next states s.

2.4 Parameter Estimation by Generalized Iterative Scaling

GIS (Darroch & Ratcliff, 1972) finds iteratively the λ_a values that form the maximum entropy solution for each transition function (Eq. 4). It requires that the values of the features sum to the same (arbitrary) constant C for each context $\langle o, s \rangle$. If this is not already true, we make it true by adding a new ordinal-valued "correction" feature f_x, where $x = n + 1$, such that $f_x(o, s) = C - \sum_{a=1}^{n} f_a(o, s)$ and C is chosen to be large enough that $f_x(o, s) \geq 0$ for all o and s.

- **Inputs:** An observation sequence $o_1 \cdots o_m$, a corresponding sequence of labels $l_1 \cdots l_m$, a certain number of states, each with a label, and potentially having a restricted transition structure. Also, a set of observation-state features.
- Determine the state sequence associated with the observation-label sequence. (When this is ambiguous, it can be determined probabilistically by iterating the next two steps with EM).
- Deposit state-observation pairs (s, o) into their corresponding previous states s' as training data for each state's transition function $\mathrm{P}_{s'}(s|o)$.
- Find the maximium entropy solution for each state's discriminative function by running GIS.
- **Output:** A maximum-entropy-based Markov model that takes an unlabeled sequence of observations and predicts their corresponding labels.

Table 1. An outline of the algorithm for estimating the parameters of a Maximum-Entropy Markov model.

The application of GIS to learning the transition function $\mathrm{P}_{s'}$ for state s' consists of the following steps:

1. Calculate the training data average of each feature $F_a = \frac{1}{m_s} \sum_{k=1}^{m_s} f_a(o_{t_k}, s_{t_k})$.
2. Start iteration 0 of GIS with some arbitrary parameter values, say $\lambda_a^{(0)} = 1$.
3. At iteration j, use the current $\lambda_a^{(j)}$ values in $\mathrm{P}_{s'}^{(j)}(s|o)$ (Eq. 4) to calculate the expected value of each feature: $E_a^{(j)} = \frac{1}{m_s} \sum_{k=1}^{m_s} \sum_{s \in S} \mathrm{P}_{s'}^{(j)}(s|o_{t_k}) f_a(o_{t_k}, s)$.
4. Make a step towards satisfying the constraints by changing each λ_a to bring the expected value of each feature closer to corresponding training data average: $\lambda_a^{(j+1)} = \lambda_a^{(j)} + \frac{1}{C} \log\left(\frac{F_a}{E_a^{(j)}}\right)$.
5. Until convergence is reached, return to step 3.

To summarize the overall MEMM training procedure, we first split the training data into the events—observation-destination state pairs—relevant to the transitions from each state s'. (Let us assume for the moment that, given the labels in the training sequence, the state sequence is unambiguously known.) We then apply GIS using the feature statistics for the events assigned to each s' in order to induce the transition function $\mathrm{P}_{s'}$ for s'. The set of these functions defines the desired maximum-entropy Markov model. Table 2.4 contains an overview of the maximum entropy Markov model training algorithm.

2.5 Parameter Estimation with Unknown State

The procedure described above assumes that the state sequence of the training observation sequence is known; that is, the states have to be predicted at test time but not training time. Often, it is useful to be able to train when the state sequence is not known. For example, there may be more than one state with the same label, and for a given label sequence it may be ambiguous which state produced which label instance.

We can use a variant of the Baum-Welch algorithm for this. The E-step calculates state occupancies using the forward-backward algorithm with the current transition functions. The M-step uses the GIS procedure with feature frequencies based on the E-step state occupancies to compute new transition functions. This will maximize the likelihood of the label sequence given the observations. Note that GIS does not have to be run to convergence in each M-step; not doing so would make this an example of Generalized Expectation-Maximization (GEM), which is also guaranteed to converge to a local maximum.

Notice also that the same Baum-Welch variant can be used with unlabeled or partially labeled training sequences where, not only is the state unknown, but the label itself is missing. These models could be trained with a combination of labeled and unlabeled data, which is often extremely helpful when labeled data is sparse.

2.6 Variations

We have thus far described one particular method for maximum entropy Markov models, but there are several other possibilities.

Factored state representation.

One difficulty that MEMMs share with HMMs is that there are $O(|S|^2)$ transition parameters, making data sparseness a serious problem as the number of states increases. Recall that in our model observations are associated with transitions instead of states. This has advantages for expressive power, but comes at the cost of having many more parameters. For HMMs and related graphical models, tied parameters and factored state representations (Ghahramani & Jordan, 1996; Kanazawa, Koller, & Russell, 1995) have been used to alleviate this difficulty.

We can achieve a similar effect in MEMMs by not splitting $P(s|s', o)$ into $|S|$ different functions $P_{s'}(s|o)$. Instead we would use a distributed representation for the previous state s' as a collection of features with weights set by maximum entropy, just as we have done for the observations. For example, state features might include "we have already consumed the start-time extraction field," "we haven't yet exited the preamble of the document," "the subject of the previous sentence is female" or "the last paragraph was an answer." One could also have second-order features linking observation and state features. With such a representation, information would be shared among different source states, reducing the number of parameters and thus improving generalization. Furthermore, this proposal does not require the difficult step of hand-crafting a parameter-tying scheme or graphical model for the state transition function as is required in HMMs and other graphical models.

Observations in states instead of transitions.

Rather than combining the transition and emission parameters in a single function, the transition probabilities could be represented as a traditional multinomial, $P(s|s')$, and the influence of the observations $P(s|o)$ could be represented by a maximum-entropy exponential:

$$\mathrm{P}(s|s', o) = \mathrm{P}(s|s') \frac{1}{Z(o, s')} \exp\left(\sum_a \lambda_a f_a(o, s)\right). \tag{5}$$

This method of "correcting" a simple multinomial or prior by adding extra features with maximum entropy has been used previously in various statistical language modeling problems. These include the combination of traditional trigrams with "trigger word" features (Rosenfeld, 1994) and the combination of arbitrary features of sentences with trigram models (Chen & Rosenfeld, 1999).

Note here that the observation and the previous state are treated as independent evidence for the current state. This approach would put the observations back in the states instead of the transitions. It would reduce the number of parameters, and thus might be useful when training data is especially sparse.

An Environmental Model for Reinforcement Learning.

The transition function can also include an action, a, resulting in $\mathrm{P}(s|s', o, a)$—a model suitable for representing the envinronment of a reinforcement agent. The dependency on the action could be modeled either with separate functions for each action, or with a factored represention of actions in terms of arbitrary overlapping features, such as "steer left," "beep," and "raise arm." Certain particular actions, states and observations with strong interactions can be modeled as features that represent their conjunction.

3. Experimental Results

We tested our method on a collection of 38 files[3] belonging to 7 Usenet multi-part FAQs downloaded from the Internet. All documents in this data set are organized according to the same basic structure: each contains a header, which

[3] See http://www.cs.cmu.edu/∼mccallum/faqdata.

Table 2. Excerpt from a labeled FAQ. Lines have been truncated for reasons of space. The tags at the beginnings of lines were inserted manually.

```
    <head>X-NNTP-Poster: NewsHound v1.33
    <head>
    <head>Archive-name: acorn/faq/part2
    <head>Frequency: monthly
    <head>
<question>2.6) What configuration of serial cable should I use
  <answer>
  <answer>   Here follows a diagram of the necessary connections
  <answer>programs to work properly. They are as far as I know t
  <answer>agreed upon by commercial comms software developers fo
  <answer>
  <answer>   Pins 1, 4, and 8 must be connected together inside
  <answer>is to avoid the well known serial port chip bugs. The
```

Table 3. Line-based features used in these experiments.

begins-with-number	contains-question-mark
begins-with-ordinal	contains-question-word
begins-with-punctuation	ends-with-question-mark
begins-with-question-word	first-alpha-is-capitalized
begins-with-subject	indented
blank	indented-1-to-4
contains-alphanum	indented-5-to-10
contains-bracketed-number	more-than-one-third-space
contains-http	only-punctuation
contains-non-space	prev-is-blank
contains-number	prev-begins-with-ordinal
contains-pipe	shorter-than-30

includes text in Usenet header format and occasionally a preamble or table of contents; a series of one or more question/answer pairs; and a tail, which typically includes items such as copyright notices and acknowledgments, and various artifacts reflecting the origin of the document. There are also some formatting regularities, such as indentation, numbered questions and styles of paragraph breaks. The multiple documents belonging to a single FAQ are formatted in a consistent manner, but there is considerable variation between different FAQs.

We labeled each line in this document collection into one of four categories, according to its role in the document: *head*, *question*, *answer*, *tail*, corresponding to the parts of documents described in the previous paragraph. Table 2 shows an excerpt from a labeled FAQ. The object of the task is to recover these labels. This excerpt demonstrates the difficulty of recovering line classifications by only looking at the tokens that occur in the line. In particular, the numerals in the answer might easily confuse a token-based classifier.

We defined 24 Boolean features of lines, shown in Table 3, which we believed would be useful in determining the class of a line. No effort was made to control statistical dependence between pairs of features. Although the set contains a few feature pairs which are mutually disjoint, the features represent partitions of the data that overlap to varying degrees. Note also that the usefulness of a particular feature, such as indented, depends on the formatting conventions of a particular FAQ.

The results presented in this section are meant to answer the question of how well can a MEMM trained on a single manually labeled document label novel documents formatted according to the same conventions. Our experiments treat each group of documents belonging to the same FAQ as a separate dataset. We train a model on a single document in such a group and test it on the remaining documents in the group. In other words, we perform "leave-n-minus-1-out" evaluation. Each group of n documents yields $n(n-1)$ results. Scores are the average performance across all FAQs in the collection.

Given a sequence of lines (a test document) and a MEMM we use the Viterbi algorithm to compute the most likely state sequence. We consider three metrics in evaluating the predicted sequences. The first is the *co-occurrence agreement probability* (COAP), proposed by Beeferman, Berger, and Lafferty (1999):

$$\mathrm{P}_\mu(\mathtt{act}, \mathtt{pred}) = \sum_{i,j} D_\mu(i,j)\delta_{\mathtt{act}}(i,j)\overline{\oplus}\delta_{\mathtt{pred}}(i,j)$$

where $D_\mu(i,j)$ is a probability distribution over the set of distances between lines; $\delta_{\mathtt{act}}(i,j)$ is 1 if lines i and j are in the same actual segment, and 0 otherwise; $\delta_{\mathtt{pred}}(i,j)$ is a similar indicator function for the predicted segmentation; and $\overline{\oplus}$ is the XNOR function. This metric gives the empirical probability that the actual and predicted segmentations agree on the placement of two lines drawn according to D_μ. In computing the COAP we define a segment to be any unbroken sequence of lines with the same label. In Beeferman et al. (1999), D_μ is an exponential distribution depending on μ, a parameter calculated on features of the dataset, such as average document length. For simplicity, we set D_μ to a uniform distribution of width 10. In other words, our COAP measures the probability that any two lines within 10 lines of each other are placed correctly by the predicted segmentation.

In constrast with the COAP, (which reflects the probability that segment boundaries are properly identified by the learner, but ignores the labels assigned to the segments themselves), the other two metrics only count as correct those predicted segments that have the right labels. A segment is counted as correct if it has the same boundaries *and* label (e.g., *question*) as an actual segment. The *segmentation precision* (SP) is the number of correctly identified segments divided by the number of segments predicted. The *segmentation recall* (SR) is the number of correctly identified segments divided by the number of actual segments.

We tested four different models on this dataset:

Table 4. Co-occurrence agreement probability (COAP), segmentation precision (SegPrec) and segmentation recall (SegRecall) of four learners on the FAQ dataset. All these averages have 95% confidence intervals of 0.01 or less.

Learner	*COAP*	*SegPrec*	*SegRecall*
ME-Stateless	0.520	0.038	0.362
TokenHMM	0.865	0.276	0.140
FeatureHMM	0.941	0.413	0.529
MEMM	0.965	0.867	0.681

- **ME-Stateless:** A single maximum entropy classifier trained on and applied to each line independently, using the 24 features shown in Table 3. ME-Stateless can be considered typical of any approach that treats lines in isolation from their context.
- **TokenHMM:** A traditional, fully connected HMM with four states, one for each of the line categories. The states in the HMM emit individual tokens (groups of alphanumeric characters and individual punctuation characters). The observation distribution at a given state is a smoothed multinomial over possible tokens. The label assigned to a line is that assigned to the state responsible for emitting the tokens in the line. In computing a state sequence for a document, the model is allowed to switch states only at line boundaries, thereby ensuring that all tokens in a line share the same label. This model was used in previous work on information extraction with HMMs (e.g. Freitag & McCallum, 1999).
- **FeatureHMM:** Identical to *TokenHMM*, only the lines in a document are first converted to sequences of features from Table 3. For every feature that tests true for a line, a unique symbol is inserted into the corresponding line in the converted document. The HMM is trained to emit these symbols. Notice that the emission model for each state is in this case a naïve Bayes model.
- **MEMM:** The maximum entropy Markov model described in this paper. As in the other HMMs, the model contains four labeled states and is fully connected.

Note that because training is fully supervised, the sequence of states a training document passes through is unambiguous. Consequently, training does not involve Baum-Welch reestimation.

Table 4 shows the performance of the four models on FAQ data. It is clear from the table that *MEMM* is the best of the methods tested. What is more, the results support two claims that underpin our research into this problem. First, representing lines in terms of features salient to the problem at hand is far more effective than a token-level representation, which is the essential difference between *TokenHMM* and *FeatureHMM*. *FeatureHMM* does as well as it does (surprisingly well) because it has access to meaningful features of the segmentation problem.

The performance of *ME-Stateless* show that it is not possible to classify lines unambiguously from the features alone. Our second claim is that structural regularities such as "if you have left the header, do not classify any further lines as header lines" are critical. Markov models provide a convenient way to model such regularities. For the purposes of segmentation, the results suggest that it is more important to model structure than to have access to line features.

The scores of the three Markov model methods on the COAP metric indicate that they all do reasonably well at segmenting FAQs into their constituent parts. In particular, *FeatureHMM* segments almost as well as *MEMM*. The more stringent metrics, SP and SR, which punish any misclassified line in a predicted segment, hint at the main shortcoming of the two non-maximum entropy Markov models: they occasionally interpolate bad predictions into otherwise correctly handled segments.

The precision (SP score) of *MEMM* has significant practical implications. If the results of the segmentation are to be used in an automatic system, then precision is critical. In the case of FAQs, at least one such system, a question-answering system, has been described in the literature (Burke, Hammond, Kulyukin, Lytinen, & Tomuro, 1997). Whereas the segmentation returned by *FeatureHMM* is probably not of high enough quality for such a use—not without manual intervention or rule-based post-processing—*MEMM* segmentation might be.

4. Related Work

A wide range of machine-learning techniques have been used in information extraction and text segmentation. In this paper we focus exclusively on techniques based on probabilistic models. Non-probabilistic methods such as memory-based techniques (Argamon, Dagan, & Krymolowski, 1998), transformation-based learning (Brill, 1995), and Winnow-based combinations of linear classifiers (Roth, 1998), do not give normalized scores to each decision that can be combined into overall scores for decision sequences. Therefore, they do not support the standard dynamic programming methods for finding the best segmentation (Viterbi), and must thus resort to sub-optimal methods to label the observation sequence. Furthermore, they do not support hidden-variable reestimation (Baum-Welch) methods, which are required for missing or incomplete training labels.

Exponential models derived by maximum entropy have been applied with considerable success to many natural language tasks, including language modeling for speech recognition (Rosenfeld, 1994; Chen & Rosenfeld, 1999), segmentation of newswire stories (Beeferman et al., 1999), part-of-speech tagging, prepositional phrase attachment and parsing (Ratnaparkhi, 1998).

HMMs have also been successful in similar natural-language tasks, including part-of-speech tagging (Kupiec, 1992), named-entity recognition (Bikel et al., 1999) and information extraction (Leek, 1997; Freitag & McCallum, 1999).

However, we know of no previous general method that combines the rich state representation of Markov models with the flexible feature combination of exponential models. The MENE named-entity recognizer (Borthwick, Sterling, Agichtein, & Grishman, 1998) uses an exponential model to label each word with a label indicating the position of the word in a labeled-entity class (start, inside, end or singleton), but the conditioning information does not include the previous label, unlike our model. Therefore, it is closer to our *ME-Stateless* model. It is possible that its inferior performance compared to an HMM-based named-entity recognizer (Bikel et al., 1999) may have similar causes to the corresponding weakness of *ME-Stateless* relative to *FeatureHMM* in our experiments—the lack of representation of sequential dependencies.

The model closest to our proposal is the part-of-speech tagger of Ratnaparkhi (1998). He starts with a maximum-entropy model of the joint distribution of word sequences and the corresponding part-of-speech tags, but the practical form of his model is a conditional Markov model whose states encode the past two parts-of-speech and features of the previous two and next two words. While our model splits the transition functions for different source states, Ratnaparkhi's combines all of them into a single exponential model, which is more complex but may handle sparse data better. Note, however, that it does not allow for arbitrary state-transition structures and the relatively more expressive context representation they allow.

Nevertheless, the most direct inspiration for our model was the work on Markov processes on curves (MPCs) (Saul & Rahim, 1999), which defines a class of conditional Markov models mapping (continuous) segments of a trajectory in acoustic space to states representing phonetic distinctions. Our model is a simpler, discrete time version of the same observation-conditional Markovian architecture.

5. Conclusions and Further Work

We have shown that it is possible to combine the advantages of HMMs and maximum-entropy models into a general model that allows state transitions to depend on non-independent features of the sequence under analysis. The new model performs considerably better than either HMMs or stateless maximum-entropy models on the task of segmenting FAQs into questions and answers, and we believe that the same technique can be advantageously applied to many other text-related applications, for example named entity recognition.

We believe that a distributed state representation and non-fully connected topologies may facilitate applications to more demanding tasks, such as information extraction with large vocabulary and many features, as well as automatic feature generation and selection following Della Pietra et al. (1997). We also believe it would be worth investigating training with partially labeled data using the combination of Baum-Welch and GIS discussed earlier. In the longer term, the combination of maximum entropy and conditional parameterization may be useful for a wider range of graphical models than finite-state networks.

Acknowledgements

Many thanks to John Lafferty for helpful discussions on training with unknown state and on associating observations with states instead of transitions, to Kamal Nigam for help with GIS, and to Michael Collins for guidance on related work.

References

Argamon, S., Dagan, I., & Krymolowski, Y. (1998). A memory-based approach to learning shallow natural language patterns. In *COLING-ACL 98*, pp. 67–73 New Brunswick, New Jersey. Association for Computational Linguistics.

Beeferman, D., Berger, A., & Lafferty, J. (1999). Statistical models for text segmentation. *Machine Learning*, *34*(1–3), 177–210.

Bikel, D. M., Schwartz, R. L., & Weischedel, R. M. (1999). An algorithm that learns what's in a name. *Machine Learning Journal*, *34*, 211–231.

Borthwick, A., Sterling, J., Agichtein, E., & Grishman, R. (1998). Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Proceedings of the Sixth Workshop on Very Large Corpora* New Brunswick, New Jersey. Association for Computational Linguistics.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging. *Computational Linguistics*, *21*(4), 543–565.

Burke, R., Hammond, K., Kulyukin, V., Lytinen, S., & Tomuro, N. (1997). Question answering from frequently-asked question files: Experiences with the FAQ Finder system. *AI Magazine*, *18*, 57–66.

Chen, S., & Rosenfeld, R. (1999). Efficient sampling and feature selection in whole sentence maximum entropy language models. In *Proceedings of ICASSP'99*. IEEE.

Darroch, J. N., & Ratcliff, D. (1972). Generlized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, *43*(5), 1470–1480.

Della Pietra, S., Della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(4).

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

Freitag, D., & McCallum, A. K. (1999). Information extraction using hmms and shrinkage. In *Papers from the AAAI-99 Workshop on Machine Learning for Information Extration*, pp. 31–36 Menlo Park, California. AAAI.

Ghahramani, Z., & Jordan, M. I. (1996). Factorial hidden Markov models. In Mozer, M., Touretzky, D., & Perrone, M. (Eds.), *Advances in Neural Information Processing Systems 8*. MIT Press.

Kanazawa, K., Koller, D., & Russell, S. (1995). Stochastic simulation algorithms for dynamic probabilistic networks. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* Montreal, Canada. Morgan Kaufmann.

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.

Paz, A. (1971). *Introduction to Probabilistic Automata*. Academic Press.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–285.

Ratnaparkhi, A. (1998). *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

Rosenfeld, R. (1994). *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. Ph.D. thesis, Carnegie Mellon University.

Roth, D. (1998). Learning to resolve natural language ambiguities: a unified approach. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp. 806–813 Menlo Park, California. AAAI Press.

Saul, L., & Rahim, M. (1999). Markov processes on curves for automatic speech recognition. In Kearns, M. S., Solla, S. A., & Cohn, D. A. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 11 Cambridge, Massachusetts. MIT Press.

Yamron, J., Carp, I., Gillick, L., Lowe, S., & van Mulbregt, P. (1998). A hidden Markov model approach to text segmentation and event tracking. In *Proceedings of ICASSP'98*. IEEE.