Proceedings of the International Symposium on Intelligent Data Analysis (IDA-2001). 2001.

Active Hidden Markov Models for Information Extraction

Tobias Scheffer[1,2], Christian Decomain[1,2], and Stefan Wrobel[1]

[1] University of Magdeburg, FIN/IWS, PO Box 4120, 39016 Magdeburg, Germany
[2] SemanticEdge, Kaiserin-Augusta-Allee 10-11, 10553 Berlin, Germany
{scheffer, decomain, wrobel}@iws.cs.uni-magdeburg.de

Abstract. Information extraction from HTML documents requires a classifier capable of assigning semantic labels to the words or word sequences to be extracted. If completely labeled documents are available for training, well-known Markov model techniques can be used to learn such classifiers. In this paper, we consider the more challenging task of learning hidden Markov models (HMMs) when only *partially (sparsely) labeled* documents are available for training. We first give detailed account of the task and its appropriate loss function, and show how it can be minimized given an HMM. We describe an EM style algorithm for learning HMMs from partially labeled data. We then present an active learning algorithm that selects "difficult" unlabeled tokens and asks the user to label them. We study empirically by how much active learning reduces the required data labeling effort, or increases the quality of the learned model achievable with a given amount of user effort.

1 Introduction

Given the enormous amounts of information available only in unstructured or semi-structured textual documents, tools for *information extraction* (IE) have become enormously important (see [5, 4] for an overview). IE tools identify the relevant information in such documents and convert it into a structured format such as a database or an XML document [1]. While first IE algorithms were hand-crafted sets of rules (*e.g.,* [7]), researchers soon turned to learning extraction rules from hand-labeled documents (*e.g.,* [9, 11]). Unfortunately, rule-based approaches sometimes fail to provide the necessary robustness against the inherent variability of document structure, which has led to the recent interest in the use of hidden Markov models (HMMs) [15, 12] for this purpose.

Markov model algorithms that are used for part-of-speech tagging [2], as well as known hidden Markov models for information extraction [12] require the training documents to be labeled *completely*, *i.e.,* each token is manually given an appropriate semantic label. Clearly, this is an expensive process. We therefore concentrate on the task of learning information extraction models from *partially* labeled texts, and develop appropriate EM-style HMM learning algorithms.

We develop an *active* hidden Markov model that selects *unlabeled* tokens from the available documents and asks the user to label them. The idea of *active learning* algorithms (*e.g.,* [3]) is to identify unlabeled observations that would be most useful when labeled by the user. Such algorithms are known for classification (*e.g.,* [3]), clustering [8], and regression [10]; here, we present the first algorithm for active learning of hidden Markov models.

The paper is organized as follows. We give formal account of task and loss function in Section 2, followed by a description of how it can be minimized given an HMM. We then present in detail our EM style algorithm for learning HMMs from partially labeled data (Section 3). We then extend this algorithm to perform active learning in Section 4. We report on our experiments with active hidden Markov models in Section 5.

2 Information Extraction Problem and HMMs

We begin by giving a definition of the task considered in this paper. A document is a sequence of observations, $O = (O_1, \ldots, O_T)$. The observations O_t correspond to the tokens of the document. Technically, each token is a vector of attributes generated by a collection of NLP tools. Attributes may include the word stem, the part of speech, the HTML context, and many other properties of the word, sentence, or paragraph.

The IE task is to attach a semantic *tag* X_i to some of the tokens O_t. Observations can also be left untagged (special tag *none*). An extraction algorithm f maps an observation sequence $O_1, \ldots, O_T$ to a single sequence of tags $(\tau_1, \ldots, \tau_T); \tau_t \in \{X_1, \ldots, X_n, none\}$ (multi-valued assignments would have to be handled by using several IE models, one per label). An IE problem is defined by a joint distribution $P(\tau_1, \ldots, \tau_T, O_1, \ldots, O_T)$ on documents (observation sequences) and their corresponding tag sequences. We can now define the error rate of an extraction algorithm for this IE problem.

Definition 1 (Per-token error). *Assuming that all documents are* finite *token sequences, we can define the* per-token error *of f as the probability of tokens with false tags (Equation 1). We write $f(O)[i]$ for the ith tag returned by f for O.*

$$E_{token}(f) = \int \frac{1}{T} \sum_{t=1}^{T} \delta(f(O)[t], \tau_t) dP(\tau_1, \ldots, \tau_T, O_1, \ldots, O_T) \qquad (1)$$

We can then state the task of *learning* IE models from partially labeled documents as follows.

Definition 2 (Task). *Let D be a distribution $P(\tau_1, \ldots, \tau_T, O_1, \ldots, O_T)$, $\tau_i \in \{X_1, \ldots, X_n, none\}$ over observations and tags, H a space of possible IE models (hypotheses).* **Given** *a set E of example vectors drawn according to D in which however some or all of the τ_i may be hidden, the task is to* **find** *the IE model $f \in H$ to minimize the per-token error.*

Hidden Markov models (see, [13] for an introduction) are a very robust statistical method for structural analysis of temporal data. An HMM $\lambda = (\pi, a, b)$ consists of finitely many states $\{S_1, \ldots, S_N\}$ with probabilities $\pi_i = P(q_1 = S_i)$, the probability of starting in state S_i, and $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$, the probability of a transition from state S_i to S_j. Each state is characterized by a probability distribution $b_i(O_t) = P(O_t | q_t = S_i)$ over observations. In the information extraction context, an observation is a token. The tags X_i correspond to the n target states $S_1, \ldots, S_n$ of the HMM. Background tokens without tag are emitted in all HMM states $S_{n+1}, \ldots, S_N$ which are not one of the target states. We might, for instance, want to convert an HTML phone directory into a database using an HMM with four nodes (labeled *name, firstname, phone, none*). The observation sequence "John Smith, extension 7343" would then correspond to the state sequence (*firstname, name, none, phone*).

How can the IE task be addressed with HMMs? Let us first consider what we would do if we already knew an HMM which can be assumed to have generated the observations (in Section 3, we will then discuss how learn HMMs from partially labeled texts). Given an observation sequence $O = O_1, \ldots, O_T$, which tag sequence should we return to minimize the per-token error? According to Bayes' principle, for each observation O_t, we have to return the tag X_i which maximizes the probability $P(\tau_t = X_i | O)$. This means that we have to identify a sequence of states $q_1, \ldots, q_T$ which maximize $P(q_t = S_i | O, \lambda)$ and return that tag X_i that corresponds to the state S_i for each token O_t.

We briefly describe some elements that we need for our HMM algorithms and refer to [13] for a more detailed discussion. $\alpha_t(i) = P(q_t = S_i, O_1, \ldots, O_t | \lambda)$ is the forward variable; it quantifies the probability of reaching state S_i at time t and observing the initial part $O_1, \ldots, O_t$ of the observation sequence. $\beta_t(i) = P(O_{t+1} \ldots O_T | q_t = S_i, \lambda)$ is the backward variable and quantifies the chance of observing the rest sequence $O_{t+1}, \ldots, O_T$ when in state S_i at time t.

$\alpha_t(i)$ and $\beta_t(i)$ can be computed recursively as in steps 2, 3, and 4 of the *forward-backward* procedure (Table 1). In step 5, we calculate $P(O|\lambda)$, the probability of observation sequence O given the model λ. We can now express the probability of being in state S_i at time t given observation sequence O (called $\gamma_t(i) = P(q_t = S_i | O, \lambda)$) in terms of α and β, as in step 6 of Table 1. Given an observation sequence $(O_1, \ldots, O_T)$ that has been generated by HMM λ, we can minimize the *per-token* error by returning the sequence of states $q_t^* = S_i$ that maximize $\gamma_t(i)$. Table 2 shows how we can assign semantic tags to the tokens of a document such that the per-token error is minimized.

3 Learning HMMs from Partially Labeled Documents

We have now seen how we can use a given HMM for information extraction, but we still have to study how we can learn the parameters λ of an HMM from a set $\mathcal{O} = \{O^{(1)}, \ldots, O^{(n)}\}$ of example sequences. We assume that the user labels some of the *tokens* (not whole sequences) with the appropriate tag. We express this by means of a labeling function $l : \{O_t^{(s)}\} \rightarrow \{X_1, \ldots, X_n, unknown, nolabel\}$. The

Table 1. Forward-Backward algorithm

1. **Input** Observation sequence $(O_1, \ldots, O_T)$, HMM λ.
2. **Let** $\alpha_1(i) = \pi_i b_i(O_1)$. **Let** $\beta_T(i) = 1$.
3. **For** all $t = 1 \ldots T$ and $i = 1 \ldots N$ **Let** $\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right) b_j(O_{t+1})$.
4. **For** all $t = T \ldots 1$ and $i = 1 \ldots N$ **Let** $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$.
5. **Let** $P(O|\lambda) = \sum_{i=1}^{n} \alpha_T(i)$
6. **For** $t = 1 \ldots T$, $i = 1 \ldots N$, **Let** $\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)}$.

Table 2. Information extraction using hidden Markov models

1. **Input** Document $Doc = (w_1, \ldots w_T)$; HMM λ; set of tags $X_1, \ldots X_n$ corresponding to target HMM states $S_1, \ldots, S_n$.
2. **Call** the tokenizer, POS tagger, parser, etc, to generate sequence $O = (O_1, \ldots, O_T)$ of augmented tokens, where each O_t is a vector containing word w_t and attributes. The attributes refer to properties of the word w_t, the sentence, or the paragraph in which w_t occurs.
3. Call Forward-Backward and **Let** $q_t^* = \max_{S_i} \gamma_t(i)$ for all $t = 1 \ldots T$.
4. **For** $t = 1 \ldots T$
   (a) **If** $q_t^* = S_i \in \{S_1, \ldots, S_n\}$ (target state) **Then Output** "$\langle X_i \rangle w_t \langle / X_i \rangle$".
   (b) **Else** (background state) **Output** w_t.

user may label a token ($l(O_t^{(s)}) = X_i$) which means that the HMM must have been in state S_i while emitting $O_t^{(s)}$. Tokens may be labeled as not possessing a tag ($l(O_t^{(s)}) = nolabel$) which implies that the HMM must have been in one of the background states, or may be left unlabeled ($l(O_t^{(s)}) = unknown$) which says nothing about the state.

The Baum-Welch algorithm finds the *maximum likelihood (ML) hypothesis* λ which maximizes $P(\mathcal{O}|\lambda)$. Unfortunately, there is no obvious relation between the ML hypothesis and the Bayes hypothesis that minimizes the expected error given sample $\mathcal{O}$. The ML hypothesis does not take the prior probability of hypotheses $P(\lambda)$ into account. In practice, ad-hoc regularization mechanisms have to be applied in order to avoid finding ML hypotheses that explain the data well but are very unlikely a priori. We use Laplace smoothing with an additive value of 0.0001; maximum entropy is a suitable regularization technique, too [12].

Baum-Welch is an instantiation of the EM algorithm. The main difficulty that this algorithm addresses is that, in order to estimate the transition and emission probabilities, we have to know the state sequence that corresponds to each of the observation sequences. But, unless all observations are labeled with one of the tags, we need to know the transition and emission probabilities in order to calculate the probability of a state sequence, given one of the observation sequences. The algorithm starts with a random model, and then interleaves calculation of the state probabilities (the E-step) with estimation of the transition and emission probabilities based on the calculated state probabilities (the M-step). This bootstrapping procedure converges towards a stable (at least local) optimum with small error rate. We will only elaborate on those elements of the Baum-Welch algorithm that are required to explain our modifications; we refer the reader to [13] for detailed explanations.

Two more variables are needed to define the algorithm; we have to modify their definition and introduce a third to adapt the algorithm to partially labeled sequences. $\xi_t(i,j)$ is the probability of a transition from S_i to S_j at time t. When O_t and O_{t+1} are unlabeled, it is defined in Equation 2 and can be calculated as in Equation 4. In Equation 3, we split ξ into two parts; the right hand side of Equation 3 is equal to our definition of $\gamma_t(i)$. In Equation, 5, we factor this definition of $\gamma_t(i)$ and introduce a new term $\gamma'_t(i,j)$ (defined in Equation 6) for the residual of Equation 5.

$$\begin{aligned}\xi_t(i,j) &= P(q_t = S_i, q_{t+1} = S_j|O,\lambda) &(2)\\ &= P(q_t = S_i|O,\lambda)P(q_{t+1} = S_j|q_t = S_i, O, \lambda) &(3)\end{aligned}$$

$$\begin{aligned}\xi_t(i,j) &= \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} &(4)\\ &= \left(\frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)}\right)\left(\frac{a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\beta_t(i)}\right) = \gamma_t(i)\gamma'_t(i,j) &(5)\end{aligned}$$

$$\gamma'_t(i,j) = P(q_{t+1} = S_j|q_t = S_i, O, \lambda) = \left(\frac{a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\beta_t(i)}\right) \quad (6)$$

Let us now consider the case in which a token is labeled X_k. In this case, we have to force $\gamma_t(i)$ to one for $i = k$ and to zero for all other i (step 3(c)iii of Table 3). Similarly, $\gamma'_t(i,j) = 1$ if O_{t+1} is labeled X_j and zero for other indices (step 3(c)vi). When $l(O_t) = nolabel$, then we we know that the HMM cannot be in any of the target states. In this case, we have to set these probabilities to zero and renormalize (step 3(c)ii for γ and step 3(c)v for γ'). Table 3 shows the resulting Baum-Welch algorithm which maximizes the likelihood of the token sequences while obeying the constraints imposed by the manually added labels.

Theorem 1. *When the documents are* completely unlabeled *($l(O_t^{(s)} = unknown$ for all t,s), then $P(\mathcal{O}|\lambda)$ increases at each iteration or stays constant in which case it has reached a (local) maximum. When all documents are* completely labeled *($l(O_t^{(s)} = X_i$ for some i and all t,s), then the algorithm stabilizes after the first iteration and λ maximizes $P(l(O_1^1),\ldots,l(O_{T_m}^{(m)})|\lambda)$, the likelihood of the labels.*

4 Active Revision of Hidden Markov Models

Unlabeled documents can be obtained very easily for most information extraction problems; only labeling tokens in a document imposes effort. An active learning

Table 3. Baum-Welch Algorithm for partially labeled observation sequences

1. **Input** Set $\mathcal{O}$ of m token sequences $\mathcal{O} = \{(O_1^{(1)}, \ldots, O_{T_1}^{(1)}), \ldots, (O_1^{(m)}, \ldots, O_{T_m}^{(m)})\}$. Labeling function $l : \{O_t^{(s)}\} \rightarrow \{X_1, \ldots, X_n, \mathit{unknown}, \mathit{nolabel}\}$. Number of states N. Optionally an initial parameter set λ.
2. **If** no initial model λ is provided **Then** initialize (π, a, b) at random, but making sure that $\sum_{i=1}^{N} \pi_i = 1$, $\sum_o b_i(o) = 1$ (for all i), $\sum_{j=1}^{N} a_{ij} = 1$ (for all i), and that $0 < p < 1$ for all probabilities $p \in \{\pi, a, b\}$.
3. **Repeat**
   (a) **For** all $s = 1 \ldots m$ Call the forward-backward procedure to calculate the $\alpha_t^{(s)}(i)$, $\beta_t^{(s)}(i)$, and $\gamma_t^{(s)}(i)$.
   (b) **For** all $s = 1 \ldots m$ **Let** $P(O^{(s)}|\lambda) = \sum_{i=1}^{N} \alpha_T^{(s)}(i)$.
   (c) **For** $s = 1 \ldots m$, $t = 1 \ldots T_s$, and $i = 1 \ldots N$ **Do**
      i. **If** $l(O_t^{(s)}) = \mathit{unknown}$ **Then Let** $\gamma_t(i) = \frac{\alpha_t^{(s)}(i)\beta_t^{(s)}(i)}{P(O^{(s)}|\lambda)}$.
      ii. **Else If** $l(O_t^{(s)}) = \mathit{nolabel}$ **Then Let** $\gamma_t^{(s)}(i) = \frac{\alpha_t^{(s)}(i)\beta_t^{(s)}(i)}{\sum_{j=n+1}^{N} \alpha_t^{(s)}(j)\beta_t(j)}$ for $i > n$, and 0 for all $i \leq n$.
      iii. **Else Let** $\gamma_t^{(s)}(i) = 1$ for $l(O_t^{(s)}) = X_i$ and 0 for all other i.
      iv. **If** $l(O_{t+1}^{(s)}) = \mathit{unknown}$ and $t < T_s$ **Then Let** ${\gamma'}_t^{(s)}(i,j) = \frac{a_{ij} b_j(O_{t+1})\beta_{t+1}^{(s)}(j)}{\beta_t^{(s)}(i)}$ for all $j = 1 \ldots N$.
      v. **Else If** $l(O_{t+1}^{(s)}) = \mathit{nolabel}$ and $t < T_s$ **Then Let** ${\gamma'}_t^{(s)}(i,j) = \frac{a_{ij} b_j(O_{t+1})\beta_{t+1}^{(s)}(j)}{\sum_{k=n+1}^{N} a_{ik} b_k(O_{t+1})\beta_{t+1}^{(s)}(k)}$ for all $j > n$, and 0 for all $j \leq n$.
      vi. **Else If** $t < T_s$ **Then Let** ${\gamma'}_t^{(s)}(i,j) = 1$ for $l(O_{t+1}^{(s)}) = X_j$, and 0 for all other j.
      vii. **If** $t < T_s$ **Then Let** $\xi_t^{(s)}(i,j) = \gamma_t^{(s)}(i){\gamma'}_t^{(s)}(j)$ (frequency of a transition from S_i to S_j at time t).
   (d) **Let** $\pi_i = \frac{1}{m}\sum_{s=1}^{m} \gamma_1^{(s)}$ (expected frequency of starting in state S_i).
   (e) **For** $i = 1 \ldots N$, $j = 1 \ldots N$ **Let** $a_{ij} = \frac{1}{m}\sum_{s=1}^{m} \frac{\frac{1}{m}\sum_{s=1}^{m}\sum_{t=1}^{T_s-1} \xi_t^{(s)}(i,j)}{\frac{1}{m}\sum_{s=1}^{m}\sum_{t=1}^{T_s-1} \gamma_t^{(s)}(i)}$ (expected number of transitions from S_i to S_j / number of transitions from S_i).
   (f) **For** $i = 1 \ldots N$ and all observation values o **Let** $b_i(o) = \frac{1}{m}\sum_{s=1}^{m} \frac{\sum_{t=1}^{T_s} \gamma_t^{(s)} \cdot \delta(O_t^{(s)} = o)}{\sum_{t=1}^{T_s} \gamma_t^{(s)}}$ (expected frequency of observing o when in state S_i; δ returns 1 if $O_t^{(s)} = o$, 0 otherwise).
4. **Until** the probabilities a_{ij} and $b_j(O_t)$ stay nearly constant over an iteration.
5. **Output** parameters $\lambda = (\pi, a, b)$.

approach to utilizing the available amount of user effort most effectively is to select, from the available unlabeled tokens, the "difficult" ones of which to know the labels that would be most interesting.

When our objective is to minimize the *per-token error*, then a Bayes-optimal extraction algorithm has to label each token $O_t^{(s)}$ with the tag X_i that maximizes $P(S_i|O^{(s)},\lambda) = \gamma_t^{(s)}(i)$ (or *none*, if $i > n$). We deviate from this optimal strategy when our parameter estimates λ' differ from the true parameters λ such that some state S_j seems to be most likely although state S_i really is most likely ($\max_i P(q_t = S_i|O,\lambda) \neq \max_i P(q_t = S_i|O,\lambda')$) We can see the difference between the probability of the most likely and that of the second most likely state as the confidence of the state given O. When such confidence values or margins can be defined for instances, then we often see empirically that instances with low margins are most relevant to adjust the hypothesis parameters (*e.g.,* [16, 3]).

In analogy, we define the *margin* $M(q_t|O,\lambda)$ of the token that we read to time t as the difference between the highest and second highest probability for a state (Equation 8).

$$M(q_t|O,\lambda) = \max_i\{P(q_t = S_i|O,\lambda)\} - \max_{j\neq i}\{P(q_t = S_j|O,\lambda)\} \tag{7}$$

$$= \max_i\{\gamma_t(i)\} - \max_{j\neq i}\{\gamma_t(j)\} \tag{8}$$

Intuitively, the margin can be seen as quantifying how "difficult" (low margin) or "easy" (large margin) an example is. Our active HMM learning algorithm (Table 4) first learns an initial model λ_1 from a set of partially labeled documents. It then determines the margins of all tokens and starts asking the user to label those tokens which have a particularly low margin. The Baum-Welch algorithm is restarted, using the previous parameters λ_{k-1} as initial model and adapting λ_k to the new data.

5 Experiments

For our experiments, we generated HMMs with variable numbers of background and target states at random. We used these HMMs to generate unlabeled observation sequences; we label a number of initial tokens drawn at random. We then study how the error developes with the number of additional labels added to the observation sequences according to three strategies. The "random" stragegy is to label randomly drawn observations; the "margin" strategy is to label "difficult" tokens with smallest margins; this corresponds to our active hidden Markov model. As a control strategy ("large margins"), we also try selecting "easy" tokens that have the largest margins. If "margins" is really better than "random", we expect "large margins" to perform worse.

We used three different HMM sizes. The "easy" HMM consists of one background and two target states. Each state emits three out of 20 observations with randomly drawn probabilities. We generated 50 sequences of 20 initially unlabeled observations. The "medium size" HMM possesses 10 nodes, and the

Table 4. Active Revision of hidden Markov Models

---

1. **Input** Set $\mathcal{O}$ of m Sequences $\mathcal{O} = \{(O_1^{(1)}, \ldots, O_{T_1}^{(1)}), \ldots, (O_1^{(m)}, \ldots, O_{T_m}^{(m)})\}$ of tokens; Labeling function $l : O_t^{(s)} \mapsto \{S_1, \ldots, S_n, unknown, nolabel\}$; Number of states N; query parameter n_q.
2. **Call** the Baum-Welch algorithm to determine initial parameters λ_1.
3. **For** $k = 2 \ldots \infty$ **Repeat**
   (a) **For** $s = 1 \ldots m$ **Call** the forward-backward algorithm.
   (b) **For** $s = 1 \ldots m$ and $t = 1 \ldots T_s$ **Let** $M(q_t|O^{(s)}) = \max_i\{\gamma_t(i)\} - \max_{j \neq i}\{\gamma_t(j)\}$.
   (c) **Ask** the user to label the n_q unlabeled tokens $O_t^{(s)}$ ($l(O_t^{(s)}) = nolabel$) with smallest margin $M(q_t|O^{(s)})$. Update the labeling function l.
   (d) **Estimate** the new model parameters λ_k using the Baum-Welch algorithm with initial model λ_{k-1}.
4. **Until** the user gets tired of labeling data.
5. **Return** HMM parameters $\lambda_i = (\pi, a, b)$.

"large" HMM consists of 15 states. The curves in Figure 1 are averages over 50 leaning problems. The initial sample contains only unlabeled tokens (Figures 1a for the easy, 1d for the medium size and 1g for the hard learning problem), labels of 80 (Figures 1b, 1e, and 1h, respectively) and 160 tokens (Figures 1c, 1f, and 1i) drawn at random.

In Figures 1a, 1d, and 1g we see a slight but significant advantage of random token selection over selecting tokens with small margins. Using only difficult tokens from the beginning is not beneficial on average. In the later phase of learning, token selection by small margins gains a small advantage. The benefit of the margin strategy becomes more clearly visible, when the initial sample contains the labels of 80 (Figures 1b, 1e, and 1h) or 160 (Figures 1c, 1f, and 1i) tokens drawn at random, and only from then on tokens with smallest margins are selected. For the small HMM learning problem (when much unlabeled data relative to the problem complexity is available), the bottom line error is reached after about 300 labels under the margin strategy and after 600 labeled tokens under the random strategy.

Using active learning with small margin examples after 70 initial random tokens seems to be most beneficial. In this case (Figure 1b), the base level error is reached after less than 200 examples for active and after 1000 examples regular learning – *i.e.,* five times fewer labels are needed. Choosing only the most "easy" examples (large margins) is clearly a bad strategy in all cases. Our experiments show that, at least for the classes of HMM that we generated, using only difficult low-margin tokens for learning from the beginning results in higher error rates. However, when the training is started by labeling randomly drawn tokens and the active HMM chooses difficult low-margin tokens after that initial phase, then a significant improvement is achiever over regular HMMs that can result in the sufficiency of many times fewer labeled examples.

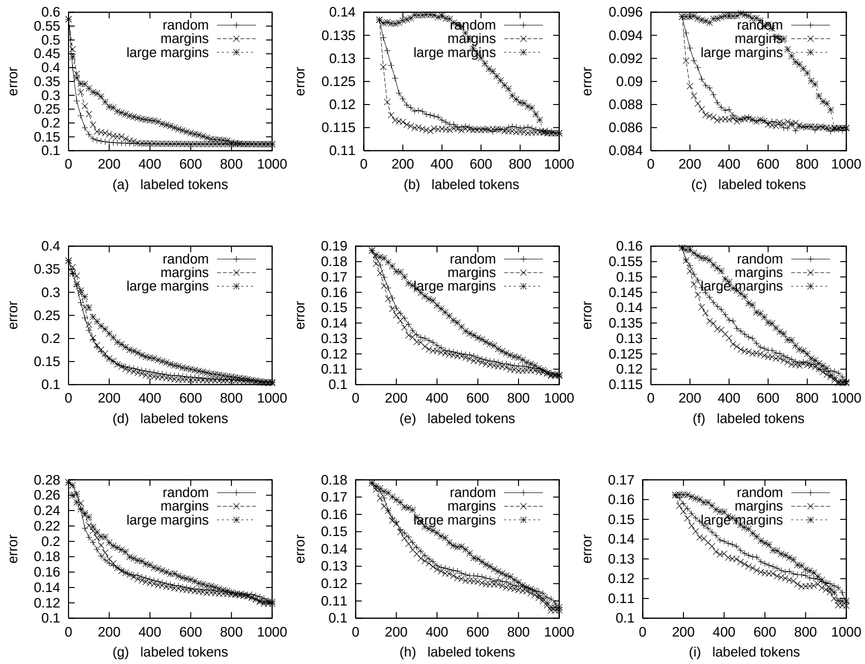

Fig. 1. Error rate of active and regular learning over number of labeled tokens. (a)-(c), easy HMM; initial sample contains (a) no (b) 80 (c) 160 labeled tokens drawn at random. (d)-(f) medium size HMM; initial sample with (d) no (e) 80 (f) 160 initial labels. (g)-(i) large HMM; (g) no (h) 80 (i) 160 initial labels.

6 Discussion and Related Work

We defined a hidden Markov model that operates on partially labeled observation sequences. We defined the margin of tokens as a measure of their difficulty. Our experiments show that it is particularly important to know the labels of difficult examples with low margins. We observed that active HMMs sometimes require three times fewer examples than regular HMMs to achiev a high level of accuracy.

An alternative hidden Markov model for information extraction has been described by [12]. Instead of using the usual distributions a_{ij} and $b_i(O_t)$, the alternative model uses a conditional probability $P(q_{t+1}|q_t, O_{t+1})$. This appears appropriate at first blush but has considerable consequences.

Firstly, the number of probabilities to be estimated is $|Q|^2 \times |O|$ as opposed to $|Q|^2 + |Q| \times |O|$ in our model. Secondly, $P(O|\lambda)$ cannot be computed in the alternative model which renders it impossible to use such HMMs for text classification which seems possible, in principle, in our model. The Baum-Welch algorithm of [12] requires all tokens to be labeled with exactly one state. However, the modifications proposed here could be applied to the alternative model as well. See [14] for a more detailed comparison of the two models.

A restriction of the model discussed here is that it is not possible to enclose a token sequence in nested tags. Only when all tokens are labeled with a tag that corresponds to exactly one state, then cascaded Markov models [2] solve this problem. For the more general case of partially labeled documents, the hierarchical hidden Markov model [6] has to be adapted to partially labeled token sequences, analogously to our adaptation of the standard HMM.

References

1. T. Berners-Lee. Semantic web road map. Internal note, World Wide Web Consortium, 1998.
2. T. Brants. Cascaded markov models. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics*, 1999.
3. D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.
4. Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew K. McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence*, 118(1-2):69–113, 2000.
5. L. Eikvil. Information extraction from the world wide web: a survey. Technical Report 945, Norwegian Computing Center, 1999.
6. S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, 32:41–64, 1998.
7. Ralph Grishman and Beth Sundheim. Message understanding conference - 6: A brief history. In *Proceedings of the International Conference on Computational Linguistics*, 1996.
8. Thomas Hofmann and Joachim M. Buhmann. Active data clustering. In *Advances in Neural Information Processing Systems*, volume 10, 1998.
9. N. Hsu and M. Dung. Generating finite-state transducers for semistructured data extraction from the web. *Journal of Information Systems, Special Issue on Semistructured Data*, 23(8), 1998.
10. Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems*, volume 7, pages 231–238, 1995.
11. N. Kushmerick. Wrapper induction: efficiency and expressiveness. *Artificial Intelligence*, 118:15–68, 2000.
12. Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.
13. L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.
14. T. Scheffer, S. Hoche, and S. Wrobel. Learning hidden markov models for information extraction actively from partially labeled text. Technical report, University of Magdeburg, 2001.
15. Kristie Seymore, Andrew McCallum, and Roni Rosenfeld. Learning hidden markov model structure for information extraction. In *AAAI'99 Workshop on Machine Learning for Information Extraction*, 1999.
16. V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.