Proceedings of the ICML-Workshop on Text Learning. 2002.

Text Classification Beyond the Bag-of-Words Representation

Tobias Scheffer SCHEFFER@IWS.CS.UNI-MAGDEBURG.DE
University of Magdeburg, FIN/IWS, Universitätsplatz 2, 39106 Magdeburg, Germany

Stefan Wrobel WROBEL@AIS.FHG.DE
FhG AiS, Schloß Birlinghoven 53754 Sankt Augustin, Germany
University of Bonn, Informatik III, Römerstr. 164, 53117 Bonn, Germany

Abstract

Most known text classifiers represent documents as bags of words and process documents as a whole. In practice, *e.g.,* when handling long documents with sections, it would be useful to have algorithms that can seamlessly switch from labeling documents to labeling regions, or individual tokens, and be able to account for the sequence and context of words at least in a limited way. We discuss how both of these issues can be addressed by treating text classification as an instance of a generalized token labeling problem which we define. We show that Hidden Markov models (HMMs) cover this range of labeling problems and point out that in our setting a HMM is in fact a direct generalization of the naive Bayes classifier and allows limited context to be treated. We derive algorithms for the associated learning problems that cover the full range from completely labeled data, to labels that impose only weak constraints on the possible state sequences. In experiments, we demonstrate the clear advantages of HMMs for classification.

1. Introduction

With the rapid growth of information available online in textual documents, algorithms that analyze the content of documents have become enormously important. Text classification is the problem of mapping a document to a set of topics. In order to learn classifiers for text, one often employs the so-called *bag-of-words* representation, where each document is represented as a feature vector counting the number of occurrences of different words as features. One can then use learning algorithms such as the naive Bayes algorithm (*e.g.,* (Langley et al., 1992)), which is one of this simplest and yet a very effective and efficient text classifier.

While successful in many applications, representing entire documents as bag-of-words feature vectors limits the power of what can be learned and excludes practically relevant problem settings. First of all, many text documents consist of several sections that may well be about different topics. Such documents are difficult to deal with in the bag-of-words representation, since the counts of words are summed up across the different sections. Secondly, the bag-of-words representation does not take into account the context nor the sequence of words, and thus cannot distinguish between e.g. "I have no objections, thanks" and " No thanks, I have objections."

In this paper, we show that both of the above problems can be addressed by treating the document classification problem as an instance of a generalized *token labeling* problem inspired by information extraction (*e.g.,* (Kushmerick et al., 1997)). We precisely define this setting, and then describe how *Hidden Markov models* (HMMs) for this generalized task can be learned even from *partially labeled* data. We show that in the context of our generalized task, HMMs in fact constitute a generalization of the popular Naive Bayes algorithm, and that the number of allowed HMM states is a simple way of controlling how strongly word ordering and context are taken into account. We present empirical experiments with our approach which clearly show the advantages when compared to a Naive Bayes classifier using the bag-of-words representation.

The paper is organized as follows. We discuss our problem setting in Section 2 and review hidden Markov models in Section 3. In Section 4, we focus on learning HMM parameters from partially labeled data. Labels impose constraints on the possible states in which a token may have been emitted. In general, the constraints do not identify a state uniquely; we present a derivation of a new forward-backward algorithm that handles partial labels in a mathematically sound way. We show how HMMs are applied to text classification in Section 5 and discuss the relation between HMMs and naive Bayes. We present experiments in Section 6 and discuss future work on information extraction in Section 7.

2. Problem Setting

Let us first define the token labeling problem that we focus on. We assume that there is an (unknown) stationary process $P((O_1, c_1), \ldots, (O_T, c_T))$ that generates sequences of tokens O_t with corresponding class labels c_t. We are given a training set $\{((O_1, c_1), \ldots, (O_T, c_T)), \ldots\}$ of independently drawn sequences, each sequence governed by $P((O_1, c_1), \ldots, (O_T, c_T))$. The learning problem is to find an extraction function $f : (O_1, \ldots, O_T) \mapsto (c_1^f, \ldots, c_T^f)$ assigning a class label to each token such that the error criterion is minimized.

The goal of *text classification* is to map a whole document to a given set of semantic categories. Text classification can be seen as a token labeling problem with the constraint that $c_1 = \ldots = c_T$; *i.e.,* all token labels within a single document are equal.

Classification accuracy $acc(f)$ is a performance criterion used rarely in text classification. Since class distributions are often skewed, precision and recall are more popular criteria for binary problems ($c \in \{0, 1\}$). Precision $prec(f)$ refers to the probability of a correct classification given that $f(O) = 1$. Recall $rec(f)$ is the probability of $f(O) = 1$ given that the correct class label of O is 1.

Token labeling is also the core of the *information extraction* problem. Since class labels are generated for individual tokens rather than for documents as a whole, it is natural to define the performance of information extraction algorithms with respect to individual tokens as well. We define the *per-token accuracy* acc_{token} as the probability of a correct class label for a token with respect to the underlying stationary process. Accordingly, we define $prec_{token}(f)$ and $rec_{token}(f)$, precision and recall on a per-token basis.

3. Hidden Markov Models

Hidden Markov models (HMMs) (see (Rabiner, 1989), for an introduction) are a very robust statistical method for analysis of sequential data. A HMM $\lambda = (\pi, a, b)$ consists of finitely many states $V = \{S_1, \ldots, S_N\}$. The state that the HMM is in at time t is denoted as q_t (q_t is the random variable, $S_1, \ldots, S_N$ its possible values). At time step t, the HMM generates an observation O_t. $\pi_i = P(q_1 = S_i|\lambda)$ denotes the probabilities of starting in state S_i; $a_{ij} = P(q_{t+1} = S_j|q_t = S_i, \lambda)$ is the probability of a transition from state S_i to S_j. Each state is characterized by a distribution $b_i(O_t) = P(O_t|q_t = S_i, \lambda)$ over observations. At each time step t, the state changes according to a and an observation is generated according to b.

The states through which the model passes are invisible to the outside, only the observation sequence which is a probabilistic function of the state sequence is known; hence the name hidden Markov model. HMMs are useful to model how underlying events probabilistically induce surface events and thus have been found suitable to model natural language.

Underlying HMMs is the Markov assumption which says that the states q_t summarize all past information relevant for q_{t+1}, and that the probability of an observation only depends on the current state.

HMM algorithms are described in terms of variables α, β, and γ. Most relevant for our further discussion is $\gamma_t(i) = P(q_t = S_i|O_1, \ldots, O_T, \lambda)$, the probability that token O_t has been emitted in state S_i, given the whole observation sequence $O = (O_1, \ldots, O_T)$. $\gamma_t(i)$ is decomposed into $\alpha_t(i)$ (calculated in a forward pass over the observation sequence) and $\beta_t(i)$ (calculated in a backward pass); the whole process is referred to as the forward-backward algorithm. The known decomposition (Rabiner, 1989) is not well suited to handle constraints on $\gamma_t(i)$ which are imposed by labels. We will discuss a new γ-decomposition in Section 4.

In order to learn HMM parameters λ from data, the Baum-Welch algorithm, an instance of EM, iterates two steps. In the E-step, it calculates the state probabilities of all tokens. Based on the state probabilities, in the M-step the transition probabilities and state dependent observation probabilities are estimated.

Learning HMM parameters simplifies if, for each token, the state it has been emitted in is known. This is the case when the data is completely labeled and every label corresponds to exactly one state; *e.g.,* in part-of-speech tagging (Brants, 1999). In this case, the transition probabilities a_{ij} can be estimated by simply counting how frequently each state S_j succeeds state S_i in the training data (emission probabilities $b_i(O_t)$ are counted accordingly).

4. Weakly Labeled Data

When learning token labeling hypotheses, we are given a set of example documents $O = (O_1, \ldots, O_T)$. Documents O are labeled with two or more mutually exclusive classes $(c_{i_1}, \ldots, c_{i_T})$; in text classification, all labels c_{i_t} of a single document O are equal. To every class c_i we associate a set of states V_i; tokens of class c_i are emitted by the HMM in states V_i. For each observation O_t in a document O, let $\sigma_t = V_i$ be the set of one or more states consistent with the label c_i. If O is missing a label, then $\sigma_t = V$ is consistent with all nodes.

Starting with random parameters λ, we employ the Baum-Welch algorithm to find parameters λ by iterating the following steps.

1. For all documents O, invoke the forward-backward algorithm to calculate all $\gamma_t(i)$ probabilities of state S_i at time t given the document O and its labels.
2. For all documents O, time steps t, and states S_i and S_j, calculate $\xi_t(i,j)$, the probability of a transition from S_i to S_j at time t.
3. Based on the estimated γ and ξ, estimate the HMM parameters π_i, a_{ij}, and $b_i(O_t)$ by counting the expected number of times the HMM starts in state S_i, transits from S_i to S_j, and outputs O_t in S_i, accordingly.

The Baum-Welch algorithm is well-known; our contribution here is a new forward-backward algorithm that identifies $\gamma_t(i)$ and $\xi_t(i,j)$ taking the constraints σ_t into account.

The known forward-backward procedure calculates $P(q_t = S_i|O_1, \ldots, O_T, \lambda)$. In order to take labels into account, we instead have to calculate $\gamma_t(i) = P(q_t = S_i|O_1, \ldots, O_T, \sigma_1, \ldots, \sigma_T, \lambda)$, the state probability given O and given the states σ_t that are consistent with the labels. Note that a label attached to token O_t will generally have non-local effects on the state probability $\gamma_t(i)$ for other time steps t because it "blocks" certain state transition sequences. We now have to find a decomposition of our new $\gamma_t(i)$ which exhibits the recursive structure needed to evaluate $\gamma_t(i)$ efficiently by dynamic programming. Equation 2 illustrates our decomposition.

$$\begin{aligned}
\gamma_t(i) &= P(q_t = S_i|\sigma_1, \ldots, \sigma_T, O_1, \ldots, O_T) && (1)\\
&= cP(q_t = S_i|O_1, \ldots O_{t-1}, \sigma_1, \ldots, \sigma_{t-1}) \\
&\quad P(O_t, \ldots, O_T, \sigma_t, \ldots, \sigma_T|q_t = S_i) && (2)
\end{aligned}$$

The desired recursive structure is presented in Theorems 1 and 2.

Definition 1 *We define $\alpha_t(i)$ as the probability of arriving in state S_i at time t given an initial observation sequence and labels (Equation 3); $\beta_t(i)$ is the probability of encountering the remaining observations and labels given the state at time t (Equation 4). $\gamma_t(i)$ is the probability of state S_i at time t given a partially labeled observation sequence, and $\xi_t(i,j)$ is the probability of a transition from S_i to S_j at time t.*

$$\begin{aligned}
\alpha_t(i) &= P(q_t = S_i|O_1, \ldots O_{t-1}, \sigma_1, \ldots, \sigma_{t-1}, \lambda) && (3)\\
\beta_t(i) &= P(O_t, \ldots, O_T, \sigma_t, \ldots, \sigma_T|q_t = S_i, \lambda) && (4)\\
\gamma_t(i) &= P(q_t = S_i|O_1, \ldots, O_T, \sigma_1, \ldots, \sigma_T, \lambda) && (5)\\
\xi_t(i,j) &= P(q_{t+1} = S_j, q_t = S_i|O, \sigma_1, \ldots, \sigma_T, \lambda) && (6)
\end{aligned}$$

For the proofs of the following Theorems, we refer the reader to our technical report (Scheffer & Wrobel, 2002).

Theorem 1 *$\alpha_t(i)$ can be computed recursively as in Equations 7 and 8.*

$$\begin{aligned}
\alpha_1(i) &= \pi_i && (7)\\
\alpha_{t+1}(j) &= \frac{\sum_{i\in\sigma_t} a_{ij} b_i(O_t)\alpha_t(i)}{\sum_{k\in\sigma_t} b_k(O_t)\alpha_k(i)} && (8)
\end{aligned}$$

Theorem 2 *$\beta_t(i)$ can be computed recursively as in Equations 9 and 10.*

$$\beta_T(i) = \begin{cases} b_i(O_T) & [\ q_T \in \sigma_T] \\ 0 & [\ \text{else}] \end{cases} \quad (9)$$

$$\beta_t(i) = \begin{cases} b_i(O_t) \sum_{j\in\sigma_{t+1}} a_{ij}\beta_{t+1}(j) & [\ q_t \in \sigma_t] \\ 0 & [\ \text{else}] \end{cases} \quad (10)$$

Theorem 3 *$\gamma_t(i)$ can be computed as in Equation 11.*

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j\in\sigma_t} \alpha_t(j)\beta_t(j)} \quad (11)$$

Theorems 1, 2, and 3 say that it is possible to determine $\gamma_t(i)$ in time linear in the observation length. The difference between our modified forward-backward algorithm which calculates $\gamma_t(i)$ and the original one (Rabiner, 1989) lies in the equations which calculate $\alpha_t(i)$, $\beta_t(i)$, and $\gamma_t(i)$ (Theorems 1, 2, and 3).

1. **For** $t = 1 \ldots T$, calculate the $\alpha_t(i)$ according to Theorem 1.
2. **For** $t = T \ldots t$, calculate the $\beta_t(i)$ according to Theorem 2 and, in the same pass, calculate the $\gamma_t(i)$ according to Theorem 3.

Finally, we have to show how we can calculate ξ relative to the new α and β.

Theorem 4 *$\xi_t(i,j)$ can be calculated as in Eq. 12*

$$\xi_t(i,j)=\begin{cases} \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{k\in\sigma_t}\sum_{\ell\in\sigma_{t+1}}\alpha_t(k)a_{k\ell}b_{t+1}(O_\ell)\beta_{t+1}(\ell)} & [i\in\sigma_t] \\ 0 & [\text{else}] \end{cases} \tag{12}$$

In case no labels are available (all $\sigma_t = V$), the $\gamma_t(i) = P(q_t = S_i|O_1,\ldots,O_T,\sigma_1,\ldots,\sigma_T,\lambda)$ calculated by our forward-backward algorithm are equal to the $P(q_t = S_i|O_1,\ldots,O_T,\lambda)$ calculated by the traditional forward-backward algorithm. While the regular Baum-Welch algorithm finds parameters λ to maximize $\prod_k P((O_1^{(k)},\ldots,O_{T_k}^{(k)}|\lambda)$, our modified Baum-Welch takes the constraints imposed by the labels into account.

Theorem 5
The modified Baum-Welch algorithm will converge to a parameter set $\lambda = (\pi, a, b)$ which is at least a local maximum of $\prod_k P(O_1^{(k)},\ldots,O_{T_k}^{(k)},\sigma_1^{(k)},\ldots,\sigma_{T_k}^{(k)}|\lambda)$.

The proof of Theorem 5 follows from the convergence Theorem of EM and our definition of $\gamma_t(i)$.

In the special case that all observations in the training set are labeled and the labels are consistent with only a single state ($|\sigma_t^{(k)}| = 1$), Baum-Welch converges after the first iteration.

5. Text Classification

Let us now see how we can apply a HMM with parameters λ to classify a new document $(O_1,\ldots,O_T)$. Theorem 6 says that $\sum_{j\in V_i}\gamma_1(j)$ is the optimal decision function for class c_i. V_i is the set of states that we have associated with class c_i. $\gamma_1(j)$ is calculated by the forward-backward algorithm where all $\sigma_t = V$ (during the application phase, no labels constraining the possible states are available).

Theorem 6 *Given a HMM λ which encodes the joint distribution $P(O,c)$, where documents of class c_i are emitted in states V_i. Let O be a document drawn according to $P(O,c)$. $P(c = c_i|O) = \sum_{j\in V_i}\gamma_1(j)$; i.e., $\text{argmax}_i\sum_{j\in V_i}\gamma_1(j)$ is a Bayes optimal classifier which maximizes $acc(f)$.*

For text classification, each training document is attached one single label; in our notation, we assign the document label to each token of that document. Hence, all tokens of one document have equal class labels and the transition probability between states corresponding to distinct classes is zero – *i.e.,* $a_{kl} = 0$ if $k\in V_i$ and $l\in V_j$ and $i\neq j$. Hence, $\sum_{j\in V_i}\gamma_t(j)$ stays constant over all t.

In order to achieve a desired precision-recall balance for class c_i, our decision function has to be compared against a threshold; increasing the threshold increases precision for that class and decreases recall.

We have so far considered the case that two or more mutually exclusive classes are distinguished by a single classifier. In many application scenarios, classes are overlapping and each example is a positive example for some classes, and negative for others. When a document can belong to any subset of n overlapping classes, n binary classifiers have to be learned.

A binary classifier has to model both the class and its complement; the complementary class can be rather heterogenous because its elements come from many different categories. In order to distribute equally labeled examples among multiple states, we have so far employed the Baum-Welch algorithm. The MC_1 heuristic assigns all positive examples to one single state. For the negative examples, it uses as many states as there are binary classification tasks. Which state a negative example is assigned to depends on which class it is a positive example for (if it is a positive example for multiple classes, a state is drawn at random). The HMM parameters are then estimated in one step by simply counting relative frequencies.

Naive Bayesian classifiers assume independence for words in documents (given the class): $P(O_t|O_1,\ldots,O_{t-1},c_i) = P(O_t|c_i)$. Under this assumption, the optimal decision $\text{argmax}_i P(c_i|O)$ simplifies to $\text{argmax}_i\left(\prod_{1\leq t\leq T}P(O_t|c_i)\right)P(c_i)$. Naive Bayesian classifiers obtain estimates $\hat{P}(O_t|c_i)$ and $\hat{P}(c_i)$ by counting instances of word O_i and class c_i, accordingly. This is identical to a HMM with exactly one state per class.

Theorem 7 *Let λ be HMM parameters and let $V_1,\ldots,V_n$ with $|V_i| = 1$ for all i be the states corresponding to classes c_1 through c_n. Then λ is equivalent to a naive Bayesian classifier – i.e.,*

$$\text{argmax}_i\sum_{j\in V_i}\gamma_1(j) = \text{argmax}_i\left(\prod_{1\leq i\leq T}\hat{P}(O_i|c_i)\right)\hat{P}(c_i)$$

Sketch of Proof. For text classification, all labels of tokens within one single document are equal; hence $a_{ij} = 0$ for $i\neq j$. Together with the recursive structure of $\alpha_t(i)$ (Theorem 1) this implies that $\alpha_t(i)$, $\beta_t(i)$, and therefore also $\gamma_t(i)$ are constant over t. Together with Theorem 3 ($\gamma_t(i)$ is the normalized product of $\alpha_t(i)$ and $\beta_t(i)$) we can conclude that $\gamma_1(i) = \gamma_T(i) = \alpha_T(i)\beta_T(i) = \alpha_T(i)$. Therefore, our decision function

$\sum_{j \in V_i} \gamma_1(j)$ resolves to $\sum_{j \in V_i} \pi_j b_j(O_1) \ldots b_j(O_T)$. As $|V_i| = 1$ this equals $\hat{P}(c_i) \prod_{t=1}^{T} \hat{P}(O_t|c_i)$ which is the decision function of naive Bayes. ∎

HMMs with multiple states per class relax the naive Bayesian independence assumption since such HMMs possess a fixed size memory that can store information on past observations. The HMM independence assumption $P(O_t|O_1, \ldots, O_{t-1}, q_t) = P(O_t|q_t)$ says that observations are independent given the memory state, rather than only given the class label. When there are multiple states per class c_i, then $P(O_t|O_1, \ldots, O_{t-1}, q_t) = P(O_t|q_t)$ is a weaker assumption than $P(O_t|O_1, \ldots, O_{t-1}, c_i) = P(O_t|c_i)$; we can eliminate the independence assumption by setting the number of states equal to the document length.

6. Empirical Studies

Let us first look at a "sanity check" problem. We try to distinguish positive examples including the string "I have no objections, thanks" from negative examples which include the string "No thanks, I have objections." Positives and negatives are indistinguishable in a bag-of-word representation. Figure 1 shows that a HMM with three or more states per class solves this problem perfectly while a naive Bayesian classifier has an error rate of 0.5.

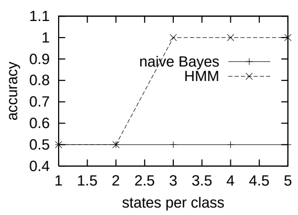

Figure 1. Confirmation versus rejection

In our next set of experiments, we study binary decision problems and compare the accuracy of a HMM with one state per class (equivalent to a naive Bayesian classifier) to a HMM with multiple states (which replaces the naive Bayesian independence assumption by the weaker Markov assumption). We use the "20 newsgroups" data set with 19997 training examples. Roughly 60,000 words occur in the documents which we reduce to 20,000 stems; we always use 80% of the data for training and 20% for testing. We remove the headers from all postings (the newsgroups are listed in the header which makes the problem less thrilling). With removed headers, we obtain an accuracy of 86.9% over all 20 newsgroups using a single HMM with 20 states. We extract binary decision problems. Building blocks of five classes, we discriminate each of the classes against its four neighbors.

We first compare three sets of precision recall curves. We learn a HMM with one state for the positives and one for the negative examples (a naive Bayes classifier). Next, we learn a HMM with one positive and four negative states, using the Baum-Welch algorithm to train the parameters for the subset of four nodes corresponding to the negative class. Finally, we learn a HMM with one node for the positive examples and 4 nodes for the negatives, using the class membership to assign each negative example to exactly one of the 4 background states (the MC_1 heuristic).

Figure 2 clearly shows that the HMMs with 5 states are uniformly superior to HMMs with 2 states. The rejection class of each of the binarized decision problems is intrinsically a mixture of 4 message categories. Modeling the negative class by four states and assuming independence only in each state locally seems to be more appropriate than modeling the union of four classes in a single state with a naive Bayesian classifier.

Figure 2 also shows the resulting precision recall curves using four nodes for the negative examples and using the Baum-Welch algorithm rather than the MC_1 heuristic. There is no clear result on whether one single state or four states with Baum-Welch for the negatives is better. Unfortunately, although there is a classifier with five states that is much more accurate than the one with two states (the MC_1 heuristic returns it), the Baum-Welch algorithm does not find its parameters.

In the next set of experiments, we generate long and heterogenous documents by pasting 10 randomly drawn postings together. We measure per-token precision and recall. We train an HMM and measure its performance on a holdout set of joined documents with up to 10 distinct classes. We compare the precision recall curves to those of a naive Bayesian classifier (a HMM with two states). The naive Bayesian classifier can only predict one single class per document, so we can expect its performance to be considerably lower.

In order to determine the per-token precision and recall of an HMM with optimal segmentation, we set the class transition probabilities τ_{ij} of the trained HMM to zero for $i \neq j$ and provide the HMM with individual documents.

Figure 3 shows that the HMM which has to detect the class transitions performs worse than a classifier which receives perfectly segmented documents. In particular, the HMM seems to have difficulties

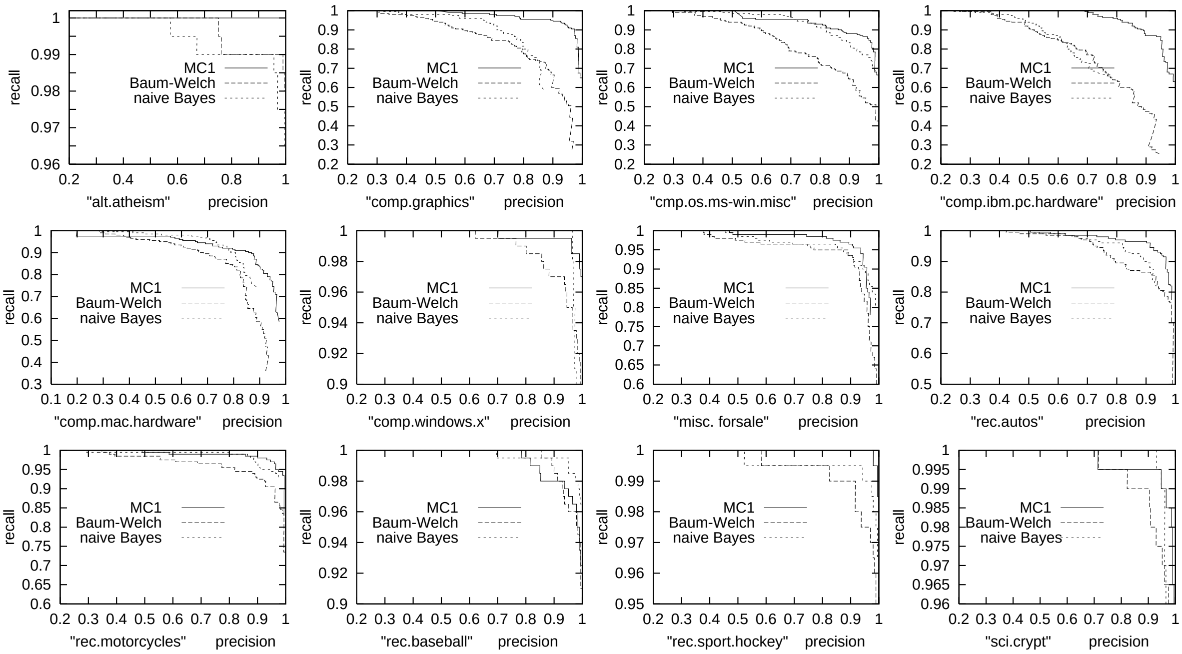

Figure 2. Precision recall curves for the 20 newsgroups problem

in finding the exact location of transitions of similar classes (*e.g.,* between "comp.sys.ibm.pc.hardware" and "comp.sys.mac.hardware"). The difference between the HMM and a classifier with perfect segmentation is smaller for easier classes such as "misc.forsale" or "alt.atheism".

7. Further Work: Information Extraction

We have generalized the text classification problem by defining the token labeling problem and providing a solution to this problem. A second interesting special case of the token labeling problem is the information extraction problem. Training data for information extraction problems consists of sequences of tokens. Each token O_t can be labeled; σ_t is the set of one or more states consistent with the labeling. An unlabeled token is consistent with all states ($\sigma_t = V$) whereas a labeled token may only be consistent with some or only one token.

After determining the σ_t for all examples O, we invoke the Baum-Welch algorithm, using our modified definition of γ which takes the labels into account.

Once we have found parameters λ, and want to process a new document O, we apply the forward-backward algorithm to determine the $\gamma_t(j)$. During the application phase, no labels are available and hence $\sigma_t = V$.

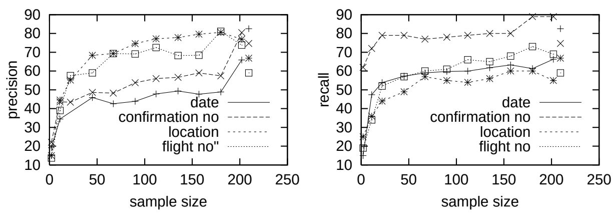

Figure 4. Extraction of flight details from emails

Theorem 8 *Given a HMM λ that encodes a joint distribution $P((O_1, c_{i_1}), \ldots, (O_T, c_T))$ where tokens of label c_i are emitted in states V_i. For all tokens O_t, $\sum_{j \in V_i} \gamma_t(j) = P(class_t = c_i|O)$; i.e.,* $\operatorname{argmax}_i \sum_{j \in V_i} \gamma_t(j)$ *is a Bayes-optimal classification of a token O_t that maximizes $acc_{token}(f)$.*

Again, a desired precision recall balance can be achieved by comparing the decision function $\sum_{j \in V_i} \gamma_t(j)$ to a threshold parameter. Remarkably, Theorem 8 says that, when the task is to minimize the probability of misclassified tokens, class decisions should be based on $\gamma_t(j)$ calculated by the forward-backward algorithm, rather than on $\delta_t(j)$ determined by the Viterbi algorithm.

Figure 4 shows precision and recall for the problem of extracting flight details from flight confirmation emails sent to an airline. The HMM has one state for each of "date", "location", "flight number" and "confirmation number"; the HMMs have five background states

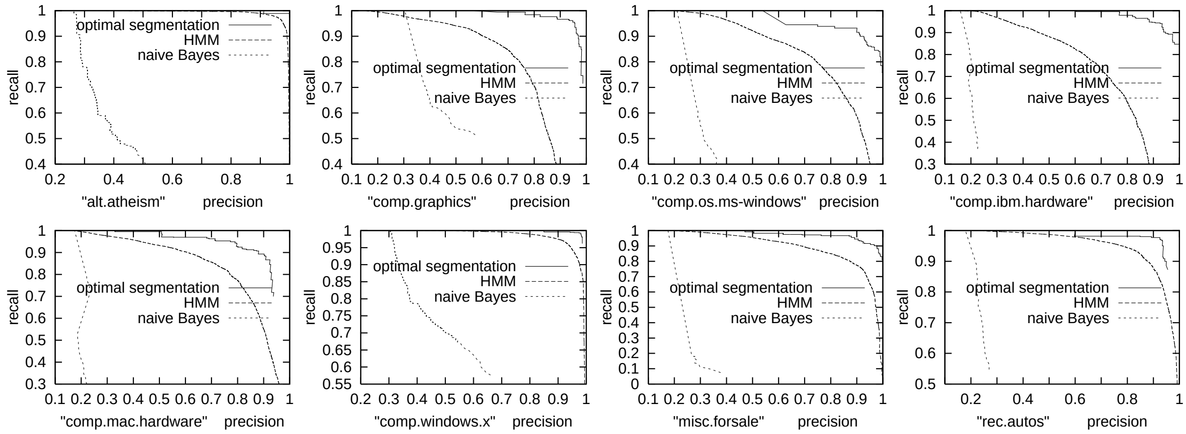

Figure 3. Precision recall curves for unsegmented postings of the 20 newsgroups problem

which emit unlabeled tokens that are not to be extracted. In the example emails, tokens belonging to one of these five classes are labeled, all other tokens are unlabeled. For this problem, observations are vectors consisting of the token and general attributes that indicate whether the token is numeric, whether it starts with a capital letter, and similar information. In order to estimate $b_i(O_t)$, we assume independence for these attributes.

8. Discussion and Related Work

Text classification can be seen as a special case of the token labeling problem. Probabilistic sequence models such as HMMs are able to assign semantic labels to tokens, regions, or whole documents. While previous work has focused on example sequences with completely known state sequences (McCallum et al., 2000) or examples with unknown state (McCallum & Nigam, 1998), we derived an algorithm that handles constraints on states that are imposed by labels.

Durbin et al. (Durbin et al., 1998) handle partially labeled data ad-hoc by setting $\alpha_t(i)$ and $\beta_t(i)$ (in the original definitions of (Rabiner, 1989)) to zero for states that are not consistent with the labels. The resulting $\gamma_t(i)$ differ from our $\gamma_t(i)$ which are the probabilities of state q_t *given* the observation sequence and its labels. HMMs have also been applied to text segmentation (Beeferman et al., 1999).

HMMs relax the independence assumption of naive Bayesian classifiers. Instead of assuming independence of all words given the class — $P(O_t|O_1,\ldots,O_{t-1},c_i) = P(O_t|c_i)$ — the Markov assumption says that observations are independent given the state of the fixed size memory of the HMM – $P(O_t|O_1,\ldots,O_{t-1},q_t) = P(O_t|q_t)$. When each class is represented by a single HMM state, then both assumptions are identical. Hence, a HMM with one state per class is a naive Bayesian classifier. In case of multiple states per class, the Markov assumption weakens the naive Bayesian independence assumption since independence is assumed only locally, in each state, rather than for each class.

In most of the work done in text classification, the word order has been disregarded. Attempts have been made to exploit information in the word order by adding n-grams (Fürnkranz, 1998; Mladenic & Grobelnik, 1998) or phrases (Dumais et al., 1998) to the feature vector. While such approaches do lead to a richer representation containing more information, it also results in the classifier having to deal with a number of additional features growing exponentially in n.

Our empirical studies show some interesting results. HMMs clearly offer the potential to improve classification results over bag-of-word classifiers. When negative examples fall into natural subclusters, like with the 20 newsgroups data set, then assigning each of these clusters to a distinct state nearly always leads to uniformly superior precision-recall curves. The positive class membership of examples that are negative for the currently focused class is information that is nearly always disregarded but, as our experiments show, can help greatly to improve classification accuracy.

In our experiments, the Baum-Welch algorithm was generally not able to find HMM parameters that perform similarly well. We believe that considerable potential lies in better optimization techniques to find HMM parameters.

Many well-known text classifiers, such as Support Vector Machines are only able to deal with binary decision problems. In order to solve multi-class problems accurately, error correcting codes (*e.g.,* (Dietterich & Kong, 1995; Ghani, 2000)) have to be employed. Using a single HMM, it is possible to determine the most likely of arbitrarily many classes as well as a decision function value for each of the involved classes.

Maximum entropy models (della Pietra et al., 1997; McCallum et al., 2000) and conditional random fields (Lafferty et al., 2001) seem to be an interesting learning bias for text analysis. It would be interesting to see whether the computationally expensive GIS/IIS algorithms are able to handle the samples sizes that are typical for text classification. Like naive Bayes, HMMs are very quick to train. An iteration of Baum-Welch for a HMM with 20 states and 20,000 examples from labeled data; when every observation is labeled with one state, Baum-Welch terminates after one iteration.

References

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

Brants, T. (1999). Cascaded markov models. *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics.*

della Pietra, S., della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*, 380–393.

Dietterich, T., & Kong, E. (1995). *Machine learning bias, statistical bias, and statistical variance of decision tree algorithms* (Technical Report). Department of Computer Science, Oregon State University.

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. *Proceedings of the Seventh International Conference on Information and Knowledge Management.*

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). *Biological sequence analysis: Probabilistic models of proteins and nucleic acids.* Cambridge University Press.

Fürnkranz, J. (1998). *A study using n-gram models for text classification* (Technical Report). Austrian Research Institute for Artificial Intelligence.

Ghani, R. (2000). Using error-correcting codes for text classification. *Proceedings of the International Conference on Machine Learning.*

Kushmerick, N., Weld, D., & Doorenbos, R. (1997). Wrapper induction for information extraction. *Proceedings of the International Joint Conference on Artificial Intelligence.*

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. *Proceedings of the International Conference on Machine Learning.*

Langley, P., Iba, W., & Thompson, K. (1992). An analysis of bayesian classifiers. *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 223–228).

McCallum, A., Freitag, D., & Pereira, F. (2000). Maximum entropy Markov models for information extraction and segmentation. *Proceedings of the Seventeenth International Conference on Machine Learning.*

McCallum, A., & Nigam, K. (1998). Employing em and pool-based active learning for text classification. *Proceedings of the International Conference on Machine Learning.*

Mladenic, D., & Grobelnik, M. (1998). Feature selection for classification based on text hierarchy. *Working Notes of Learning from Text and the Web, Conference on Automated Learning and Discovery.*

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

Scheffer, T., & Wrobel, S. (2002). *Hidden markov models for text classification and information extraction* (Technical Report). University of Magdeburg.