A Maximum Entropy Tagger with Unsupervised Hidden Markov Models

Jun'ichi Kazama, Yusuke Miyao and **Jun'ichi Tsujii**

Department of Computer Science, Graduate School of Information Science and Technology,
University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
{kazama,yusuke,tsujii}@is.s.u-tokyo.ac.jp

Abstract

We describe a new tagging model where the states of a hidden Markov model (HMM) estimated by unsupervised learning are incorporated as the features in a maximum entropy model. Our method for exploiting unsupervised learning of a probabilistic model can reduce the cost of building taggers with no dictionary and a small annotated corpus. Experimental results on English POS tagging and Japanese word segmentation show that in both tasks our method greatly improves the tagging accuracy when the model is trained with a small annotated corpus. Furthermore, our English POS tagger achieved better-than-state-of-the-art POS tagging accuracy (96.84%) when a large annotated corpus is available.

1 Introduction

Recent statistical methods achieve high accuracy for various tagging tasks such as part-of-speech (POS) tagging, phrase chunking, and named entity recognition (Brill, 1994; Ratnaparkhi, 1996; Brants, 2000; Kudoh and Matsumoto, 2000; Borthwick, 1999). However, existing methods assume the existence of a large annotated corpus and/or a large dictionary, which cannot be expected in practice. When we develop a tagger for a new domain or task of interest, it is often the case that we have no annotated corpus. It is also not sure that we have a large dictionary for the task. Therefore, there is an urgent need to reduce the building cost of a tagger, or equivalently the developing cost of an annotated corpus and a dictionary, as well as a need to achieve high accuracy.

In this paper, we propose a new tagging model which uses the states of a hidden Markov model (HMM) estimated by unsupervised learning as the features in a maximum entropy model. Unsupervised learning of an HMM provides a reliable language model without annotated corpora. Meanwhile, the maximum entropy method is powerful enough to achieve the state-of-the-art accuracy in tagging tasks (Ratnaparkhi, 1996). By combining the two models, we expect to achieve (i) moderately high accuracy with a small annotated corpus, and (ii) the state-of-the-art accuracy with a sufficiently large annotated corpus. These characteristics are desirable when we develop a tagger from scratch in an incremental manner where we repeat training of a tagger, automatic tagging, and hand correction.

The architecture of our model is motivated by the following two aspects of the learning of a tagging model: *task learning* (e.g., to be a POS tagger or a phrase chunker) and *domain adaptation* (e.g., to be for newspaper texts or Web texts; for Japanese or English). Task learning inherently requires annotated corpora or dictionaries because the relation between observed events and unobservable tags must be learned. On the other hand, there is a hope for annotation-free (i.e., unsupervised) learning in domain adaptation. For example, an n-gram model and an HMM can be trained without annotated corpora so as to capture the occurrence and cooccurrence patterns of words in the domain in the form of probability parameters. The trained models should provide the matrix for the task learning. We will achieve higher tagging accuracy with a small annotated corpus by exploiting such unsupervised learning for domain adaptation.

We conducted experiments on two different tagging tasks, English POS tagging and Japanese word segmentation. In English POS tagging, our method significantly improved the accuracy with a small annotated corpus, and achieved 96.84% accuracy, which is better than the state-of-the-art accuracy, for the Wall Street Journal (Marcus et al., 1993) when a large annotated corpus is used for training. We estimated the cost of build-

ing a tagger from scratch, and found that about 40% of the hand corrections required to achieve 95% accuracy are reduced in English POS tagging. In Japanese word segmentation, we observed the similar results, which attest the applicability of our model to various tagging tasks.

2 Probabilistic Tagging Models

In this paper, we define tagging as the process which assigns a tag τ for each symbol o_t at the position t in a given symbol sequence. We denote a sequence $x_k x_{k+1} \cdots x_{l-1} x_l$ as x_k^l, and thus we denote a symbol sequence (i.e., a sentence) of length T as o_1^T. In English POS tagging, a symbol corresponds to a word, and in Japanese character-based word segmentation it corresponds to a character. This section presents a brief explanation for the existing tagging models which our model based on.

2.1 Tagging with a Maximum Entropy Model

In a maximum entropy tagging model (Ratnaparkhi, 1996), tags to be assigned are determined according to the conditional probability of the form:

$$\begin{aligned} p(\tau|h_t) &= \frac{1}{Z(h_t)} \prod_i \alpha_i^{f_i(\tau,h_t)}, \\ Z(h_t) &= \sum_\tau \prod_i \alpha_i^{f_i(\tau,h_t)}, \end{aligned} \quad (1)$$

where τ is a tag and h_t is a context at the position t. One can consider h_t as an information source from which we can obtain the current symbol, surrounding symbols, previously assigned tags, and so on. A function $f_i(\tau, h_t)$, called a *feature function*, indicates the occurrence of a certain clue to determine tags. In the model which we present, feature functions have the following form as in (Ratnaparkhi, 1996).

$$f_i(\tau, h_t) = \begin{cases} 1 & \text{if } H(h_t) \wedge \tau = Y \\ 0 & \text{otherwise} \end{cases}$$

where $H(h_t)$ is a predicate whose truth value is determined by h_t. For example, $H(h_t)$ is defined as follows.

$$\begin{aligned} o_{t-(n-1)}^t &= X_1 X_2 \cdots X_n \ (\textit{symbol n-gram}) \\ \tau_{t-n}^{t-1} &= Y_1 Y_2 \cdots Y_n \ (\textit{previous tags}), \end{aligned} \quad (2)$$

where X_i and Y_i are instantiated with the actual symbols or tags.

Parameters α_i can be interpreted as the importance of each feature function f_i. The parameters are computed so as to maximize the likelihood of the training data by using numerical optimization methods such as Improved Iterative Scaling (IIS) (Pietra et al., 1997). The training data are pairs of tag and context such as $\{(\tau_1, h_1), (\tau_2, h_2), \cdots, (\tau_N, h_N)\}$. We can obtain the training data from an annotated sentence (i.e., o_1^T and corresponding τ_1^T). For example, if we use the feature functions defined by $H(h_t)$ in (2), h_t will be constructed as follows, using o_1^T and τ_1^T.

$$h_t = \langle o_{t-(n-1)}, \cdots, o_t, \tau_{t-n}, \cdots, \tau_{t-1} \rangle \quad (3)$$

Given a maximum entropy model (i.e., f_i and α_i), the tagging procedure is formulated as the search which finds a sequence of tags $\hat{\tau}_1^T$ which has the maximum probability given a symbol sequence o_1^T:

$$\begin{aligned} \hat{\tau}_1^T &= \operatorname*{argmax}_{\tau_1^T} p(\tau_1^T | o_1^T) \\ &= \operatorname*{argmax}_{\tau_1^T} \prod_{t=1}^{T} p(\tau_t | h_t). \end{aligned} \quad (4)$$

When h_t includes tags previously assigned, a search algorithm such as the Viterbi-type algorithm or the beam search must be employed.

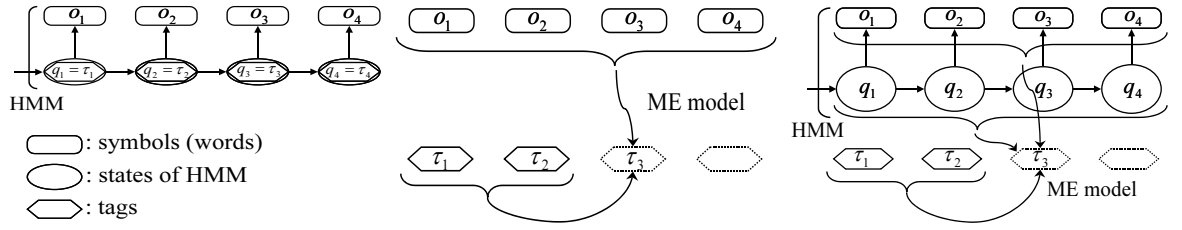
2.2 Tagging with a Hidden Markov Model

Hidden Markov models (HMMs) have been used in various NLP tasks because of their simplicity and efficiency (Cutting et al., 1992; Merialdo, 1994; Bikel et al., 1997). Hidden Markov models are regarded as probabilistic finite automata, which transit state to state according to *transition probabilities* and emit symbols according to *emission probabilities*. A detailed explanation of HMMs is found in (Rabiner, 1989).

In a classical tagging model using an HMM, states correspond to tags. The tagging procedure is formulated as the search which finds the state sequence $\hat{q}_1^T$ which has the maximum probability given a symbol sequence o_1^T:

$$\begin{aligned} \hat{q}_1^T &= \operatorname*{argmax}_{q_1^T} p(q_1^T | o_1^T) \\ &= \operatorname*{argmax}_{q_1^T} p(o_1^T, q_1^T) \\ &= \operatorname*{argmax}_{q_1^T} \prod_{t=1}^{T} p(q_t | q_{t-1}) p(o_t | q_t), \end{aligned} \quad (5)$$

where $p(q_t|q_{t-1})$ is a transition probability and $p(o_t|q_t)$ is an emission probability. The Viterbi algorithm (Viterbi, 1967) performs this search efficiently.



(a) HMM-based tagger (b) ME-based tagger (c) Our model

Figure 1: Other tagging models and our model. Note that states and tags are equivalent in a classical HMM-based tagger (a), while they are not necessarily equivalent in our model (c).

What we focus on in this paper is unsupervised learning of HMM's parameters (i.e., transition probabilities and emission probabilities). The Baum-Welch algorithm (Baum and Eagon, 1967) repeatedly updates parameters from the initial parameters, and tries to find the parameters which maximize the likelihood of given symbol sequences. We can say that the model trained by the Baum-Welch algorithm is a task-independent language model which adapts to the domain from which the symbol sequences are collected.

Unsupervised learning by the Baum-Welch algorithm appears useful for our purpose. Cutting et al. achieved 96% accuracy in English POS tagging without an annotated corpus by training an HMM by the Baum-Welch algorithm (Cutting et al., 1992). However, their success is much due to the use of a word dictionary built from a large annotated corpus. As mentioned earlier, we do not assume any dictionary since we are thinking of building a tagger for a new domain or task where such a dictionary is not available. In our setting, the direct application of the Baum-Welch algorithm to an HMM-based tagger *always* decreases the accuracy in our setting, as shown in the experiments on English POS tagging in Section 4. This result seems to be the worst case expected from the report by Merialdo (Merialdo, 1994) that the Baum-Welch algorithm does not always improve the tagger's accuracy even though it always improves the model in terms of the likelihood.

In spite of these anxieties, we will see that the Baum-Welch training of HMMs really improves the accuracy by being combined with the maximum entropy method using our method which is presented in the next section.

3 Integration of HMM and Maximum Entropy Tagging

Our method to integrate a maximum entropy method and unsupervised learning of an HMM is quite simple: use the states of an HMM as the features of a maximum entropy tagging model.

We introduce feature functions which access the state sequence $\hat{q}_1^T$ of an HMM obtained by the Viterbi algorithm. That is, we introduce the feature functions of the form:

$$f_i(\tau, h_t) = \begin{cases} 1 & \text{if } \hat{q}_{t+k-(n-1)}^{t+k} = Z_1 \cdots Z_n \\ & \wedge\, \tau = Y \\ 0 & \text{otherwise,} \end{cases}$$

where k indicates the offset from the current position t. We call these features *state features*. Figure 1 illustrates the relation between previously described standard models and our model. Our method is described as follows.

Training:

1. Train an HMM M_1 with an unannotated corpus by using the Baum-Welch algorithm.
2. Find the most probable state sequence $\hat{q}_1^T$ for a symbol sequence o_1^T in an annotated corpus by using the Viterbi algorithm of M_1. Then, make the training data $\{(\tau_1, h_1), \cdots, (\tau_T, h_T)\}$ from $\hat{q}_1^T$, o_1^T, and the corresponding tag sequence τ_1^T.
3. Estimate the parameters of the maximum entropy model M_2, using the training data for all the symbol sequences in the annotated corpus.

Runtime:

1. Find the most probable state sequence $\hat{q}_1^T$ for a symbol sequence o_1^T by using the Viterbi algorithm of M_1.
2. Find for each symbol o_t the most appropriate tag τ_t determined by h_t, which includes $\hat{q}_1^T$, using the maximum entropy model M_2.

Our method is very flexible from the viewpoint of model design. Because unsupervised learning of an HMM is separated from the maximum en-

tropy model, which is responsible for assigning tags, we can choose the HMM's properties such as the number of states and the initial parameters independently from the tag set (i.e., the task) so that we can achieve the best domain adaptation. There are several possibilities for the HMM's setting as follows.

- Assume one-to-one state-tag correspondence as in classical models. The initial parameters are estimated from an available small annotated corpus.
- Not assume one-to-one correspondence. Therefore, the number of states can be greater than the number of tags. The HMM can be initialized, for example, randomly.

If we select the former, we can say that our method combines a classical HMM-based tagger trained by the Baum-Welch algorithm with a maximum entropy model. This can be a reasonable way, but we emphasize the latter in this paper. A larger state space possibly leads to a better model and might cause higher accuracy. In fact, the experiments on English POS tagging in Section 4 show that the best performance is achieved when we use the latter scheme. In Japanese character-based word segmentation, the former would yield a very poor model since we have only two tags. As shown in the experiments, our method becomes applicable to Japanese character-based word segmentation by using the latter scheme.

4 Experimental Results

4.1 Experiments on English Part-of-Speech Tagging

We applied our method to English POS tagging. The experiments are conducted on the Wall Street Journal from the Penn Treebank (Marcus et al., 1993). The Wall Street Journal consists of approximately 50,000 annotated sentences, and the number of POS tags is 45. Over 96.5% accuracy is reported in previous studies (Ratnaparkhi, 1996; Brants, 2000). We divided the corpus into two parts, one for training (section 00–22), and one for test (section 23 and 24). The training part consists of 45,446 sentences (1,084,229 words) and the test part consists of 3,762 sentences (89,537 words). First 32,000 sentences of the training corpus are used as an unannotated corpus for unsupervised learning.

Experiment 1

To see the effect of our method, we compare the accuracies of the following three models, (A), (B), and (C), varying the number of available annotated sentences from 100 to 40,000[1].

Table 1: Features used in our maximum entropy English POS tagger. The expressions such as $(\{a, b\}, c)$ mean $(a, c), (b, c)$.

category	$H(h_t)$
symbols	$o_{t+k-(n-1)}^{t+k} = X_1 \cdots X_n$ $(k, n) = (\{-2, -1, 0, 1, 2\}, 1)$
previous tags	$\tau_{t-n}^{t-1} = Y_1 \cdots Y_n$ $n = 1, 2$
HMM states	$q_{t+k-(n-1)}^{t+k} = Z_1 \cdots Z_n$ $(k, n) = (\{-1, 0, 1\}, 1), (0, 2), (0, 3)$
number	o_t contains a number
uppercase	o_t contains an uppercase
hyphen	o_t contains a hyphen
suffix	suffix of $o_t = W$, $\|W\| \leq 4$
prefix	prefix of $o_t = W$, $\|W\| \leq 4$

(A) A baseline tagger which uses a 45-state HMM estimated from annotated sentences by relative frequency with absolute discounting[2].

(B) A maximum entropy based tagger which is almost the same as the state-of-the-art English POS tagger described in (Ratnaparkhi, 1996). The difference is that we ignored whether a word is rare or not and used the first-order Viterbi search instead of the beam search. This difference has little impact on the accuracy and the result is consistent with the published accuracy 96.43%.

(C) Our model, which uses state features in addition to the features in (B). A 160-state HMM trained from random parameters is used. The HMM is trained by using a variant of the Baum-Welch algorithm which employs the Gibbs sampling as described in (Ghahramani and Jordan, 1997) to enable fast training. The whole set of features used in our model is listed in Table 1.

Annotated sentences are used to estimate the parameters of the HMM (A) and the maximum entropy models (B) and (C). In all the models, the most frequent 10,000 words are treated as distinct symbols and all other words are shrunk to one symbol *UNK*.

Table 2 shows the results, and Figure 2 depicts them graphically. We can see that our model improves the accuracy especially when we have only a small annotated corpus. In addition, our model performs better than the state-of-the-art maximum entropy model when we have sufficiently

[1] Annotated sentences are contiguous, starting from the beginning of the training corpus.

[2] The discount value is 0.5.

Table 2: English POS tagging accuracy for the Wall Street Journal corpus.

# of annotated sentences	(A)	(B)	(C)	(C') cut-off=2
100	72.29	70.74	**75.73**	**83.54**
200	78.25	80.03	**84.48**	**87.62**
500	83.89	86.03	**89.14**	**91.45**
1,000	87.19	89.76	**91.84**	**92.89**
2,000	89.98	92.58	**93.82**	**94.22**
4,000	91.63	94.16	**94.91**	**95.02**
8,000	92.86	95.25	**95.66**	95.54
16,000	93.61	96.11	**96.30**	96.13
32,000	93.86	96.54	**96.73**	96.52
40,000	93.94	96.74	**96.84**	96.63

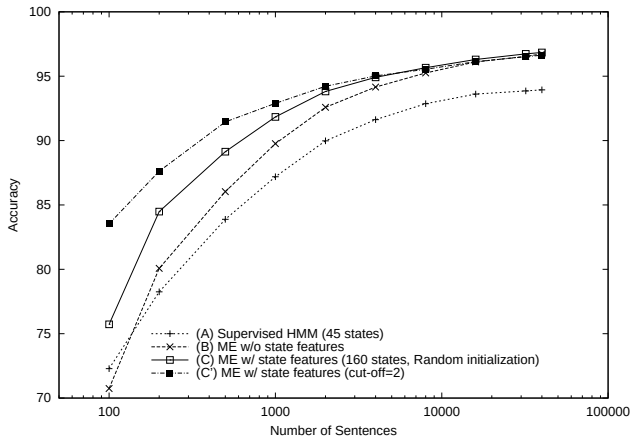


Figure 2: English POS tagging accuracy for the Wall Street Journal corpus.

large annotated corpora. This indicates that the use of state features has its own right in maximum entropy tagging as well as it prevents the accuracy declining with a small annotated corpus.

We can further improve the accuracy with a small annotated corpus by tuning the cut-off value in the estimation of the maximum entropy model. In a maximum entropy model, it is preferable to omit (cut-off) features which occur rarely and not statistically reliable. The optimal cut-off value may vary depending on many factors such as the size of available annotated corpus, the set of features, and the number of symbols. While we omitted features which occur less than 10 times in (B) and (C) as in (Ratnaparkhi, 1996) for the comparison, we also included the accuracy of the model (C'), which is the same as (C) except that the cut-off value is set to 2. We can see that such a small cut-off value increases the accuracy with a small annotated corpus but decreases with a large annotated corpus. Although we do not investigate this issue further in this paper, the results indicate that we can expect further improvement if we employ a more sophisticated cut-off method.

Table 3: Required hand corrections (corr.) and total annotations (total.) for English POS tagging. Numbers are shown in terms of word. 1 sentence $\approx$ 25 words.

	(B)		(C)			
target acc.	*corr.*	*total*	*corr.*	*total*	reduction *corr.*	*total*
90.00	3,983	23,355	2,101	13,259	47%	43%
94.00	9,526	89,391	5,502	54,218	42%	39%
95.00	14,171	167,600	8,532	105,162	40%	37%
96.00	23,503	359,025	17,275	292,191	26%	19%
96.50	38,186	730,802	27,713	558,571	27%	24%

In addition, we roughly assessed how many hand corrections can be reduced by our method. The number of hand corrections is one measure of the building cost of a tagger, while the total cost includes other factors such as the cost of finding the incorrect regions. As stated in Section 1, we often develop an annotated corpus and a tagger in the following manner.

1. Set the milestones in the size of annotated corpus.
2. Make a temporal tagger by using an available annotated corpus at each milestone.
3. Make the next milestone corpus by tagging unannotated sentences with that tagger, and correcting incorrect tags by hand.
4. Repeat 2. and 3. until the target accuracy is achieved.

If we follow such an incremental development, the amount of hand corrections is calculated as:

$$\sum_{i=1}^{N-1} \{annot(i+1) - annot(i)\} \times \{1 - acc(i)\}, \quad (6)$$

where $annot(i)$ is the size of the annotated corpus at ith milestone, and $acc(i)$ is the accuracy of the ith tagger. The target accuracy is $acc(N)$. Table 3 shows how many hand corrections are required for the model (B) and our model (C) to reach various target accuracies[3]. We included the total amount of annotated sentences as well. Our method greatly reduces both the required hand corrections and the total amount of the annotated sentences. For example, we can reduce 40% of the hand corrections when the target accuracy is 95%.

[3]The total size of annotated corpus $annot(N)$ is approximated by using linear interpolation, when the target accuracy is in between the accuracies known from the table.

Table 4: Tagging accuracy for the Wall Street Journal corpus: combination with 45-state HMMs.

# of annotated sentences	(A)	(A')	(D)	(E)
100	72.29	65.46	74.58	**79.13**
200	78.25	65.38	82.88	**85.01**
500	83.89	64.98	88.45	**89.18**
1,000	87.19	64.81	91.52	**91.58**
2,000	89.98	65.97	93.21	**93.74**
4,000	91.63	67.63	94.73	**94.80**
8,000	92.86	68.75	95.46	**95.62**
16,000	93.61	70.31	**96.27**	96.21
32,000	93.86	71.31	**96.71**	96.67

Experiment 2

In this experiment, we investigate the effect of the Baum-Welch algorithm. As described in Section 3, it is also possible to assume one-to-one state-tag correspondence in an HMM in English POS tagging. Table 4 shows the accuracies of the following models (D) and (E).

(D) A maximum entropy model combined with a 45-state supervised HMM ((A) in Experiment 1).

(E) A maximum entropy model combined with an HMM, (A'), which is trained by the Baum-Welch algorithm with (A) as the initial parameters.

The direct application of the Baum-Welch algorithm always decreases the accuracies as shown in (A') in Table 4. In spite of this fact, the model (E) combined with (A') performs better than (D) combined with (A), when the size of annotated corpus is 100–8000 sentences. That is, the Baum-Welch algorithm really improves the accuracy when only a small annotated corpus is available.

Another fact which we can see from these experiments is that the model (C), where the number of states is greater than the number of tags, achieves the best accuracy with a large annotated corpus. However, it seems that a small number of states has an advantage to prevent the decline of accuracy with a small annotated corpus. This indicates that we have to choice the size of an HMM carefully, or switch the scheme at some appropriate point of the development.

4.2 Experiments on Japanese Word Segmentation

To see that our method is generally applicable to other tagging tasks, we conducted experiments on Japanese word segmentation. In languages such as Japanese and Chinese we must identify each word first of all since words are not separated by spaces. This task is called *word segmentation*. Character-based methods for word segmentation are considered to be robust against the unknown word problem because it does not require word dictionaries. 95–96% segmentation accuracies are reported in previous studies on character-based word segmentation (Nagamatsu and Tanaka, 1997; Oda and Kita, 1999; Oda et al., 1999). Following the previous studies, we formulate character-based Japanese word segmentation as the tagging which assigns *end* to the characters at the end of words, and *others* to the other characters.

We conducted the experiments on the Kyoto corpus (ver. 3.0) (Kurohashi and Nagao, 1997), which consists of about 40,000 (20,000 from articles and 20,000 from editorials) annotated newspaper sentences of the Mainichi Newspapers. We used only the article part for the experiments, and divided it into the training part (17,694 sentences) and the test part (1,966 sentences). Unsupervised training of an HMM was performed using about 110,000 sentences of Mainichi Newspapers (1994)[4].

We measure the performance of word segmentation by recall and precision as well as tagging accuracy per character. Let *Std* be the number of words in the test data and *Cor* the number of correctly identified words, recall is defined as *Cor*/*Std*. And let *Sys* be the number of words identified by the tagger, precision is defined as *Cor*/*Sys*. To save spaces, we show the results in F-value calculated as:

$$\textit{F-value} = \frac{2 \times \textit{Recall} \times \textit{Precision}}{\textit{Recall} + \textit{Precision}}$$

In this experiments, we compared the following three models, (A), (B), and (C), varying the number of annotated sentences.

(A) A character trigram model based on the segmentation model described in (Stolcke and Shriberg, 1996)[5]. This model's structure is almost the same as the model used in (Oda et al., 1999), except that they use character classes obtained by character clustering.

(B) A maximum entropy model without state features. The features are only the symbol features in Table 5.

(C) A maximum entropy model with state features. The features are listed in Table 5. The

[4]CD-Mainichi Shimbun'94, Nichigai Associates. These sentences and the Kyoto corpus are disjoint, since the Kyoto corpus covers the year 1995.

[5]We used SRILM (The SRI Language Modeling Toolkit) to estimate this trigram model (SRI, 2000).

Table 5: Features used in our maximum entropy model for Japanese word segmentation.

category	$H(h_t)$
symbols	$o_{t+k-(n-1)}^{t+k} = X_1 \cdots X_n \ \ (k,n) = (\{-1,0,1\},1), (\{-1,0,1\},2), (\{-1,0,1\},3)$
HMM states	$q_{t+k-(n-1)}^{t+k} = Z_1 \cdots Z_n \ \ (k,n) = (\{-1,0,1,2,3\},1), (\{0,1,2\},2), (0,3)$

Table 6: Word segmentation accuracy (F-value) for the Kyoto corpus.

# of annotated sentences	(A)	(B)	(C)
50	40.80	66.57	**76.97**
100	49.43	71.38	**82.46**
200	62.88	74.57	**85.80**
500	77.30	80.55	**89.31**
1,000	84.36	85.09	**91.32**
2,000	88.01	87.90	**92.54**
4,000	91.35	90.75	**93.97**
8,000	93.75	92.92	**94.84**
10,000	94.21	93.53	**95.12**
12,000	94.61	93.93	**95.32**
16,000	95.18	94.76	**95.68**

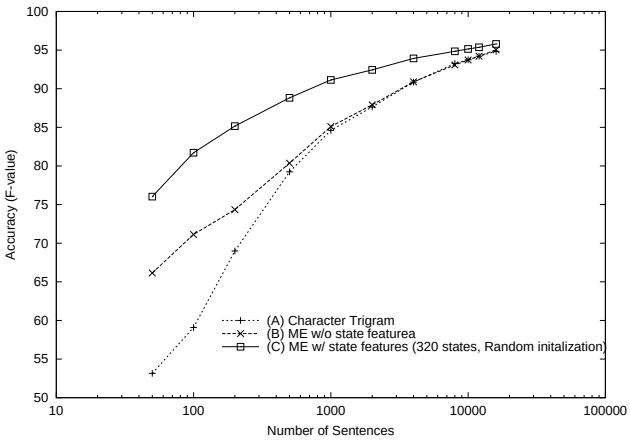


Figure 3: Word segmentation accuracy for the Kyoto corpus.

Table 7: Required hand corrections and total annotations for Japanese word segmentation. 1 sentence $\approx$ 46 characters.

	(A)		(C)			
target acc.	*corr.*	*total*	*corr.*	*total*	reduction *corr.*	*total*
94.0	22,167	411,491	7,177	185,412	68%	55%
95.0	28,399	673,520	12,824	414,486	55%	38%

cut-off value was set to 1. A 320-state HMM trained from random parameters is used.

The most frequent 2,000 characters are considered as distinct symbols, and they are augmented by a binary feature which indicates whether the next character type is different from the current character type (i.e., the number of symbols becomes approximately 4,000)[6]. We did not use this augmentation for model (A) because it had turned out to have little effects on the accuracy.

The accuracies for these three models are shown in Table 6, and graphically in Figure 3. As in the English POS tagging, our method greatly improves the accuracy especially with only a small annotated corpus, and achieves the highest accuracy when a large annotated corpus is available.

Table 7 shows the amount of hand corrections required for model (A) and (C) to reach the target accuracies. Target accuracies are shown in F-value and the required amounts are shown in the number of characters[7]. Our method reduced 55% of the hand corrections when our target is 95.0% F-value.

5 Discussion

Our method relates to data sparseness prevention by *clustering* (Brown et al., 1992; Oda et al., 1999). If we use a 320-state HMM, a state for each symbol obtained by the Viterbi algorithm is considered as the most appropriate class for the symbol among 320 classes. Our approach differs from previous works in that classes are dynamically determined by the Viterbi algorithm, depending on the context in which the symbol appears. Although we believe such dynamic classification is more suitable for tagging tasks than static classification as in the previous works, we will need more study on whether our model is better than the other models using clustering methods.

While we used only the most probable path of an HMM obtained by the Viterbi algorithm, it is also possible for the maximum entropy model to access other paths of the HMM. In that case, state feature functions will be non-binary real-valued functions which indicate the probability of being in a state. Such ambiguous features, which are analogous to soft clustering, might be useful to prevent data sparseness, while we need more research on this.

[6] In Japanese, there are three major character types, *kanji*, *hiragana*, and *katakana*, and they are very useful clues to identify a word.

[7] The accuracy per character is used as $acc(i)$ in the formula (6).

6 Conclusion

We have presented a tagging model which uses the states of an HMM as the features in a maximum entropy model, aiming at reducing the building cost of a tagger. The experiments showed that our method improves the accuracy especially with a small corpus and achieves better-than-state-of-the-art accuracy when a large annotated corpus is available. Furthermore, a series of experiments showed our method is widely applicable to several tagging tasks. We will further investigate more sophisticated cut-off methods in the estimation of the maximum entropy model, and more suitable unsupervised models. We should also apply our method to other tagging tasks such as phrase chunking and named entity recognition in the domain where there is only a small annotated corpus.

References

L. E. Baum and J. A. Eagon. 1967. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model of ecology. *Bull. Amer. Math. Soc.*, 73:360–363.

D. Bikel, S. Miller, R. Schwartz, and R. Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 194–201.

Andrew Borthwick. 1999. A maximum entropy approach to named entity recognition. Ph.D. Thesis. New York University.

Thorsten Brants. 2000. TnT – a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference ANLP-2000.*

Eric Brill. 1994. Some advances in transformation-based part of speech tagging. In *Proceedings of the Twelefth National Conference on Artificial Intelligence (AAAI-94).*

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. 18(4):467–479.

Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Language Processing*, pages 133–140.

Zoubin Ghahramani and Michael I. Jordan. 1997. Factorial hidden Markov models. *Machine Learning*, 29:245–273.

Taku Kudoh and Yuji Matsumoto. 2000. Use of support vector learning for chunk identification. In *Proceedings of CoNLL-2000 and LLL-2000.*

Sadao Kurohashi and Makoto Nagao. 1997. Kyoto University text corpus project. In *3rd Annual Meeting of Natural Language Processing*, pages 115–118. (in Japanese).

Mitchell Marcus, Beatrice Santrini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: Penn Treebank. *Computational Linguistics*, 19(2):313–330.

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

Kenji Nagamatsu and Hidehiko Tanaka. 1997. A stochastic morphological analysis for Japanese employing character n-gram and k-NN method. In *NLPRS'97*, pages 23–28.

Hiroki Oda and Kenji Kita. 1999. A character-based Japanese word segmenter using a PPM*-based language model. In *Proceedings of the 18th International Conference on Computer Processing of Oriental Languages (ICCPOL'99)*, pages 527–532.

Hiroki Oda, Shinsuke Mori, and Kenji Kita. 1999. A Japanese word segmenter by a character class model. *Natural Language Processing*, 6. (in Japanese).

S. Pietra, V. Pietra, and J. Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

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

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142.

SRI. 2000. SRILM – The SRI Language Modeling Toolkit ver. 1.0. available via `http://www.speech.sri.com/projects/srilm/`.

A. Stolcke and E. Shriberg. 1996. Automatic linguistic segmentation of conversational speech. In *Proc. ICSLP '96*, volume 2, pages 1005–1008, Philadelphia, PA.

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