NAIST-IS-DD0461022

Doctoral Dissertation

Multilingual Word Segmentation and Part-of-Speech Tagging: a Machine Learning Approach Incorporating Diverse Features

Tetsuji Nakagawa

March 17, 2006

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Tetsuji Nakagawa

Thesis Committee:

Professor Yuji Matsumoto	(Supervisor)
Professor Shin Ishii	(Co-supervisor)
Professor Kiyohiro Shikano	(Co-supervisor)
Associate Professor Kentaro Inui	(Co-supervisor)

Multilingual Word Segmentation and Part-of-Speech Tagging: a Machine Learning Approach Incorporating Diverse Features*

Tetsuji Nakagawa

Abstract

The aim of this dissertation is to study statistical methods for multilingual word segmentation and POS tagging with high accuracy. Word segmentation and part-of-speech (POS) tagging are fundamental language analysis tasks in natural language processing, and used in many applications. Existence of unknown words is a large problem in these tasks and they need to be properly handled. We attempt to develop suitable methods for word segmentation and POS tagging which can utilize informative features effectively.

Firstly, we study a method for unknown word guessing and part-of-speech tagging using support vector machines (SVMs), which can handle a number of features effectively. We apply the method to English unknown word guessing and part-of-speech tagging.

Secondly, we propose a method for POS guessing of unknown words using global information as well as local information. Global features often give useful information for POS guessing, and the method takes into consideration interactions between the POS tags of all the unknown words in a document by using Gibbs sampling. We apply the method to Chinese, Japanese and English unknown word guessing.

Thirdly, we propose a word segmentation method which combines the existing word-based method and character-based method, in order to compensate for the

*Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0461022, March 17, 2006.

weakness of each method and obtain high overall accuracy. We apply the method to Chinese and Japanese word segmentation and Korean morphological analysis.

Fourthly, we propose a method named revision learning, which solve the inefficiency problem of SVMs. The method combines a binary classifier with high generalization capacity like SVMs and a stochastic model with small computational cost, in order to achieve high performance with small computational cost. We apply the method to English POS tagging and Japanese word segmentation and POS tagging.

Fifthly, we study a method for corpus error detection using SVMs. The method extracts inconsistencies in corpora by finding exceptional elements from the corpora. We apply the method to detection of errors in English and Japanese corpora.

Keywords:

Word Segmentation, Part-of-Speech Tagging, Morphological Analysis, Unknown Word Processing, Corpus Error Detection.

Acknowledgements

I would like to express my sincere gratitude to Professor Yuji Matsumoto for his valuable advice and suggestions. I learned a lot from him and I am glad that I had the opportunity to study at the Computational Linguistics laboratory. I am grateful to Associate Professor Kentaro Inui and Dr. Masashi Shimbo for giving me useful comments and suggestions. My special thanks go to Dr. Masayuki Asahara for his helpful advice and comments. I thank all the members of the Computational Linguistics laboratory, especially the members of the old SVM group, for their help and fruitful discussions.

I would like to acknowledge the members of my thesis committee, Professor Shin Ishii and Professor Kiyohiro Shikano, for giving me valuable comments and suggestions.

This dissertation was completed at Oki Electric Industry Co., Ltd. I would like to thank Mr. Toshihisa Nakai and Mr. Toshiki Murata for their support. I also thank Dr. Mihoko Kitamura, Mr. Masashi Sakamoto, Ms. Miki Sasaki, Mrs. Sayori Shimohata, Mr. Tatsuya Sukehiro and Mr. Takahiro Yamasaki for their help and discussions.

I would like to thank Associate Professor Mikio Yamamoto at University of Tsukuba, who was my supervisor when I was an undergraduate student, for giving me the opportunity to learn natural language processing.

Finally, I would like to thank my parents, sister and brother for their support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Word Segmentation and Part-of-Speech Tagging

Computer systems with natural language input need language analysis in the first place. Word segmentation is a task to segment an input sentence into words and part-of-speech (POS) tagging is a task to identify the part-of-speech of each word in a sentence, and they are fundamental language analysis tasks in natural language processing (NLP). Especially in Chinese and Japanese, words are not separated by spaces, and word segmentation is an indispensable processing[1]. These analyses are necessary for other high-level language analyses including named entity recognition and syntactic parsing, and are used in many NLP applications such as machine translation systems. If an error is made in word segmentation or POS tagging, it may cause failure of the subsequent processing. If the performance of word segmentation and POS tagging is improved, it will contribute to many application systems that use them.

Many ambiguities exist in word segmentation and POS tagging, that need to be resolved. Various studies on word segmentation and POS tagging have been conducted to cope with the problem. Recently, statistical methods based on corpora have been widely used. These methods obtain statistical information

[1]In Japanese language processing, word segmentation and POS tagging are often put together with other lexical analyses, and called morphological analysis.

or rules from manually annotated corpora, and the method now achieve high accuracy without costly handmade rules if appropriate corpora are available. For example, in English part-of-speech tagging, transformation-based error-driven learning was studied by Brill (1995), decision trees were used by Schmid (1994), and maximum entropy (ME) models were used by Ratnaparkhi (1996). Markov models are major models used in many studies (Cutting et al., 1992; Charniak et al., 1993; Brants, 2000), and conditional random fields are recently proposed (Lafferty et al., 2001).

Although word segmentation and POS tagging have relatively long histories in NLP, there are still some important and difficult problems related to the tasks, such as unknown word processing and multilingual processing.

1.2. Unknown Words

In word segmentation and POS tagging, we frequently encounter words that do not appear in training data nor the dictionary used by the analyzer. Such words are called *unknown words* or *out-of-vocabulary (OOV) words*. Existence of unknown words is a large problem in these tasks, since the statistical information or rules for those words are unavailable. Unknown words are usually handled by an exceptional processing. Accuracy of word segmentation or POS tagging for unknown words is usually much lower than that for known words, and this is a non-negligible problem especially where the size of training data is limited (Brants, 2000).

The task of guessing POS tags of unknown words is called unknown word guessing (Mikheev, 1997) and is a subtask of POS tagging. One standard approach for unknown word guessing is to predict parts-of-speech of unknown words using suffixes or surrounding contexts of the unknown words (Weischedel et al., 1993; Thede, 1998). Brants (2000) used linear interpolation of fixed length suffix models for unknown word handling in his POS tagger TnT. Other methods for unknown word guessing have been studied, including the rule-based method (Mikheev, 1997) and the decision tree-based method (Orphanos and Christodoulakis, 1999).

Handling unknown words in word segmentation is also a difficult problem but

important in Chinese and Japanese. The character-based tagging approaches (Xue, 2003; Asahara et al., 2003; Peng et al., 2004) are one solution for this problem. Sproat et al. (1996) and Nagata (1999b) used stochastic word models for unknown word processing, and Uchimoto et al. (2001) proposed an ME-based method.

One approach to cope with the unknown word problem is to prepare a large dictionary, and such an approach is relatively effective in practical systems. However, preparing a large enough dictionary is costly. Avoiding unknown words is difficult because new words are created all the time, and the unknown word problem is an essential problem in lexical analysis.

1.3. Research Objectives

The objective of this dissertation is to study statistical methods for word segmentation and POS tagging with high accuracy and to investigate their applicability to various languages. In statistical language analysis, which statistical model is used and how the model is applied to a task are important and determine accuracy and efficiency of the analysis. Especially, the features used in the model have a large effect on the accuracy. We attempt to develop suitable methods for word segmentation and POS tagging which can utilize informative features effectively. Although lexical analysis methods vary for different languages in general, there is a need of processing multilingual documents. We handle Chinese, Japanese, Korean and English in this dissertation.

1.4. Dissertation Outline

We address three lexical analysis tasks in this dissertation; word segmentation, POS tagging and corpus error detection. Word segmentation and POS tagging include unknown word processing. Figure 1.1 shows relation between the tasks and chapters in this dissertation. The region enclosed with a thick line is covered by the method explained in each chapter, and a dotted line indicates that the method is a meta-learning model which utilizes existing models.

This dissertation is organized as follows:

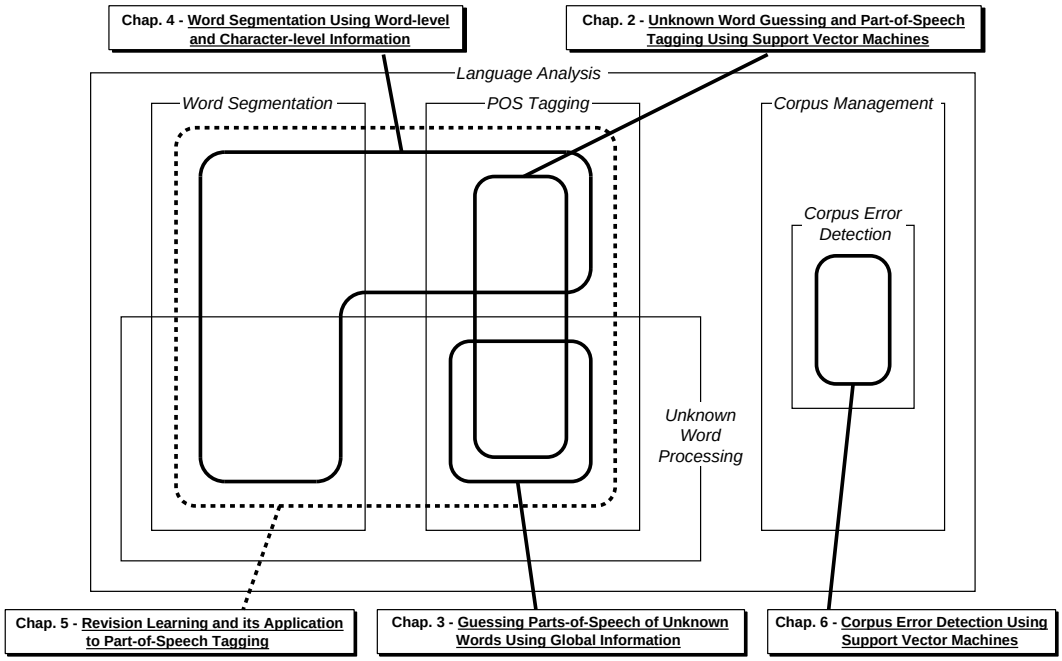

Figure 1.1. Tasks Treated in This Dissertation

Chapter 2: *Unknown Word Guessing and Part-of-Speech Tagging Using Support Vector Machines*

One of the well known models used for unknown word guessing and POS tagging are Markov models, but they have difficulty in handling a wide variety of features. In this chapter, we study a method for unknown word guessing and part-of-speech tagging using support vector machines (SVMs), which can handle a number of features effectively. We apply the method to English unknown word guessing and part-of-speech tagging.

Chapter 3: *Guessing Parts-of-Speech of Unknown Words using Global Information*

Although unknown word POS guessing is generally conducted using only local features within each sentence, document-wide global features seem to give useful information. In this chapter, we propose a method for POS guessing of unknown words using global information as well as local information. The method takes into consideration interactions between the

POS tags of all the unknown words in a document by using Gibbs sampling. We apply the method to Chinese, Japanese and English unknown word guessing.

Chapter 4: *Word Segmentation using Word-level and Character-level Information*

There are two approaches for Chinese and Japanese word segmentation; the word-based approach and the character-based approach. The former has difficulty in handling unknown words, and the latter performs worse for known words. We combine these two approaches in order to obtain high accuracy for both known words and unknown words. We apply the method to Chinese and Japanese word segmentation and Korean morphological analysis.

Chapter 5: *Revision Learning and its Application to Part-of-Speech Tagging*

Although the unknown word guessing and POS tagging methods proposed in Chapter 2 have high accuracy, their computational cost is high. The inefficiency is one large problem of SVMs. In order to solve the problem, we propose a method named revision learning. The method combines a binary classifier with high generalization capacity like SVMs and a stochastic model with small computational cost, in order to achieve high performance with small computational cost. We apply the method to English POS tagging and Japanese word segmentation and POS tagging.

Chapter 6: *Corpus Error Detection Using Support Vector Machines*

Tagged corpora used by statistical lexical analyzers often have annotation errors, and they need to be detected and fixed. In this chapter, we study a method for corpus error detection using SVMs. We apply the method to detection of errors in English and Japanese corpora.

Chapter 7: *Conclusion*

Finally, we summarize and conclude this dissertation.

Chapter 2

Unknown Word Guessing and Part-of-Speech Tagging Using Support Vector Machines

In this chapter, we discuss English unknown word guessing and POS tagging using support vector machines (SVMs). SVMs are known to have good generalization performance and we apply them to unknown word guessing and POS tagging. Experimental results show the method outperforms an existing Markov model-based method.

2.1. Introduction

The problem of unknown words is non-negligible in POS tagging. In this chapter, we propose a method for predicting POS tags of unknown words using SVMs as a post-processing of POS tagging. We handle English in this chapter. SVMs are supervised machine learning models for binary classification and are known to have good generalization performance. SVMs can handle a large number of features and hardly overfit. Consequently, SVMs have been successfully applied to several natural language processing applications (Joachims, 1998; Kudoh and Matsumoto, 2000). In unknown word guessing, handling a number of features and considering the combinations of the features are necessary, and SVMs seem to be appropriate for this purpose. In this chapter, we first apply SVMs to

unknown word guessing, and then we show how to apply SVMs to more general POS tagging.

This chapter is organized as follows: Section 2.2 explains SVMs. Sections 2.3 and 2.4 describe our method for unknown word guessing and POS tagging. Section 2.5 shows some experimental results. Section 2.6 discusses related work, and Section 2.7 concludes.

2.2. Support Vector Machines

Support vector machines (Vapnik, 1998; Cortes and Vapnik, 1995) are supervised machine learning models for binary classification on a feature vector space $\mathbf{x} \in \mathbf{R}^L$.

$$\mathbf{w} \cdot \mathbf{x} + b = 0, \qquad \mathbf{w} \in \mathbf{R}^L,\ b \in \mathbf{R}. \tag{2.1}$$

Suppose the hyperplane (2.1) separates training data, $\{(\mathbf{x}_i, y_i) \,|\, \mathbf{x}_i \in \mathbf{R}^L, y_i \in \{-1, +1\}, 1 \leq i \leq l\}$, into two classes such that

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1. \tag{2.2}$$

While a number of such separating hyperplanes exist (Figure 2.1, left hand side), SVMs find the optimal hyperplane that maximizes the margin (the distance between the hyperplane and the nearest example) (Figure 2.1, right hand side). Such a hyperplane is known to have the minimum expected test error and can be solved by quadratic programming. Given a test example $\mathbf{x}$, its label y is decided by the sign of the discriminant function $f(\mathbf{x})$ as follows:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b, \tag{2.3}$$

$$y = \mathrm{sgn}(f(\mathbf{x})). \tag{2.4}$$

For linearly non-separable cases, feature vectors are mapped into a higher dimensional space by a nonlinear function $\mathbf{\Phi}(\mathbf{x})$ and linearly separated there. Since all examples appear as forms of inner products in SVMs' formulae, we only need the inner product of two examples in the higher dimensional space. Those

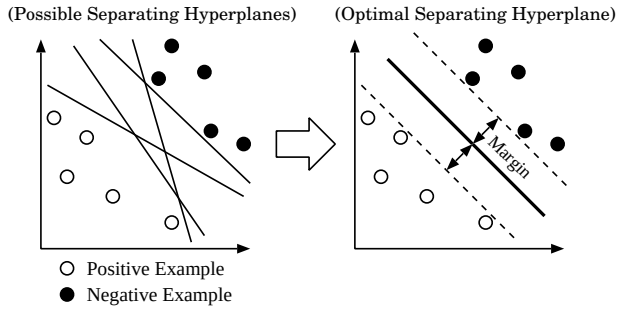

Figure 2.1. Maximizing the Margin

values may be calculated in $\mathbf{R}^L$ without mapping to the higher dimensional space by the following function $K(\mathbf{x}_i, \mathbf{x}_j)$,

$$K(\mathbf{x}_i, \mathbf{x}_j) \quad = \quad \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j). \tag{2.5}$$

The functions that conduct such calculation are called *kernel functions*. The following function is one of the kernel functions, called the polynomial kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) \quad = \quad (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d. \tag{2.6}$$

This function virtually maps the original input space into a higher dimensional space where all combinations of up to d features are taken into consideration.

Since SVMs are binary classifiers, they cannot handle multi-class classification problems by themselves. Several methods for applying SVMs to multi-class classification have been proposed (Weston and Watkins, 1999; Allwein et al., 2000), and we describe a well known approach called one-versus-rest here. We assume a k-class classification problem. In training, k classifiers $f_i(\mathbf{x})\,(1 \leq i \leq k)$ are created to classify the class i from all other classes,

$$\begin{cases} f_i(\mathbf{x}) \geq +1 & (\mathbf{x} \text{ belongs to the class } i), \\ f_i(\mathbf{x}) \leq -1 & (\text{otherwise}). \end{cases} \tag{2.7}$$

Given a test example $\mathbf{x}$, its class c is determined by the classifier that gives the largest discriminant function value,

$$c \quad = \quad \underset{i}{\operatorname{argmax}}\, f_i(\mathbf{x}). \tag{2.8}$$

2.3. Unknown Word Guessing Using Support Vector Machines

Guessing POS tags of unknown words can be handled as a multi-class classification problem. We conduct the task using SVMs in combination with the one-versus-rest method. We use polynomial kernels with SVMs which are used in many other works (Joachims, 1998; Kudoh and Matsumoto, 2000). In order to predict the POS tag of an unknown word, we use the following features:

(1) POS context: The POS tags of the two words on each side of the unknown word.

(2) Word context: The lexical forms of the two words on each side of the unknown word.

(3) Substrings: The prefixes and suffixes of the unknown word, with up to four characters, and the existence of numerals, capital letters and hyphens in the unknown word.

Let us consider the following example sentence:

```
...  she/PRP  returned/VBD  to/TO  Greenville/(Unknown Word)
two/CD  days/NNS  before/IN  ...
```

The features for the unknown word "`Greenville`" are shown in Table 2.1[1]. These features are the same as those used by Ratnaparkhi (1996), except that we do not use combinations of POS tags because the polynomial kernel can automatically consider them.

In the training phase, SVM classifiers are created for each POS tag using all words in the training data. In the testing phase, POS tags of unknown words are predicted using those classifiers.

[1] "`~`" and "`$`" indicate the beginning and the end of the word respectively.

Table 2.1. Example of Features for Unknown Word Guessing

POS Context	t_{-1} =TO, t_{-2} =VBD, t_{+1} =CD, t_{+2} =NNS
Word Context	w_{-1} =to, w_{-2} =returned, w_{+1} =two, w_{+2} =days
Substrings	^g, ^gr, ^gre, ^gree, e$, le$, lle$, ille$, $\langle Capital \rangle$

2.4. Part-of-Speech Tagging Using Support Vector Machines

In this section, we generalize the unknown word guessing method discussed above in order to handle POS tagging of any given word. In unknown word guessing, the POS tag of an unknown word is predicted using the POS context, the word context and the substrings. We assumed that POS tags of surrounding words of the unknown word are given. This method can be extended to more general POS tagging by predicting the POS tags of all the words in a given sentence. Contrary to what happens in unknown word guessing as a post-processing of POS tagging, the POS tags of succeeding words are usually not known during POS tagging. Therefore, two methods for this task are tested as described in the following subsections.

2.4.1 Using Only the Preceding POS Tags

The first method uses only the preceding POS tags of the considering word. The features used in this model for predicting the POS tag of a word w are:

(1) POS context: The POS tags of the two words preceding w.

(2) Word context: The lexical forms of the two words on each side of w.

(3) Substrings: The prefixes and suffixes of up to four characters of w and the existence of numerals, capital letters and hyphens in w.

In probabilistic models such as Markov models, the generative probabilities of all possible sequences are considered and the most likely path is selected by

the Viterbi algorithm. Since SVMs do not produce probabilities, we employ a deterministic method: The POS tag of each word in a sentence is decided from left to right. This method has the merit of having a small computational cost, but it has the demerit of not using the information of the succeeding POS tags.

A tag dictionary is used to provide the lists of possible POS tags for each known word (a word that appeared in the training data). This dictionary was also used by Ratnaparkhi (1996) to reduce the number of candidate POS tags for known words. For unknown words, all possible POS tags are taken as candidates. Except for this difference, this method processes known words and unknown words in the same way with the same features, requiring no special treatment for unknown words.

2.4.2 Using the Preceding and Succeeding POS Tags

The second method uses the POS tag information on each side of the considering word. The same features as shown in Table 2.1 are used.

In general, the POS tags of the succeeding words are unknown. Roth and Zelenko (1998) addressed the POS tagging in a case with no unknown words, and used a dictionary which has the most frequent POS tag for each word. They used such POS tags for the succeeding words to predict the POS tag of the considering word using a network of linear separators. They reported that 2% accuracy decrease was caused by incorrectly attached POS tags with the dictionary in their method. We use a similar two-pass method without using the dictionary. In the first pass, all POS tags are predicted without using the POS tag information of succeeding words (i.e., using the features discussed in Section 2.4.1). In the second pass, POS tagging is performed using the POS tags predicted in the first pass for the succeeding context (i.e., using the features described in Section 2.3).

The tag dictionary is used in this method, and this method handles known and unknown words in the same way, too.

2.5. Experiments

Experiments of unknown word guessing and POS tagging are performed using the Penn Treebank WSJ corpus, which has 50 POS tag types. We randomly selected

Table 2.2. Statistical Information of Test Data for POS Tagging

Training Tokens	Test Data	
	Number of Known/Unknown Words	Percentage of Unknown Words
1,000	153,492/131,316	46.1%
10,000	218,197/ 66,611	23.4%
100,000	261,786/ 23,022	8.1%
1,000,000	278,535/ 6,273	2.2%

Table 2.3. Performance of Unknown Word Guessing vs. Number of Training Tokens

Training Tokens	SVMs	d	TnT
1,000	69.8%	1	69.4%
10,000	82.3%	2	81.5%
100,000	86.7%	2	83.3%
1,000,000	87.1%	2	84.2%

Table 2.4. Performance of Unknown Word Guessing for Different Sets of Features

Training Tokens	Correct POS	No POS	No Word	No Substrings
1,000	71.1%	64.2%	68.7%	33.7%
10,000	82.9%	75.8%	80.2%	37.1%
100,000	87.5%	80.1%	85.2%	33.8%
1,000,000	88.5%	80.8%	86.0%	30.0%

Table 2.5. Performance of Unknown Word Guessing along Reduction of Features: Features occurring less than or equal to "cutoff" are not taken into consideration.

Cutoff	0	1	2	3	4	10
Accuracy	82.3%	82.5%	82.6%	82.6%	82.5%	81.7%
Number of Features	18,936	7,683	4,854	3,493	2,792	1,314

Table 2.6. Performance of Unknown Word Guessing for Different Kernel Function Parameters

Training Tokens	Degree of Polynomial Kernel			
	1	2	3	4
1,000	69.8%	69.8%	61.2%	34.8%
10,000	82.0%	82.3%	80.3%	79.5%
100,000	85.0%	86.7%	86.0%	84.3%
1,000,000	85.2%	87.1%	N/A	N/A

('N/A' in the table indicates that the value could not be calculated because of the long computation time required)

Table 2.7. Performance of POS Tagging (for Known/Unknown Words)

Training Tokens	SVMs				TnT
	Preceding POS	d	Preceding & Succeeding POS	d	
1,000	83.4%(96.3/68.3)	1	83.9%(96.4/69.3)	1	83.8%(96.0/69.4)
10,000	92.1%(95.5/81.2)	2	92.5%(95.7/82.2)	2	92.3%(95.7/81.5)
100,000	95.6%(96.5/85.7)	2	95.9%(96.7/86.7)	2	95.4%(96.4/83.3)
1,000,000	97.0%(97.3/86.3)	2	97.1%(97.3/86.9)	2	96.6%(96.9/84.2)

Table 2.8. Performance of POS Tagging for Different Sets of Features (for Known/Unknown Words)

Training Tokens	No POS	No Word	No Substrings
1,000	81.3%(96.0/64.2)	83.2%(96.2/68.0)	65.2%(96.2/29.0)
10,000	90.3%(94.7/75.8)	91.9%(95.5/79.9)	80.7%(94.8/34.4)
100,000	93.9%(95.1/80.1)	95.4%(96.4/85.0)	82.4%(87.0/30.7)
1,000,000	95.0%(95.3/80.8)	96.7%(96.9/85.7)	74.4%(75.6/24.1)

sentences in the corpus to construct four training data, each of which contains approximately 1,000, 10,000, 100,000 and 1,000,000 tokens respectively.

Test data for unknown word guessing consists of words that do not appear in the training data. The POS tags on each side of the unknown words were tagged by the POS tagger TnT (Brants, 2000).

Test data for POS tagging consists of about 285,000 tokens differing from the training data. The number of known/unknown words and the percentage of unknown words in the test data are shown in Table 2.2.

The accuracies are compared with TnT, a state-of-the-art POS tagger based on second order Markov models.

We use SVMs with the polynomial kernel and set the soft margin parameter C to 1.

2.5.1 Unknown Word Guessing

The accuracies of the unknown word guessing are shown in Table 2.3 together with the degree of polynomial kernels used in the experiments. Our method has higher accuracies compared to TnT in every training data.

Accuracies with various settings are shown in Table 2.4. The same kernel parameters shown in Table 2.3 are used. The first column shows the cases when the correct POS context on each side of the guessed words is given. From the second to fourth columns, some features are deleted so as to see their contribution to the accuracy. The decrease of the accuracy caused by the errors in POS tagging by TnT is about 1%. Information from substrings (prefixes, suffixes and the

existence of numerals, capital letters and hyphens) plays the most important role in predicting POS tags. On the other hand, the lexical forms of the adjacent words have much less contribution while the POS context of the adjacent words have moderate contribution to the final accuracy.

In general, features that rarely appear in the training data are statistically unreliable, and often decrease the performance of the system. Ratnaparkhi (1996) ignored features that appeared less than 10 times in training data to cope with this problem. We examine the performance when such sparse features are reduced. Table 2.5 shows the results with 10,000 training tokens. Ignoring the features that appeared only a few times improves accuracy slightly, but even if a large number of features are used without cutting-off, SVMs hardly overfit and keep good generalization performance.

Table 2.6 shows the performance when polynomial kernels with different degrees are used. The best degree seems to be 2 for this task.

2.5.2 Part-of-Speech Tagging

The accuracies of POS tagging are shown in Table 2.7. The table shows two cases: One refers to the preceding POS tags only, and the other refers to both preceding and succeeding POS tags with the two pass method. The results show that the performance is comparable to TnT in the first case and better in the second case. Between the first and the second cases, the accuracies for known words are almost equal in the two models, but the accuracies of the first case for unknown words are lower than those of the second case.

Accuracies measured by deleting each feature are shown in Table 2.8. The contribution of each feature has the same tendency as in the unknown word guessing in Section 2.5.1. In relation to features, the biggest difference between our method and TnT is the use of the word context features. Although using a lot of features such as word context is difficult in Markov models, it is easy in SVMs as seen in Section 2.5.1. For small training data, the accuracies in the case without word context are lower than that of TnT. This suggests that one reason for better performance of our method is the use of word context.

2.6. Related Work

One standard approach for English unknown word guessing is to use suffixes or the surrounding context of unknown words (Weischedel et al., 1993; Thede, 1998). Weischedel et al. (1993) estimated the conditional probability of an unknown word w given a tag t using the ending form of w, and the existence of hyphenation and capitalization in w:

$$p(w|t) = p(\text{unknown word}|t)p(\text{capital-feature}|t)p(\text{endings/hyphenation}|t). \quad (2.9)$$

Although this method has the merit of handling unknown words within the framework of probabilistic models, ending forms used in the models such as "-ed" and "-ion" are selected by hand, so applying this method to other languages is not straightforward. Brants (2000) used the linear interpolation of fixed length suffix models for unknown word handling in his POS tagger TnT. This method achieves relatively high accuracy and was reported to be effective in other languages (Džeroski et al., 2000; Megyesi, 2001). Cucerzan and Yarowsky (2000) proposed paradigmatic similarity measures and obtained good results in highly inflectional languages using a large amount of unannotated text. Other methods for unknown word guessing have been studied, including the rule-based method (Mikheev, 1997) and the decision tree-based method (Orphanos and Christodoulakis, 1999).

2.7. Conclusion

In this chapter, we applied SVMs to unknown word guessing and showed that they perform quite well using context and substring information. Furthermore, extending the method to POS tagging, the resulting POS tagger achieved higher accuracy than the state-of-the-art Markov model-based tagger. Comparing to other machine learning algorithms, SVMs have the advantage of considering combinations of features automatically by introducing kernel functions and seldom overfit even with a large set of features. Our methods do not depend on particular characteristics of English, therefore, our methods are applicable to other languages such as German and French. However, for languages like Japanese and

Chinese, it is difficult to apply our methods straightforwardly because words are not separated by spaces in these languages, and segmenting sentences into words is necessary as a preprocessing.

One problem of our methods is computational cost. It takes about 16.5 hours for training with 100,000 tokens and 4 hours for testing 285,000 tokens in POS tagging using POS tags on each side, with an Alpha 21164A 500MHz processor. Although SVMs have good properties and performance, their computational cost is high. It is difficult to train on a large amount of training data. Furthermore, the amount of training data increases, the testing time increases because the number of support vectors increases.

Another point to be improved is the search algorithm for POS tagging. We used a deterministic method as the search algorithm. This method does not consider the overall likelihood of a sentence and considers only local optimality compared to generative probabilistic models. Furthermore, our method outputs only the best answer and cannot output the second or the third best answers. There is a way to translate the value of the discriminant function of SVMs into probabilities (Platt, 1999), which may be applied directly to remedy these problems.

Chapter 3

Guessing Parts-of-Speech of Unknown Words using Global Information

In this chapter, we address utilization of global information for guessing POS tags of unknown words. Although many methods for POS guessing of unknown words use only local information (i.e., word n-gram-wide or sentence-wide features), global information (document-wide features) seems to provide valuable clues for predicting unknown word POS tags. We propose a probabilistic model for guessing POS tags of unknown words using global information, and evaluate it on multiple corpora.

3.1. Introduction

Guessing part-of-speech (POS) tags of unknown words is a difficult task. But it is an important issue both for conducting POS tagging accurately and for creating word dictionaries automatically or semi-automatically. There have been many studies on POS guessing of unknown words (Mori and Nagao, 1996; Mikheev, 1997; Chen et al., 1997; Nagata, 1999b; Orphanos and Christodoulakis, 1999). In most of these previous works, POS tags of unknown words were predicted using only local information, such as lexical forms and POS tags of surrounding words or word-internal features (e.g. suffixes and character types) of the unknown words.

However, this approach has limitations in available information. For example, common nouns and proper nouns are sometimes difficult to distinguish with only the information of a single occurrence because their syntactic functions are almost identical. In English, proper nouns are capitalized and there is generally little ambiguity between common nouns and proper nouns. In Chinese and Japanese, no such convention exists and the problem of the ambiguity is serious. However, if an unknown word with the same lexical form appears in another part with informative local features (e.g. titles of persons), this will give useful clues for guessing the part-of-speech of the ambiguous one, because unknown words with the same lexical form usually have the same part-of-speech. For another example, there is a part-of-speech named *sahen-noun* (verbal noun) in Japanese. Verbal nouns behave as common nouns, except that they are used as verbs when they are followed by a verb "*suru*"; e.g., a verbal noun "*dokusho*" means "reading" and "*dokusho-suru*" is a verb meaning to "read books". It is difficult to distinguish a verbal noun from a common noun if it is used as a noun. However, it will be easy if we know that the word is followed by "*suru*" in another part in the document. This issue was mentioned by Asahara (2003) as a problem of *possibility-based POS tags*. A possibility-based POS tag is a POS tag that represents all the possible properties of the word (e.g., a verbal noun is used as a noun or a verb), rather than a property of each instance of the word. For example, a sahen-noun is actually a noun that can be used as a verb when it is followed by "*suru*". This property cannot be confirmed without observing real usage of the word appearing with "*suru*". Such POS tags may not be identified with only local information of one instance, because the property that each instance has is only one among all the possible properties.

To cope with these issues, we propose a method that uses global information as well as local information for guessing the parts-of-speech of unknown words. With this method, all the occurrences of the unknown words in a document[1] are taken into consideration at once, rather than that each occurrence of the words is processed separately. Thus, the method models the whole document and finds a set of parts-of-speech by maximizing their conditional probability given the

[1]In this chapter, we use the word *document* to denote the whole data consisting of multiple sentences (training corpus or test corpus).

document, rather than independently maximizing the conditional probability of each part-of-speech given each sentence. Global information is known to be useful in other NLP tasks, especially in the named entity recognition task, and several studies successfully used global features (Chieu and Ng, 2002; Finkel et al., 2005).

One potential advantage of our method is its ability to incorporate unlabeled data. Global features can be increased by simply adding unlabeled data into the test data.

Models in which the whole document is taken into consideration need a lot of computation compared to models with only local features. They also cannot process input data one-by-one. Instead, the entire document has to be read before processing. We adopt Gibbs sampling in order to compute the models efficiently, and these models are suitable for offline use such as creating dictionaries from raw text where real-time processing is not necessary but high-accuracy is needed to reduce human labor required for revising automatically analyzed data.

The rest of this chapter is organized as follows: Section 3.2 describes a method for POS guessing of unknown words which utilizes global information. Section 3.3 shows experimental results on multiple corpora. Section 3.4 discusses related work, and Section 3.5 gives conclusions.

3.2. POS Guessing of Unknown Words with Global Information

We handle POS guessing of unknown words as a sub-task of POS tagging. We assume that POS tags of known words are already determined beforehand, and positions in the document where unknown words appear are also identified. Thus, we focus only on prediction of the POS tags of unknown words.

In the rest of this section, we first present a model for POS guessing of unknown words with global information. Next, we show how the test data is analyzed and how the parameters of the model are estimated. A method for incorporating unlabeled data with the model is also discussed.

3.2.1 Probabilistic Model Using Global Information

We attempt to model the probability distribution of the parts-of-speech of all occurrences of the unknown words in a document which have the same lexical form. We suppose that such parts-of-speech have correlation, and the part-of-speech of each occurrence is also affected by its local context. Similar situations to this are handled in physics. For example, let us consider a case where a number of electrons with spins exist in a system. The spins interact with each other, and each spin is also affected by the external magnetic field. In the physical model, if the state of the system is $\mathbf{s}$ and the energy of the system is $E(\mathbf{s})$, the probability distribution of $\mathbf{s}$ is known to be represented by the following Boltzmann distribution:

$$P(\mathbf{s}) = \frac{1}{Z}\exp\{-\beta E(\mathbf{s})\}, \tag{3.1}$$

where β is inverse temperature and Z is a normalizing constant (also called partition function) defined as follows:

$$Z = \sum_{\mathbf{s}} \exp\{-\beta E(\mathbf{s})\}. \tag{3.2}$$

Takamura et al. (2005) applied this model to an NLP task, semantic orientation extraction, and we apply it to POS guessing of unknown words here.

Suppose that unknown words with the same lexical form appear K times in a document. Assume that the number of possible POS tags for unknown words is N, and they are represented by integers from 1 to N. Let t_k denote the POS tag of the kth occurrence of the unknown words, let w_k denote the local context (e.g. the lexical forms and the POS tags of the surrounding words) of the kth occurrence of the unknown words, and let $\mathbf{w}$ and $\mathbf{t}$ denote the sets of w_k and t_k respectively:

$$\begin{aligned} \mathbf{w} &= \{w_1, \cdots, w_K\}, \\ \mathbf{t} &= \{t_1, \cdots, t_K\}, \\ t_k &\in \{1, \cdots, N\}. \end{aligned}$$

$\lambda_{i,j}$ is a weight which denotes strength of the interaction between parts-of-speech i and j, and is symmetric ($\lambda_{i,j} = \lambda_{j,i}$). We define the energy where POS tags of

unknown words given $\mathbf{w}$ are $\mathbf{t}$ as follows:

$$E(\mathbf{t}|\mathbf{w}) = -\left\{\frac{1}{2}\sum_{k=1}^{K}\sum_{\substack{k'=1\\k'\neq k}}^{K}\lambda_{t_k,t_{k'}} + \sum_{k=1}^{K}\log p_0(t_k|w_k)\right\}, \tag{3.3}$$

where $p_0(t|w)$ is an initial distribution (local model) of the part-of-speech t which is calculated with only the local context w, using arbitrary statistical models such as maximum entropy (ME) models. The right hand side of the above equation consists of two components; one represents global interactions between each pair of parts-of-speech, and the other represents the effects of local information.

In this study, we fix the inverse temperature $\beta = 1$. The distribution of $\mathbf{t}$ is then obtained from Equation (3.1), (3.2) and (3.3) as follows:

$$P(\mathbf{t}|\mathbf{w}) = \frac{1}{Z(\mathbf{w})}p_0(\mathbf{t}|\mathbf{w})\exp\left\{\frac{1}{2}\sum_{k=1}^{K}\sum_{\substack{k'=1\\k'\neq k}}^{K}\lambda_{t_k,t_{k'}}\right\}, \tag{3.4}$$

$$Z(\mathbf{w}) = \sum_{\mathbf{t}\in\mathcal{T}(\mathbf{w})}p_0(\mathbf{t}|\mathbf{w})\exp\left\{\frac{1}{2}\sum_{k=1}^{K}\sum_{\substack{k'=1\\k'\neq k}}^{K}\lambda_{t_k,t_{k'}}\right\}, \tag{3.5}$$

$$p_0(\mathbf{t}|\mathbf{w}) \equiv \prod_{k=1}^{K}p_0(t_k|w_k), \tag{3.6}$$

where $\mathcal{T}(\mathbf{w})$ is the set of possible configurations of POS tags given $\mathbf{w}$. The size of $\mathcal{T}(\mathbf{w})$ is N^K, because there are K occurrences of unknown words and each unknown word can have one of N POS tags. The above equations can be rewritten as follows by defining a function $f_{i,j}(\mathbf{t})$:

$$f_{i,j}(\mathbf{t}) \equiv \frac{1}{2}\sum_{k=1}^{K}\sum_{\substack{k'=1\\k'\neq k}}^{K}\delta(t_k,i)\delta(t_{k'},j), \tag{3.7}$$

$$P(\mathbf{t}|\mathbf{w}) = \frac{1}{Z(\mathbf{w})}p_0(\mathbf{t}|\mathbf{w})\exp\left\{\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_{i,j}f_{i,j}(\mathbf{t})\right\}, \tag{3.8}$$

$$Z(\mathbf{w}) = \sum_{\mathbf{t}\in\mathcal{T}(\mathbf{w})}p_0(\mathbf{t}|\mathbf{w})\exp\left\{\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_{i,j}f_{i,j}(\mathbf{t})\right\}, \tag{3.9}$$

where $\delta(i,j)$ is the Kronecker delta:

$$\delta(i,j) = \begin{cases}1 & (i=j),\\ 0 & (i\neq j).\end{cases} \tag{3.10}$$

As shown above, we consider the joint distribution of all the occurrences of the unknown words with the same lexical form in the document, in contrast to conventional approaches which assume independence of the sentences in the document and use the joint distribution of all the words in a sentence. Note that we assume independence between the unknown words with different lexical forms, and each set of the unknown words with the same lexical form is processed separately from the sets of other unknown words.

3.2.2 Decoding

Let us consider how to find the optimal POS tags $\mathbf{t}$ basing on the model, given test data $\mathbf{w}$, an initial distribution $p_0(t|w)$ and a set of model parameters $\Lambda = \{\lambda_{1,1}, \cdots, \lambda_{N,N}\}$. One way to do this is to find a set of POS tags which maximizes $P(\mathbf{t}|\mathbf{w})$ among all possible candidates of $\mathbf{t}$. However, the number of all possible candidates of the POS tags is N^K and the calculation is generally intractable. Dynamic programming cannot be used, because we are considering interactions (dependencies) between POS tags. Therefore, we use a sampling technique and approximate the solution using samples obtained from the probability distribution[2].

We can obtain a solution $\hat{\mathbf{t}} = \{\hat{t}_1, \cdots, \hat{t}_K\}$ as follows:

$$\hat{t}_k = \operatorname*{argmax}_t P_k(t|\mathbf{w}), \tag{3.11}$$

where $P_k(t|\mathbf{w})$ is the marginal distribution of the part-of-speech of the kth occurrence of the unknown words given a set of local contexts $\mathbf{w}$, and is calculated as an expected value over the distribution of the unknown words as follows:

$$\begin{aligned} P_k(t|\mathbf{w}) &= \sum_{\substack{t_1,\cdots,t_{k-1},t_{k+1},\cdots,t_K \\ t_k=t}} P(\mathbf{t}|\mathbf{w}), \\ &= \sum_{\mathbf{t}\in\mathcal{T}(\mathbf{w})} \delta(t_k, t) P(\mathbf{t}|\mathbf{w}). \end{aligned} \tag{3.12}$$

Expected values can be approximately calculated using enough number of samples generated from the distribution (MacKay, 2003). Suppose that $A(\mathbf{x})$ is a function

[2]In later experiments, we calculate exact solutions without approximation when $K \leq 2$.

initialize $\mathbf{t}^{(1)}$
for m **:=** 2 **to** M
  for k **:=** 1 **to** K
    $t_k^{(m)} \sim P(t_k|\mathbf{w}, t_1^{(m)}, \cdots, t_{k-1}^{(m)}, t_{k+1}^{(m-1)}, \cdots, t_K^{(m-1)})$

Figure 3.1. Gibbs Sampling

of a random variable $\mathbf{x}$, $P(\mathbf{x})$ is a distribution of $\mathbf{x}$, and $\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(M)}\}$ are M samples generated from $P(\mathbf{x})$. Then, the expectation of $A(\mathbf{x})$ over $P(\mathbf{x})$ is approximated by the samples:

$$\sum_{\mathbf{x}} A(\mathbf{x})P(\mathbf{x}) \simeq \frac{1}{M}\sum_{m=1}^{M} A(\mathbf{x}^{(m)}). \tag{3.13}$$

Thus, if we have M samples $\{\mathbf{t}^{(1)}, \cdots, \mathbf{t}^{(M)}\}$ generated from the probability distribution $P(\mathbf{t}|\mathbf{w})$, the marginal distribution of each POS tag is approximated as follows:

$$P_k(t|\mathbf{w}) \simeq \frac{1}{M}\sum_{m=1}^{M} \delta(t_k^{(m)}, t). \tag{3.14}$$

Next, we describe how to generate samples from the distribution. We use Gibbs sampling for this purpose. Gibbs sampling is one of the Markov chain Monte Carlo (MCMC) methods, which can generate samples efficiently from high-dimensional probability distributions (Andrieu et al., 2003). The algorithm is shown in Figure 3.1. The algorithm firstly set the initial state $\mathbf{t}^{(1)}$, then one new random variable is sampled at a time from the conditional distribution in which all other variables are fixed, and new samples are created by repeating the process. Gibbs sampling is easy to implement and is guaranteed to converge to the true distribution. The conditional distribution $P(t_k|\mathbf{w}, t_1, \cdots, t_{k-1}, t_{k+1}, \cdots, t_K)$ in Figure 3.1 can be calculated simply as follows:

$$\begin{aligned} & P(t_k|\mathbf{w}, t_1, \cdots, t_{k-1}, t_{k+1}, \cdots, t_K) \\ = & \frac{P(\mathbf{t}|\mathbf{w})}{P(t_1, \cdots, t_{k-1}, t_{k+1}, \cdots, t_K|\mathbf{w})}, \\ = & \frac{P(\mathbf{t}|\mathbf{w})}{\sum_{t_k=1}^{N} P(\mathbf{t}|\mathbf{w})}, \end{aligned}$$

$$
\begin{aligned}
&= \frac{\frac{1}{Z(\mathbf{w})} p_0(\mathbf{t}|\mathbf{w}) \exp\{\frac{1}{2}\sum_{k'=1}^{K}\sum_{k''=1}^{K}\lambda_{t_{k'},t_{k''}} - \frac{1}{2}\sum_{k'=1}^{K}\lambda_{t_{k'},t_{k'}}\}}{\sum_{t_k=1}^{N}\frac{1}{Z(\mathbf{w})} p_0(\mathbf{t}|\mathbf{w}) \exp\{\frac{1}{2}\sum_{k'=1}^{K}\sum_{k''=1}^{K}\lambda_{t_{k'},t_{k''}} - \frac{1}{2}\sum_{k'=1}^{K}\lambda_{t_{k'},t_{k'}}\}}, \\
&= \frac{p_0(t_k|w_k)\exp\{\sum_{\substack{k'=1\\k'\neq k}}^{K}\lambda_{t_{k'},t_k}\}}{\sum_{t_k=1}^{N} p_0(t_k|w_k)\exp\{\sum_{\substack{k'=1\\k'\neq k}}^{K}\lambda_{t_{k'},t_k}\}}, \qquad (3.15)
\end{aligned}
$$

where the last equation is obtained using the following relation:

$$
\begin{aligned}
&\frac{1}{2}\sum_{k'=1}^{K}\sum_{k''=1}^{K}\lambda_{t_{k'},t_{k''}} - \frac{1}{2}\sum_{k'=1}^{K}\lambda_{t_{k'},t_{k'}} \\
&= \frac{1}{2}\left\{\sum_{\substack{k'=1\\k'\neq k}}^{K}\sum_{\substack{k''=1\\k''\neq k}}^{K}\lambda_{t_{k'},t_{k''}} + \sum_{k''=1}^{K}\lambda_{t_k,t_{k''}} + \sum_{\substack{k'=1\\k'\neq k}}^{K}\lambda_{t_{k'},t_k}\right\} - \frac{1}{2}\left\{\sum_{\substack{k'=1\\k'\neq k}}^{K}\lambda_{t_{k'},t_{k'}} + \lambda_{t_k,t_k}\right\}, \\
&= \frac{1}{2}\sum_{\substack{k'=1\\k'\neq k}}^{K}\sum_{\substack{k''=1\\k''\neq k}}^{K}\lambda_{t_{k'},t_{k''}} - \frac{1}{2}\sum_{\substack{k'=1\\k'\neq k}}^{K}\lambda_{t_{k'},t_{k'}} + \frac{1}{2}\left\{\sum_{\substack{k''=1\\k''\neq k}}^{K}\lambda_{t_k,t_{k''}} + \sum_{\substack{k'=1\\k'\neq k}}^{K}\lambda_{t_{k'},t_k}\right\}, \\
&= \left(\frac{1}{2}\sum_{\substack{k'=1\\k'\neq k}}^{K}\sum_{\substack{k''=1\\k''\neq k}}^{K}\lambda_{t_{k'},t_{k''}} - \frac{1}{2}\sum_{\substack{k'=1\\k'\neq k}}^{K}\lambda_{t_{k'},t_{k'}}\right) + \sum_{\substack{k'=1\\k'\neq k}}^{K}\lambda_{t_{k'},t_k}. \qquad (3.16)
\end{aligned}
$$

In later experiments, the number of samples M is set to 100, and the initial state is set to the POS tags which maximizes $p_0(\mathbf{t}|\mathbf{w})$.

The optimal solution obtained by Equation (3.11) maximizes the probability of each POS tag given $\mathbf{w}$, and this kind of approach is known as the maximum posterior marginal (MPM) estimate (Marroquin, 1985). Finkel et al. (2005) used simulated annealing with Gibbs sampling to find a solution in a similar situation. Unlike simulated annealing, this approach does not need to define a cooling schedule. Furthermore, this approach can obtain not only the best solution but also the second best or the other solutions according to $P_k(t|\mathbf{w})$, which are useful when this method is applied to semi-automatic construction of dictionaries because human annotators can check the ranked lists of candidates.

3.2.3 Parameter Estimation

Let us consider how to estimate the parameter $\Lambda = \{\lambda_{1,1}, \cdots, \lambda_{N,N}\}$ in Equation (3.8) from training data consisting of L examples; $\{\langle \mathbf{w}^1, \mathbf{t}^1\rangle, \cdots, \langle \mathbf{w}^L, \mathbf{t}^L\rangle\}$ (i.e., the training data contains L different lexical forms of unknown words). We

define the following objective function $\mathcal{L}_\Lambda$, and find Λ which maximizes $\mathcal{L}_\Lambda$ (the subscript Λ denotes being parameterized by Λ):

$$\begin{aligned}
\mathcal{L}_\Lambda &= \log \prod_{l=1}^{L} P_\Lambda(\mathbf{t}^l|\mathbf{w}^l) + \log P(\Lambda), \\
&= \log \prod_{l=1}^{L} \frac{1}{Z_\Lambda(\mathbf{w}^l)} p_0(\mathbf{t}^l|\mathbf{w}^l) \exp\left\{\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_{i,j} f_{i,j}(\mathbf{t}^l)\right\} + \log P(\Lambda), \\
&= \sum_{l=1}^{L}\left[-\log Z_\Lambda(\mathbf{w}^l) + \log p_0(\mathbf{t}^l|\mathbf{w}^l) + \sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_{i,j} f_{i,j}(\mathbf{t}^l)\right] + \log P(\Lambda).
\end{aligned} \tag{3.17}$$

The partial derivatives of the objective function are:

$$\begin{aligned}
\frac{\partial \mathcal{L}_\Lambda}{\partial \lambda_{i,j}} &= \sum_{l=1}^{L}\left[f_{i,j}(\mathbf{t}^l) - \frac{\partial}{\partial \lambda_{i,j}} \log Z_\Lambda(\mathbf{w}^l)\right] + \frac{\partial}{\partial \lambda_{i,j}} \log P(\Lambda), \\
&= \sum_{l=1}^{L}\left[f_{i,j}(\mathbf{t}^l) - \sum_{\mathbf{t}\in\mathcal{T}(\mathbf{w}^l)} f_{i,j}(\mathbf{t}) P_\Lambda(\mathbf{t}|\mathbf{w}^l)\right] + \frac{\partial}{\partial \lambda_{i,j}} \log P(\Lambda).
\end{aligned} \tag{3.18}$$

We use Gaussian priors (Chen and Rosenfeld, 1999) for $P(\Lambda)$, then the objective function and its derivatives are:

$$\mathcal{L}_\Lambda = \sum_{l=1}^{L}\left[-\log Z_\Lambda(\mathbf{w}^l) + \log p_0(\mathbf{t}^l|\mathbf{w}^l) + \sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_{i,j} f_{i,j}(\mathbf{t}^l)\right] - \frac{1}{2\sigma^2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_{i,j}^2 + C, \tag{3.19}$$

$$\frac{\partial \mathcal{L}_\Lambda}{\partial \lambda_{i,j}} = \sum_{l=1}^{L}\left[f_{i,j}(\mathbf{t}^l) - \sum_{\mathbf{t}\in\mathcal{T}(\mathbf{w}^l)} f_{i,j}(\mathbf{t}) P_\Lambda(\mathbf{t}|\mathbf{w}^l)\right] - \frac{1}{\sigma^2}\lambda_{i,j}. \tag{3.20}$$

where C is a constant and σ is set to 1 in later experiments. The optimal Λ can be obtained by quasi-Newton methods using the above $\mathcal{L}_\Lambda$ and $\frac{\partial \mathcal{L}_\Lambda}{\partial \lambda_{i,j}}$, and we use L-BFGS (Liu and Nocedal, 1989) for this purpose[3]. However, the calculation is intractable because $Z_\Lambda(\mathbf{w}^l)$ (see Equation (3.9)) in Equation (3.19) and a term in Equation (3.20) contain summations over all the possible POS tags. To cope with the problem, we use the sampling technique again for the calculation, as

[3]In later experiments, L-BFGS often did not converge completely because we used approximation with Gibbs sampling, and we stopped iteration of L-BFGS in such cases.

suggested by Rosenfeld et al. (2001). $Z_\Lambda(\mathbf{w}^l)$ can be approximated using M samples $\{\mathbf{t}^{(1)}, \cdots, \mathbf{t}^{(M)}\}$ generated from $p_0(\mathbf{t}|\mathbf{w}^l)$:

$$\begin{aligned} Z_\Lambda(\mathbf{w}^l) &= \sum_{\mathbf{t} \in \mathcal{T}(\mathbf{w}^l)} p_0(\mathbf{t}|\mathbf{w}^l) \exp\left\{ \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_{i,j} f_{i,j}(\mathbf{t}) \right\}, \\ &\simeq \frac{1}{M} \sum_{m=1}^{M} \exp\left\{ \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_{i,j} f_{i,j}(\mathbf{t}^{(m)}) \right\}. \end{aligned} \tag{3.21}$$

The term in Equation (3.20) can also be approximated using M samples $\{\mathbf{t}^{(1)}, \cdots, \mathbf{t}^{(M)}\}$ generated from $P_\Lambda(\mathbf{t}|\mathbf{w}^l)$ with Gibbs sampling:

$$\sum_{\mathbf{t} \in \mathcal{T}(\mathbf{w}^l)} f_{i,j}(\mathbf{t}) P_\Lambda(\mathbf{t}|\mathbf{w}^l) \simeq \frac{1}{M} \sum_{m=1}^{M} f_{i,j}(\mathbf{t}^{(m)}). \tag{3.22}$$

In later experiments, the initial state $\mathbf{t}^{(1)}$ in Gibbs sampling is set to the gold standard tags in the training data.

3.2.4 Use of Unlabeled Data

In our model, unlabeled data can be easily used by simply concatenating the test data and the unlabeled data, and decoding them in the testing phase. Intuitively, if we increase the amount of the test data, test examples with informative local features may increase. The POS tags of such examples can be easily predicted, and they are used as global features in prediction of other examples. Thus, this method uses unlabeled data in only the testing phase, and the training phase is the same as the case with no unlabeled data.

3.3. Experiments

3.3.1 Data and Procedure

We use eight corpora for our experiments; the Penn Chinese Treebank corpus 2.0 (CTB), a part of the PFR corpus (PFR), the EDR corpus (EDR), the Kyoto University corpus version 2 (KUC), the RWCP corpus (RWC), the GENIA corpus 3.02p (GEN), the SUSANNE corpus (SUS) and the Penn Treebank WSJ corpus

Table 3.1. Statistical Information of Corpora

Corpus (Lang.)	# of POS (Open Class)	# of Tokens (# of Unknown Words) [partition in the corpus]		
		Training	Test	Unlabeled
CTB (C)	34 (28)	84,937 [sec. 1–270]	7,980 (749) [sec. 271–300]	6,801 [sec. 301–325]
PFR (C)	42 (39)	304,125 [Jan. 1–Jan. 9]	370,627 (27,774) [Jan. 10–Jan. 19]	445,969 [Jan. 20–Jan. 31]
EDR (J)	15 (15)	2,550,532 [$id = 4n + 0$, $id = 4n + 1$]	1,280,057 (24,178) [$id = 4n + 2$]	1,274,458 [$id = 4n + 3$]
KUC (J)	40 (36)	198,514 [Jan. 1–Jan. 8]	31,302 (2,477) [Jan. 9]	41,227 [Jan. 10]
RWC (J)	66 (55)	487,333 [1–10,000th sentences]	190,571 (11,177) [10,001–14,000th sentences]	210,096 [14,001–18,672th sentences]
GEN (E)	47 (36)	243,180 [1–10,000th sentences]	123,386 (7,775) [10,001–15,000th sentences]	134,380 [15,001–20,546th sentences]
SUS (E)	125 (90)	74,902 [sec. A01–08, G01–08, J01–08, N01–08]	37,931 (5,760) [sec. A09–12, G09–12, J09–17, N09–12]	37,593 [sec. A13–20, G13–22, J21–24, N13–18]
WSJ (E)	45 (33)	912,344 [sec. 0–18]	129,654 (4,253) [sec. 22–24]	131,768 [sec. 19–21]

(WSJ). All the corpora are POS tagged corpora in Chinese(C), English(E) or Japanese(J), and they are split into three portions; training data, test data and unlabeled data. The unlabeled data is used in experiments of semi-supervised learning, and POS tags of unknown words in the unlabeled data are eliminated. Table 3.1 summarizes detailed information about the corpora we used: the language, the number of POS tags, the number of open class tags (POS tags that unknown words can have, described later), the sizes of training, test and unlabeled data, and the splitting method of them. For the test data and the unlabeled data, unknown words are defined as words that do not appear in the training data. The number of unknown words in the test data of each corpus is shown in Table 3.1, parentheses. Accuracy of POS guessing of unknown words is calculated based on how many words among them are correctly POS-guessed.

Figure 3.2 shows the procedure of the experiments. We split the training data into two parts; the first half as sub-training data 1 and the latter half as sub-training data 2 (Figure 3.2, *1). Then, we check the words that appear in the

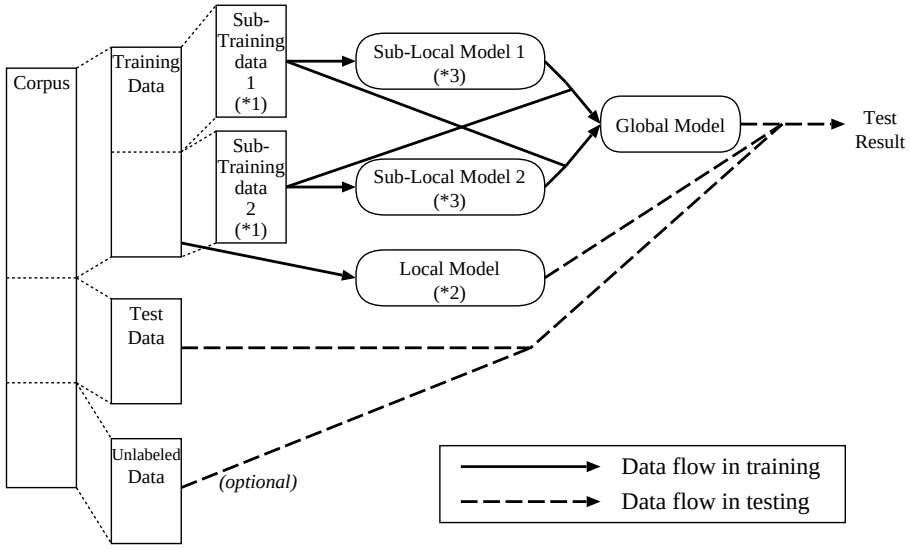

Figure 3.2. Experimental Procedure

sub-training data 1 but not in the sub-training data 2, or vice versa. We handle these words as (*pseudo*) unknown words in the training data. Such (two-fold) cross-validation is necessary to make training examples that contain unknown words[4]. POS tags that these pseudo unknown words have are defined as open class tags, and only the open class tags are considered as candidate POS tags for unknown words in the test data (i.e., N is equal to the number of the open class tags). In the training phase, we need to estimate two types of parameters; local model (parameters), which is necessary to calculate $p_0(t|w)$, and global model (parameters), i.e., $\lambda_{i,j}$. The local model parameters are estimated using all the

[4]A major method for generating such pseudo unknown words is to collect the words that appear only once in a corpus (Nagata, 1999b). These words are called *hapax legomena* and known to have similar characteristics to real unknown words (Baayen and Sproat, 1996). These words are interpreted as being collected by the leave-one-out technique (which is a special case of cross-validation) as follows: One word is picked from the corpus and the rest of the corpus is considered as training data. The picked word is regarded as an unknown word if it does not exist in the training data. This procedure is iterated for all the words in the corpus. However, this approach is not applicable to our experiments because those words that appear only once in the corpus do not have global information and are useless for learning the global model, so we use the two-fold cross validation method.

training data (Figure 3.2, *2). Local model parameters and training data are necessary to estimate the global model parameters, but the global model parameters cannot be estimated from the same training data from which the local model parameters are estimated. In order to estimate the global model parameters, we firstly train sub-local models 1 and 2 from the sub-training data 1 and 2 respectively (Figure 3.2, *3). The sub-local models 1 and 2 are used for calculating $p_0(t|w)$ of unknown words in the sub-training data 2 and 1 respectively, when the global model parameters are estimated from the entire training data. In the testing phase, $p_0(t|w)$ of unknown words in the test data are calculated using the local model parameters which are estimated from the entire training data, and test results are obtained using the global model with the local model.

Global information cannot be used for unknown words whose lexical forms appear only once in the training or test data, so we process only non-unique unknown words (unknown words whose lexical forms appear more than once) using the proposed model. In the testing phase, POS tags of unique unknown words are determined using only the local information, by choosing POS tags which maximize $p_0(t|w)$.

Unlabeled data can be optionally used for semi-supervised learning. In that case, the test data and the unlabeled data are concatenated, and the best POS tags which maximize the probability of the mixed data are searched.

3.3.2 Initial Distribution

In our method, the initial distribution $p_0(t|w)$ is used for calculating the probability of t given local context w (Equation (3.8)). We use maximum entropy (ME) models for the initial distribution. $p_0(t|w)$ is calculated by ME models as follows (Berger et al., 1996):

$$p_0(t|w) = \frac{1}{Y(w)} \exp\left\{\sum_{h=1}^{H} \alpha_h g_h(w,t)\right\}, \tag{3.23}$$

$$Y(w) = \sum_{t=1}^{N} \exp\left\{\sum_{h=1}^{H} \alpha_h g_h(w,t)\right\}, \tag{3.24}$$

where $g_h(w,t)$ is a binary feature function. We assume that each local context w contains the following information about the unknown word:

- The POS tags of the two words on each side of the unknown word: $\tau_{-2}, \tau_{-1}, \tau_{+1}, \tau_{+2}$.[5]
- The lexical forms of the unknown word itself and the two words on each side of the unknown word: $\omega_{-2}, \omega_{-1}, \omega_0, \omega_{+1}, \omega_{+2}$.
- The character types of all the characters composing the unknown word: $y_1, \cdots, y_{|\omega_0|}$. We use six character types: alphabet, numeral (Arabic and Chinese numerals), symbol, Kanji (Chinese character), Hiragana (Japanese script) and Katakana (Japanese script).

A feature function $g_h(w, t)$ returns 1 if w and t satisfy certain conditions, and otherwise 0, for example:

$$g_{123}(w,t) \;=\; \begin{cases} 1 & (\omega_{-1} = \text{“President”} \text{ and } \tau_{-1} = \text{“NNP”} \text{ and } t = 5), \\ 0 & (\text{otherwise}). \end{cases}$$

The features we use are shown in Table 3.2, which are based on the features used by Ratnaparkhi (1996) and Uchimoto et al. (2001).

The parameters α_h in Equation (3.23) are estimated using all the words in the training data whose POS tags are the open class tags. That is, words which are not pseudo-unknown words are also used in parameter estimation of the local model, in order to use as many training examples as possible.

3.3.3 Experimental Results

The results are shown in Table 3.3. In the table, *local*, *local+global* and *local+global w/ unlabeled* indicate that the results were obtained using only local information, local and global information, and local and global information with the extra unlabeled data, respectively. The results using only local information were obtained by choosing POS tags $\hat{\mathbf{t}} = \{\hat{t}_1, \cdots, \hat{t}_K\}$ which maximize the probabilities of the local model:

$$\hat{t}_k \;=\; \operatorname*{argmax}_t p_0(t|w_k). \tag{3.25}$$

[5]In both the training and the testing phases, POS tags of known words are given from the corpora. When these surrounding words contain unknown words, their POS tags are represented by a special tag *Unk*.

Table 3.2. Features Used for Initial Distribution

Language	Features
English	Prefixes of ω_0 up to four characters
	Suffixes of ω_0 up to four characters
	ω_0 contains Arabic numerals
	ω_0 contains uppercase characters
	ω_0 contains hyphens
Chinese Japanese	Prefixes of ω_0 up to two characters
	Suffixes of ω_0 up to two characters
	y_1
	$y_{\lvert\omega_0\rvert}$
	y_1 & $y_{\lvert\omega_0\rvert}$
	$\bigcup_{i=1}^{\lvert\omega_0\rvert}\{y_i\}$ (Set of character types)
(common)	$\lvert\omega_0\rvert$ (Length of ω_0)
	τ_{-1}
	τ_{+1}
	τ_{-2} & τ_{-1}
	τ_{+1} & τ_{+2}
	τ_{-1} & τ_{+1}
	ω_{-1} & τ_{-1}
	ω_{+1} & τ_{+1}
	ω_{-2} & τ_{-2} & ω_{-1} & τ_{-1}
	ω_{+1} & τ_{+1} & ω_{+2} & τ_{+2}
	ω_{-1} & τ_{-1} & ω_{+1} & τ_{+1}

Table 3.3. Results of POS Guessing of Unknown Words

Corpus (Lang.)	Accuracy for Unknown Words (# of Errors) [p-value] ⟨# of Non-unique Unknown Words⟩		
	local	local+global	local+global w/ unlabeled
CTB (C)	0.7423 (193)	0.7717 (171) [0.0000] ⟨344⟩	0.7704 (172) [0.0001] ⟨361⟩
PFR (C)	0.6499 (9723)	0.6690 (9193) [0.0000] ⟨16019⟩	0.6785 (8930) [0.0000] ⟨18861⟩
EDR (J)	0.9639 (874)	0.9643 (863) [0.1775] ⟨4903⟩	0.9651 (844) [0.0034] ⟨7770⟩
KUC (J)	0.7501 (619)	0.7634 (586) [0.0000] ⟨788⟩	0.7562 (604) [0.0872] ⟨936⟩
RWC (J)	0.7699 (2572)	0.7785 (2476) [0.0000] ⟨5044⟩	0.7787 (2474) [0.0000] ⟨5878⟩
GEN (E)	0.8836 (905)	0.8837 (904) [1.0000] ⟨4094⟩	0.8863 (884) [0.0244] ⟨4515⟩
SUS (E)	0.7934 (1190)	0.7957 (1177) [0.1878] ⟨3210⟩	0.7979 (1164) [0.0116] ⟨3583⟩
WSJ (E)	0.8345 (704)	0.8368 (694) [0.0162] ⟨1412⟩	0.8352 (701) [0.7103] ⟨1627⟩

Table 3.4. Accuracy for Non-unique Unknown Words

Corpus (Lang.)	Accuracy for Non-unique Unknown Words (# of Errors)	
	local	local+global
CTB (C)	0.8052 (67)	0.8692 (45)
PFR (C)	0.6243 (6019)	0.6573 (5489)
EDR (J)	0.9513 (239)	0.9535 (228)
KUC (J)	0.7424 (203)	0.7843 (170)
RWC (J)	0.7427 (1298)	0.7617 (1202)
GEN (E)	0.8894 (453)	0.8896 (452)
SUS (E)	0.8140 (597)	0.8181 (584)
WSJ (E)	0.8768 (174)	0.8839 (164)

Table 3.5. Results of Multiple Trials and Comparison to Simulated Annealing

Corpus (Lang.)	local	Mean±Standard Deviation	
		Marginal	S.A.
CTB (C)	0.7423	0.7696±0.0021	0.7682±0.0028
PFR (C)	0.6499	0.6707±0.0010	0.6712±0.0014
EDR (J)	0.9639	0.9644±0.0001	0.9645±0.0001
KUC (J)	0.7501	0.7595±0.0031	0.7612±0.0018
RWC (J)	0.7699	0.7777±0.0017	0.7772±0.0020
GEN (E)	0.8836	0.8841±0.0009	0.8840±0.0007
SUS (E)	0.7934	0.7997±0.0038	0.7995±0.0034
WSJ (E)	0.8345	0.8366±0.0013	0.8360±0.0021

The table shows the accuracies, the numbers of errors, the p-values of McNemar's test against the results using only local information, and the numbers of non-unique unknown words in the test data. Table 3.4 shows accuracies calculated for non-unique unknown words in the test data, that is, how many words were correctly POS-guessed among the non-unique unknown words.

In the CTB, PFR, KUC, RWC and WSJ corpora, the accuracies were improved using global information (statistically significant at $p < 0.05$), compared to the accuracies obtained using only local information. The increases of the accuracies on the English corpora (the GEN and SUS corpora) were small. Table 3.6 shows the increased/decreased number of correctly tagged words using global information in the PFR, RWC and SUS corpora. In the PFR (Chinese) and RWC (Japanese) corpora, many proper nouns were correctly tagged using global information. In Chinese and Japanese, proper nouns are not capitalized, therefore proper nouns are difficult to distinguish from common nouns with only local information. One reason that only the small increases were obtained with global information in the English corpora seems to be the low ambiguities of proper nouns. Many verbal nouns in PFR and a few sahen-nouns (Japanese verbal nouns) in RWC were also correctly tagged using global information. When the unlabeled data was used, the number of non-unique words increased. Compared with the case without the unlabeled data, the accuracies increased in several corpora but decreased in the CTB, KUC and WSJ corpora.

Since our method uses Gibbs sampling in the training and the testing phases, the results are affected by the sequences of random numbers used in the sampling. In order to investigate the influence, we conduct 10 trials with different sequences of pseudo random numbers. We also conduct experiments using simulated annealing in decoding, as conducted by Finkel et al. (2005) for information extraction. We increase inverse temperature β in Equation (3.1) from $\beta = 1$ to $\beta \approx \infty$ with the linear cooling schedule. The results are shown in Table 3.5. The table shows the mean values and the standard deviations of the accuracies for the 10 trials, and *Marginal* and *S.A.* mean that decoding is conducted using Equation (3.11) and simulated annealing respectively. The variances caused by random numbers and the differences of the accuracies between *Marginal* and *S.A.* are relatively small.

Table 3.6. Ordered List of Increased/Decreased Number of Correctly Tagged Words

PFR (Chinese)	
+162	vn (verbal noun)
+150	ns (place name)
+86	nz (other proper noun)
+85	j (abbreviation)
+61	nr (personal name)
...	...
−26	m (numeral)
−100	v (verb)
RWC (Japanese)	
+33	noun-proper noun-person name-family name
+32	noun-proper noun-place name
+28	noun-proper noun-organization name
+17	noun-proper noun-person name-first name
+6	noun-proper noun
+4	noun-sahen noun
...	...
−2	noun-proper noun-place name-country name
−29	noun
SUS (English)	
+13	NP (proper noun)
+6	JJ (adjective)
+2	VVD (past tense form of lexical verb)
+2	NNL (locative noun)
+2	NNJ (organization noun)
...	...
−2	VVN (past participle form of lexical verb)
−3	NN (common noun)
−6	NNU (unit-of-measurement noun)

3.4. Related Work

Several studies concerning the use of global information have been conducted, especially in named entity recognition, which is a similar task to POS guessing of unknown words. Chieu and Ng (2002) conducted named entity recognition using global features as well as local features. In their ME model-based method, some global features were used such as "when the word appeared first in a position other than the beginning of sentences, the word was capitalized or not". These global features are static and can be handled in the same manner as local features, therefore Viterbi decoding was used. The method is efficient but does not handle interactions between labels.

Finkel et al. (2005) proposed a method incorporating non-local structure for information extraction. They attempted to use *label consistency* of named entities, which is the property that named entities with the same lexical form tend to have the same label. They defined two probabilistic models; a local model based on conditional random fields and a global model based on log-linear models. Then the final model was constructed by multiplying these two models, which can be seen as (unnormalized) log-linear interpolation (Klakow, 1998) of the two models which are weighted equally. In their method, interactions between labels in the whole document were considered, and they used Gibbs sampling and simulated annealing for decoding. Our model is largely similar to their model. However, in their method, parameters of the global model were estimated using relative frequencies of labels or were selected by hand, while in our method, global model parameters are estimated from training data so as to fit to the data according to the objective function.

One approach for incorporating global information in natural language processing is to utilize consistency of labels, and such an approach have been used in other tasks. Takamura et al. (2005) proposed a method based on the spin models in physics for extracting semantic orientations of words. In the spin models, each electron has one of two states, *up* or *down*, and the models give probability distribution of the states. The states of electrons interact with each other and neighboring electrons tend to have the same spin. In their method, semantic orientations (*positive* or *negative*) of words are regarded as states of spins, in order to model the property that the semantic orientation of a word tends to have the

same orientation as words in its gloss. The mean field approximation was used for inference in their method.

Yarowsky (1995) studied a method for word sense disambiguation using unlabeled data. Although no probabilistic models were considered explicitly in the method, they used the property of label consistency named "one sense per discourse" for unsupervised learning together with local information named "one sense per collocation".

There exist other approaches using global information which do not necessarily aim to use label consistency. Rosenfeld et al. (2001) proposed whole-sentence exponential language models. The method calculates the probability of a sentence s as follows:

$$P(s) = \frac{1}{Z} p_0(s) \exp\left\{ \sum_i \lambda_i f_i(s) \right\},$$

where $p_0(s)$ is an initial distribution of s and any language models such as trigram models can be used for this. $f_i(s)$ is a feature function and can handle sentence-wide features. Note that if we regard $f_{i,j}(\mathbf{t})$ in our model (Equation (3.7)) as a feature function, Equation (3.8) is essentially the same form as the above model. Their models can incorporate any sentence-wide features including syntactic features obtained by shallow parsers. They attempted to use Gibbs sampling and other sampling methods for inference, and model parameters were estimated from training data using the generalized iterative scaling algorithm with the sampling methods. Although they addressed modeling of whole sentences, the method can be directly applied to modeling of whole documents which allows us to incorporate unlabeled data easily and naturally as we have discussed. This approach, modeling whole wide-scope contexts with log-linear models and using sampling methods for inference, gives us an expressive framework and will be applied to other tasks.

3.5. Conclusion

In this chapter, we presented a method for guessing parts-of-speech of unknown words using global information as well as local information. The method models a whole document by considering interactions between POS tags of unknown words

with the same lexical form. Parameters of the model are estimated from training data using Gibbs sampling. Experimental results showed that the method improves accuracies of POS guessing of unknown words especially for Chinese and Japanese. We also applied the method to semi-supervised learning, but the results were not consistent and there is some room for improvement.

In this study, we focused on POS guessing of unknown words. However, in Chinese and Japanese, words are not separated by spaces. Segmentation of unknown words is another important task, and applying the method to it is left as future work.

Chapter 4

Word Segmentation using Word-level and Character-level Information

In Chinese and Japanese word segmentation, handling of unknown words is a large issue, because identifying unknown words is difficult in these languages and they often decrease accuracy of word segmentation. In this chapter, we propose a hybrid method for Chinese and Japanese word segmentation which combines a Markov model-based method and a character-based tagging method. While the word-based Markov model method has difficulties in handling unknown words, the character-based tagging method performs worse in processing known words compared with other methods. To compensate the weaknesses of these approaches, we propose a combined method of these two. Experiments on word segmentation are conducted with multiple Chinese and Japanese corpora, showing that the proposed method achieves higher performance than most of previous methods. We also apply the hybrid method to Korean morphological analysis.

4.1. Introduction

Word segmentation for Chinese and Japanese is an important and difficult task[1]. These Asian languages have no explicit delimiters between words such as spaces in English, and most natural language processing applications for these languages need word segmentation in the first place. Sentences often have multiple possible segmentations, and word segmentation systems must resolve the ambiguity.

One of the problems which make word segmentation more difficult is existence of unknown (out-of-vocabulary) words. Unknown words are defined as words that are not registered in dictionaries of the word segmentation system. The word segmentation system has no knowledge about these unknown words and identifying word boundaries for such words is difficult, because new words with any length may appear in any position in a given sentence in principle. Word segmentation accuracy for unknown words is usually much lower than that for known words, and unknown words are major sources of errors of word segmentation. The unknown word problem can be avoided by preparing a large dictionary. Several studies (Mori and Nagao, 1996; Luo and Song, 2004) have been conducted to collect new words automatically from a large amount of raw corpora in order to enlarge the dictionary. However, preparing a sufficiently large dictionary is difficult because new words, especially proper nouns, are created all the time, and online unknown word processing is necessary for robust word segmentation.

Many studies have been conducted for Chinese and Japanese word segmentation including rule-based and statistical methods. One well known rule-based method is the deterministic (forward) maximum matching method. The method segments sentences basing on a heuristic rule using a word dictionary. A given sentence is scanned from the beginning, and the longest matching word in the dictionary is segmented deterministically. This simple method does not have high accuracy mainly because of unknown words, but has been used in combination with other methods in some studies (Wong and Chan, 1996; Goh et al., 2004).

One popular method for word segmentation is the Markov models and its variants (Nagata, 1994; Sun et al., 1997; Kurohashi and Nagao, 1998; Matsumoto

[1]In this chapter, we do not make a distinction between Chinese word segmentation and Japanese word segmentation particularly, because both can be handled in the same way, and our proposed method does not depend on either language.

et al., 2001; Zhang et al., 2003). The method constructs a word lattice consisting of possible hypotheses using a word dictionary for a given sentence, and chooses the best answer among them basing on Markov models. This method needs exceptional processing for handling unknown words and various methods have been used, e.g., a hand-crafted rule-based method (Kurohashi and Nagao, 1998; Matsumoto et al., 2001), a character n-gram models (Nagata, 1994) and HMMs with character roles (Zhang et al., 2003).

The character-based tagging method is a character-based statistical method (Yamamoto and Masuyama, 1997; Xue, 2003; Asahara et al., 2003; Peng et al., 2004). The method conducts word segmentation without word dictionaries by using not words but characters as a unit, and unknown words are handled in the same way as known words.

Previous studies show that the word-based Markov models have difficulties in handling unknown words, and the character-based tagging method performs worse in handling known words compared with other methods. In this chapter, we propose a hybrid method for Chinese and Japanese word segmentation, which utilizes both word-level and character-level information. Word-level information is useful for analysis of known words, and character-level information is useful for analysis of unknown words. We use these two types of information at the same time in order to obtain high overall performance.

The rest of this chapter is organized as follows: Section 4.2 describes previous work on Chinese and Japanese word segmentation on which our method is based. Section 4.3 introduces the hybrid method which combines word-level and character-level processing. Section 4.5 shows experimental results and analysis of Chinese and Japanese word segmentation. Section 4.6 discusses related work and the hybrid method, and Section 4.7 gives conclusions.

4.2. Previous Work on Word Segmentation

Our method is based on two existing methods; the Markov model-based method and the character-based tagging method. This section explains these two methods in detail.

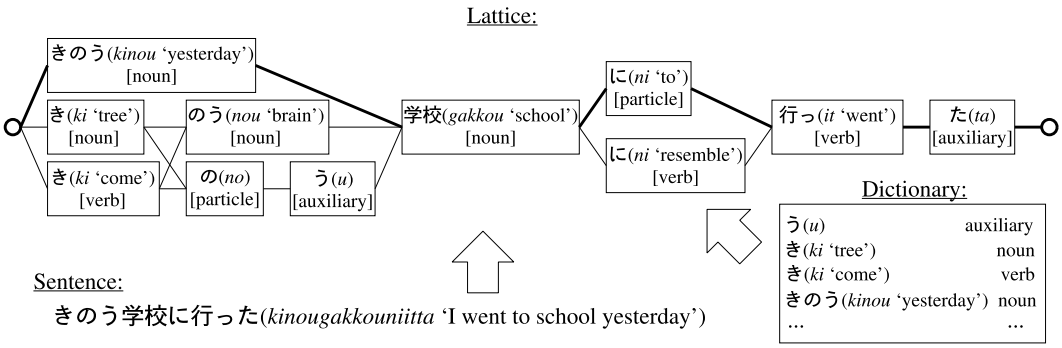

Figure 4.1. Example of a Lattice Used in the Markov Model-Based Method: Each node represents a candidate word and the thick lines indicate the correct path.

4.2.1 The Markov Model-based Method

Word-based Markov models are widely used in English part-of-speech (POS) tagging (Charniak et al., 1993; Brants, 2000). This method identifies a POS-tag sequence $T = t_1, \cdots, t_n$, given an English sentence as a word sequence $W = w_1, \cdots, w_n$, where n is the number of words in the sentence. The method assumes that each word has a state which is identical to the POS of the word and the sequence of states is a Markov chain. A state t transits to another state s with probability $P(s|t)$, and outputs a word w with probability $P(w|t)$. From such assumptions, the probability to generate a word sequence W with POS-tags T is calculated as follows:

$$\begin{aligned} P(W,T) &= \prod_{i=1}^{n} P(w_i t_i | w_0 t_0 \cdots w_{i-1} t_{i-1}), \\ &= \prod_{i=1}^{n} P(w_i | w_0 t_0 \cdots w_{i-1} t_{i-1} t_i) P(t_i | w_0 t_0 \cdots w_{i-1} t_{i-1}), \\ &\simeq \prod_{i=1}^{n} P(w_i|t_i) P(t_i|t_{i-1}), \end{aligned} \tag{4.1}$$

where $w_0(t_0)$ is a special word(part-of-speech) representing the beginning of the sentence. Given a word sequence W, its most likely POS sequence $\hat{T}$ can be found as follows:

$$\hat{T} = \operatorname*{argmax}_T P(T|W),$$

$$
\begin{aligned}
&= \operatorname*{argmax}_T \frac{P(W,T)}{P(W)}, \\
&= \operatorname*{argmax}_T P(W,T), \\
&\simeq \operatorname*{argmax}_T \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}). \qquad (4.2)
\end{aligned}
$$

The equation above can be solved efficiently with the Viterbi algorithm (Rabiner and Juang, 1993).

This method is also used in Chinese and Japanese with some modifications. Because each word in a sentence is not separated explicitly in Chinese and Japanese, both segmentation of words and identification of POS-tags of the words must be done simultaneously. Given a sentence S, its most likely word sequence $\hat{W}$ and POS sequence $\hat{T}$ can be found as follows:

$$
\begin{aligned}
\langle \hat{W}, \hat{T} \rangle &= \operatorname*{argmax}_{W,T} P(W,T|S), \\
&= \operatorname*{argmax}_{W \in \mathcal{W}(S),T} P(W,T), \\
&\simeq \operatorname*{argmax}_{W \in \mathcal{W}(S),T} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}), \qquad (4.3)
\end{aligned}
$$

where $\mathcal{W}(S)$ is the set of all possible segments of S (i.e. $\mathcal{W}(S) = \{W | w_1 \cdots w_n = S\}$). The equation above can be solved using the Viterbi algorithm as well.

The possible segments of a given sentence are represented by a lattice. Figure 4.1 shows an example.

In summary, the word-based Markov models conduct word segmentation and POS tagging as follows:

1) Given a sentence, a lattice consisting of possible segments of the sentence is constructed using a word dictionary.

2) The best path on the lattice which maximizes Equation (4.3) is chosen.

This Markov model-based method achieves high accuracy for known words with low computational cost, and many Japanese word segmentation systems adopt it (Kurohashi and Nagao, 1998; Matsumoto et al., 2001). However, the Markov model-based method cannot handle unknown words easily. In the constructing process of a lattice, only known words are dealt with by dictionary

Table 4.1. The **BIES** Tag Set

Tag	Description
B	The character is at the beginning of a word.
I	The character is at the middle of a word.
E	The character is at the end of a word.
S	The character forms a word.

Sentence: き の う | 学 校 | に | 行 っ | た
POC Tag: B I E B E S B E S

Figure 4.2. Example of the Character-based Tagging Method: Word boundaries are indicated by vertical lines ('|').

lookup and unknown words must be handled with other methods. Many practical word segmentation systems add candidates of unknown words to the lattice. The candidates of unknown words are often generated by heuristic rules based on character types (Kurohashi and Nagao, 1998; Matsumoto et al., 2001) or statistical word models which predict the probabilities for any strings to be unknown words (Sproat et al., 1996; Nagata, 1999b). However, such heuristic rules or statistical word models must be carefully designed for a specific language, and have difficulties in handling a wide variety of unknown words.

4.2.2 The Character-based Tagging Method

This method carries out word segmentation by tagging each character in a given sentence, and the tags indicate word-internal positions of the characters. We call such tags position-of-character (POC) tags (Xue, 2003) in this chapter. Several POC-tag sets have been studied (Tjong Kim Sang and Veenstra, 1999; Sekine et al., 1998), and we use the **BIES** tag set[2] shown in Table 4.1 [3].

[2]'**B**, **I**, **E** and **S**' tags are also called '**OP-CN**, **CN-CN**, **CN-CL** and **OP-CL**' tags (Sekine et al., 1998) or '**LL**, **MM**, **RR** and **LR**' tags (Xue, 2003).

[3]We tried to use **IOB2** tag set (Tjong Kim Sang and Veenstra, 1999) in preliminary experiments, but the accuracy was slightly lower than that with the **BIES** tag set.

Figure 4.2 shows an example of POC tagging for the same sentence as in Figure 4.1. The POC-tags can represent word boundaries for any sentences, and the word segmentation task can be reformulated as the POC tagging task. The POC tagging task can be solved using general machine learning techniques such as Markov models (Yamamoto and Masuyama, 1997), maximum entropy (ME) models (Xue, 2003), support vector machines (Asahara et al., 2003) and conditional random fields (Peng et al., 2004). This approach, finding chunks by tagging a position-indicating tag to each composing element, is also used in base-NP chunking (Ramshaw and Marcus, 1995; Tjong Kim Sang and Veenstra, 1999) and named entity recognition (Sekine et al., 1998) as well as word segmentation.

This character-based tagging method can easily handle unknown words, because known words and unknown words are treated equally without word dictionaries, and no other exceptional processing is necessary. However, the method is not used widely in practical systems, because the method is difficult to utilize information in word dictionaries even if they are available, and additional processing is needed if POS tagging as well as word segmentation is necessary, though the Markov model-based method can conduct both at the same time.

4.3. Word and Character-based Hybrid Method

We saw two methods for word segmentation in the previous section. In previous studies, it is observed that the Markov model-based method has high overall accuracy but has difficulties in handling unknown words, and the character-based tagging method has high accuracy for unknown words but lower accuracy for known words (Yoshida et al., 2003; Xue, 2003; Sproat and Emerson, 2003). This seems natural considering the properties of these methods: Words are used as a processing unit in the Markov model-based method, and therefore much information about known words (e.g., POS or word bigram probability) can be used. However, this method cannot handle unknown words directly and performs worse without precise models for unknown words. On the other hand, characters are used as a unit in the character-based tagging method. In general, the number of characters is finite and far smaller than the number of words which continuously increases. Thus the character-based tagging method may be robust for unknown

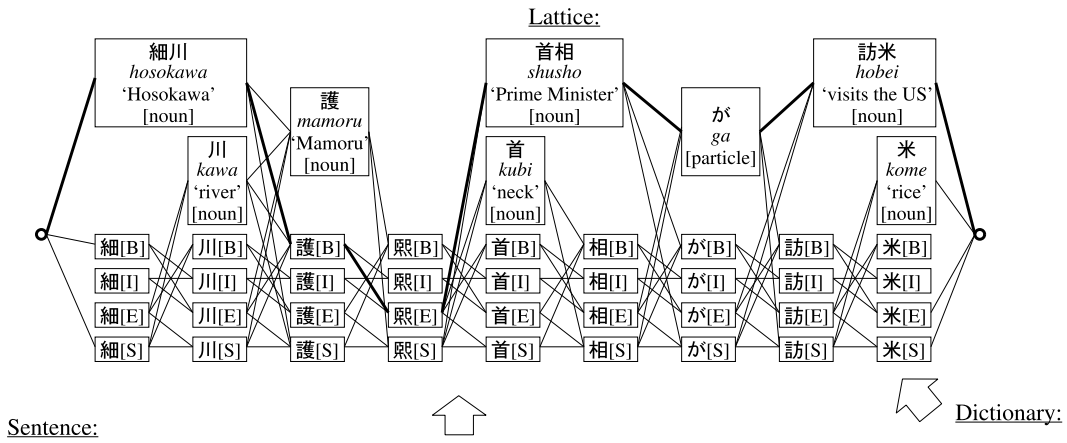

Figure 4.3. Example of the Hybrid Method

words, but cannot use more detailed or wide-scope information than character-level information. Although word level information such as in word dictionaries seems informative for word segmentation[4], the character-based tagging method does not use them.

Then, we propose a hybrid method which combines the Markov model-based method and the character-based tagging method to make the most of word-level and character-level information, in order to achieve high overall accuracy.

4.3.1 The Hybrid Method

This hybrid method is mainly based on word-level Markov models, but also uses POC-tags in order to handle unknown words. Both POS-tags and POC-tags are handled in the same way and word segmentation for known words and unknown words are conducted simultaneously.

Figure 4.3 shows an example of the hybrid method given a Japanese sentence "細川護熙首相が訪米(*hosokawamorihiroshushogahobei* 'Prime Minister Morihiro Hosokawa visits the US')", where the word "護熙(*morihiro* 'Morihiro')"(person's

[4]According to Sproat and Emerson (2003), the maximum matching method, which is one of the simplest word-based method, achieves more than 98% of F-measure in Chinese word segmentation if unknown words are not exist.

name) is an unknown word. First, given a sentence, nodes of the lattice for known words are constructed as in the usual Markov model-based method. Next, for each character in the sentence, nodes with POC-tags (four nodes for each character) are constructed. Then, the most likely path is searched (the thick lines indicate the correct path in the example). Unknown words are identified by the nodes with POC-tags. Note that some transitions of states are not allowed (e.g. from **I** to **B**, or from any POS-tags to **E**), and such transitions are ignored in searching.

The hybrid method handles POS-tags and POC-tags equally as states of Markov models. Thus, nodes for known words and nodes for characters composing unknown words in the lattice can be handled uniformly, and the most likely path is searched in the same way as the usual Markov model-based method.

4.3.2 Probabilistic Model

Since the basic Markov models (POS bigram models) in Equation (4.1) are not expressive enough, POS trigram or lexicalized n-gram models are often used in English POS tagging. We propose to use a mixture model of four models, POS unigram, POS bigram, POS trigram and word bigram[5], to estimate probability of a path in a lattice as follows[6]:

$$
\begin{aligned}
P(W,T) &= \prod_{i=1}^{n} P(w_i t_i | w_0 t_0 \cdots w_{i-1} t_{i-1}), \\
&\simeq \prod_{i=1}^{n} \{ \lambda_1 P^{\text{POS unigram}}(w_i t_i | w_0 t_0 \cdots w_{i-1} t_{i-1}) \\
&\quad + \lambda_2 P^{\text{POS bigram}}(w_i t_i | w_0 t_0 \cdots w_{i-1} t_{i-1}) \\
&\quad + \lambda_3 P^{\text{POS trigram}}(w_i t_i | w_0 t_0 \cdots w_{i-1} t_{i-1}) \\
&\quad + \lambda_4 P^{\text{word bigram}}(w_i t_i | w_0 t_0 \cdots w_{i-1} t_{i-1}) \}, \\
&= \prod_{i=1}^{n} \{ \lambda_1 P(w_i | t_i) P(t_i) \\
&\quad + \lambda_2 P(w_i | t_i) P(t_i | t_{i-1})
\end{aligned}
$$

[5] We call the bigram probability of a word and POS-tag pair as *word bigram* in this chapter.

[6] Note that this is not a mixture model (linear interpolation) of the probabilities of parts-of-speech (t) which is often used in trigram-based POS taggers, but is a mixture model of the probabilities of word and part-of-speech pairs ($\langle w, t \rangle$).

$$
\begin{aligned}
&+\lambda_3 P(w_i|t_i)P(t_i|t_{i-2}t_{i-1}) \\
&+\lambda_4 P(w_i t_i|w_{i-1}t_{i-1})\}, \\
&(\lambda_1+\lambda_2+\lambda_3+\lambda_4=1).
\end{aligned} \tag{4.4}
$$

The probabilities in the equation above are estimated from a word segmented and POS-tagged corpus using the maximum-likelihood method. Unseen events in the training data are handled as they occurred 0.5 times. $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are calculated by the leave-one-out method (Brants, 2000) as shown in Figure 4.4. A word dictionary for a Markov model-based system is often constructed from a training corpus, and no unknown words exist in the training corpus in such a case. Therefore, when the parameters of the above probabilities are calculated from a training corpus, words that appear only once in the training corpus are regarded as unknown words and are decomposed into characters with POC-tags so that statistics about unknown words are obtained. In a case when a word dictionary apart from the training corpus is available, words in the training corpus but not in the dictionary are just regarded as unknown words.

4.3.3 Incorporation of Character-level Features

Character-level features such as character types seem to be useful for segmenting unknown words. Actually, in Japanese, character types give important clues for identifying word boundaries (Nagata, 1999b). In order to incorporate various character-level features into the probabilistic model, we calculate word emission probabilities *for POC-tags* in Equation (4.4) by Bayes' theorem:

$$
P(w_i|t_i) = \frac{P(t_i|w_i)P(w_i)}{P(t_i)}, \tag{4.5}
$$

where w_i is a character and t_i is a POC-tag. In the right hand side of the above equation, $P(t_i)$ and $P(w_i)$ are estimated by the maximum-likelihood method, and the probability of a POC-tag t_i given a character w_i ($P(t_i|w_i)$) is estimated using maximum entropy (ME) models (Berger et al., 1996). We use the following features for ME models, where c_x is the xth character in the sentence, i' is the position of the character corresponding to w_i (i.e. $w_i = c_{i'}$) and y_x is the character type of c_x (Table 4.2 shows a list of character types we use):

```
# W:  Set of words in the training corpus,
# T:  Set of POS-tags in the training corpus,
# f(w_i, t_i, ···):  Frequency of a word and tag sequence w_i, t_i, ··· in the
#                    training corpus,
# N:  Number of words in the training corpus.

# Initialization
for i := 1 to 4
  λ_i := 0
# Calculation of λ_i
foreach t_{-2}, t_{-1}, t_0 ∈ T, w_{-1}, w_0 ∈ W, f(t_{-2}, w_{-1}, t_{-1}, w_0, t_0) ≥ 1
begin
  # In the following calculations, the result of division is defined
  # as 0 if its divisor is 0.
  p_1 := (f(w_0,t_0)-1)/(f(t_0)-1) (f(t_0)-1)/(N-1)
  p_2 := (f(w_0,t_0)-1)/(f(t_0)-1) (f(t_{-1},t_0)-1)/(f(t_{-1})-1)
  p_3 := (f(w_0,t_0)-1)/(f(t_0)-1) (f(t_{-2},t_{-1},t_0)-1)/(f(t_{-2},t_{-1})-1)
  p_4 := (f(w_{-1},t_{-1},w_0,t_0)-1)/(f(w_{-1},t_{-1})-1)
  for i := 1 to 4
    if p_i = max_j p_j
      λ_i := λ_i + f(t_{-2}, w_{-1}, t_{-1}, w_0, t_0)
end
# Normalization
Z := Σ_i λ_i
for i := 1 to 4
  λ_i := λ_i/Z
```

Figure 4.4. Algorithm for Calculating λ_i

Table 4.2. Character Types

Character Type	Description
Alphabet	Alphabets
Numeral	Arabic and Chinese numerals
Symbol	Symbols
Kanji	Chinese characters
Hiragana	Hiragana (Japanese scripts)
Katakana	Katakana (Japanese scripts)

(1) Characters ($c_{i'-2}$, $c_{i'-1}$, $c_{i'}$, $c_{i'+1}$, $c_{i'+2}$)

(2) Pairs of characters ($c_{i'-2}c_{i'-1}$, $c_{i'-1}c_{i'}$, $c_{i'-1}c_{i'+1}$, $c_{i'}c_{i'+1}$, $c_{i'+1}c_{i'+2}$)

(3) Character types ($y_{i'-2}$, $y_{i'-1}$, $y_{i'}$, $y_{i'+1}$, $y_{i'+2}$)

(4) Pairs of character types ($y_{i'-2}y_{i'-1}$, $y_{i'-1}y_{i'}$, $y_{i'-1}y_{i'+1}$, $y_{i'}y_{i'+1}$, $y_{i'+1}y_{i'+2}$)

Parameters of ME models are trained using all the words in a training corpus. We use the Generalized Iterative Scaling algorithm (Darroch and Ratcliff, 1972) for parameter estimation, and features that appeared less than or equal to 10 times in training data are ignored in order to avoid overfitting.

4.4. Application to Korean Morphological Analysis

The hybrid method described so far can conduct Chinese and Japanese word segmentation. However, the method cannot conduct Korean morphological analysis directly. This section presents a method to apply the hybrid method to Korean morphological analysis.

4.4.1 Korean Morphological Analysis

Korean is largely different from Chinese or Japanese as well as English. In Korean, each word (also called *word phrase* or *eojeol*) is separated by spaces like a word in

English. However, Korean is an agglutinative language and each word consists of one or more morphemes. Each morpheme is not separated explicitly like a word in Chinese and Japanese. Therefore, in Korean morphological analysis, we must segment each word into morphemes, obtain their base forms, and predict their POS tags. Figure 4.5 shows an example of POS tagged Korean corpus. Words are shown on the left and corresponding morphemes are shown on the right. The morphemes are separated by '+', and '/' is followed by a POS tag. The goal of Korean morphological analysis is to obtain the analyzed morphemes (the right part in the figure) given the sentence (the left part in the figure).

Korean has complex morphology as shown below:

아는	알/VV+는/EAN
졌습니다	지/VX+었/EPF+습니다/EFN
뭔지는	무엇/NPN+이/CO+ㄴ지/EFN+는/PAU

When inflected words are created, characters are often deleted or contracted, and we must recover base forms of morphemes from given inflected words.

4.4.2 Previous Work

Several methods for Korean morphological analysis have been proposed (Lee et al., 2002). Han and Palmer (2004) studied a completely corpus-based method for Korean morphological analysis. The method consists of the following four steps:

1. *Tokenization*

An inputted sentence is segmented into tokens by white spaces and punctuation symbols.

2. *Spelling Recovery (with reduced POS tagging)*

In this step, each word in the input sentence is firstly POS tagged with trigram models, using a subset of POS tags. Then, spelling of each word with morphological deletion or contraction is recovered based on the POS tags and suffixes of the word using a template dictionary. The template dictionary is obtained from training data.

3. *Morphological Tagging*

A complex tag is attached to each word with trigram models. A complex tag is the concatenation of the POS tags of the morphemes in a word.

4. *Lemma/Inflection Identification*

Each morpheme and its POS tag are identified based on the complex tag using a morpheme dictionary.

The method is corpus-based, so that no hand-crafted dictionaries are necessary and any corpora can be used. However, the method processes input sentences deterministically in each step, and optimal solutions may not be obtained.

4.4.3 Two Step Method for Korean Morphological Analysis

We conduct Korean morphological analysis with a two step method, which is similar to the method explained in the previous subsection. In the method, given a word sequence $W = w_1, \cdots, w_l$, the morpheme sequence $M = m_1, \cdots, m_n$ and their POS tags $T = t_1, \cdots, t_n$ are identified as follows:

$$\begin{aligned} \langle \hat{M}, \hat{T} \rangle &= \operatorname*{argmax}_{M,T} P(M, T|W), \\ &\simeq \operatorname*{argmax}_{M,T} P(M, T|W')P(W'|W), \end{aligned} \tag{4.6}$$

where $W' = w'_1, \cdots, w'_l$ is a sequence of words whose spells are recovered, and we assume that the concatenation of M and that of W' are equal ($m_1 \cdots m_n = w'_1 \cdots w'_l$). This method firstly calculates $P(W'|W)$ for all possible W' given a sentence W, then calculates $P(M, T|W')$ for all possible M and T, and choose the best one. In the above equation, $P(M, T|W')$ can be calculated in the same way as Chinese and Japanese word segmentation, and we will explain how to calculate $P(W'|W)$ below.

We use spelling recovery rules, which recover the base forms of morphologically contracted words. Spelling recovery rules are collected from a training corpus. Given a pair of a contracted word and its base form, a spelling recovery rule is made by eliminating the longest common prefix. For example, if a contracted word is "abcde" and its base form is "abcdfg", the spelling recovery rule is "e→fg". Note that we do not eliminate the longest common prefix if the contracted word is a prefix of the base form. In such a case, provided the length of the contracted word is n, we eliminate the first $n - 1$ characters from the contracted word and

the base form, to make a spelling recovery rule. For example, a contracted word "abcd" and its base form "abcdfg" produce a spelling recovery rule "d→dfg". We use a special spelling recovery rule, $\epsilon \rightarrow \epsilon$, where ϵ means an empty string. This rule is used to represent that no spelling recovery is conducted.

The value of $P(W'|W)$ in Equation 4.6 can be decomposed as follows:

$$P(W'|W) \simeq \prod_{i=1}^{l} P(w'_i|w_i). \tag{4.7}$$

We assume that a word w is recovered to w' using a spelling recovery rule $r \rightarrow r'$, and calculate the value of $P(w'|w)$ as follows:

$$P(w'|w) \equiv P(r \rightarrow r'|w), \tag{4.8}$$

$$P(r \rightarrow r'|w) \equiv \left\{1 - \sum_{\substack{s,s' \\ r \prec s \preceq w}} P(s \rightarrow s'|w)\right\} P(r \rightarrow r'|r), \tag{4.9}$$

where $P(r \rightarrow r'|x)$ is the probability that the spelling recovery rule $r \rightarrow r'$ is applied to the string x, $x \preceq y$ means that the string y ends with the string x (i.e., x is a suffix of y), and $x \prec y$ means $x \preceq y$ and $x \neq y$. We calculate $P(r \rightarrow r'|r)$ as follows:

$$P(r \rightarrow r'|r) = \begin{cases} 1 & (r = \epsilon \text{ and } r' = \epsilon), \\ \frac{f(r \rightarrow r')}{\sum_{\substack{s,s' \\ \epsilon \preceq s \preceq r}} f(s \rightarrow s')} & (\text{otherwise}), \end{cases} \tag{4.10}$$

where $f(r \rightarrow r')$ is the frequency that the spelling recovery rule $r \rightarrow r'$ appears in a training corpus.

We explain the method using an example shown in Figure 4.6. Suppose that we have spelling recovery rules and a dictionary as shown in the figure, and the input sentence is "pqr abcde xyz". Spelling recovery rules can have constraints of morpheme boundaries and POS tags as shown in the example (represented by '+' and '/'). Here we only consider analysis of the word "abcde" in the input sentence. The input sentence is a sequence of words (Graph (a)). By applying the spelling recovery rules, we can obtain spelling recovered words and their probabilities (Graph (b)). We apply the hybrid method and all the hypotheses are generated (Graph (c)) (Nodes with POC tags are represented as nodes with a tag U for simplicity). The best path in the lattice is searched in the same way as word segmentation of Chinese and Japanese, except that the probabilities of spelling recovery are also considered.

2	2/NNU
대대에는	대대/NNC+에/PAD+는/PAU
예하	예하/NNC
구분대와	구분대/NNC+와/PAD
교신하는	교신/NNC+하/XSV+는/EAN
무전망이	무전망/NNC+이/PCA
몇	몇/NNU
개나	개/NNX+나/PAU
있지	있/VJ+지/EFN
?	?/SFN

Figure 4.5. Example of POS Tagged Korean Corpus

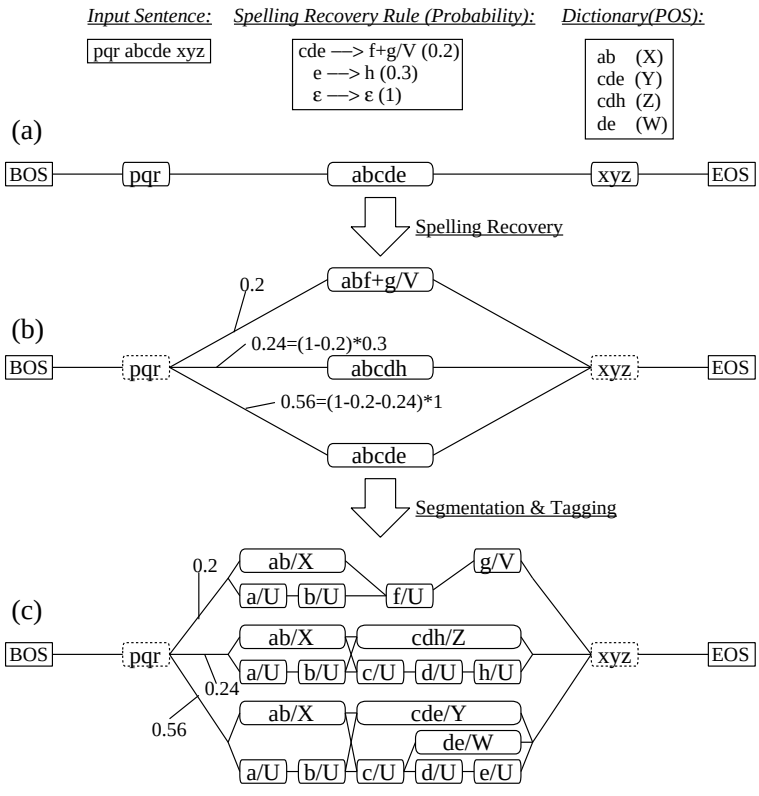

Figure 4.6. Example of Korean Morphological Analysis

Table 4.3. Statistical Information of Corpora

Corpus	# of Training Words	# of Testing Words (known/unknown)	# of Words in Dictionary	# of POS Tags
AS	5,806,611	11,985 (11,727/ 258)	146,212	(64)
CTB	250,841	39,922 (32,706/7,216)	19,730	(64)
HK	239,852	34,955 (32,463/2,492)	23,747	(64)
PK	1,121,017	17,194 (16,005/1,189)	55,226	(64)
EDR	2,452,891	2,652,156 (2,600,051/52,105)	82,410	15
KY	198,514	31,302 (29,926/1,376)	1,870,461	42
RWCP	840,879	93,155 (93,085/ 70)	315,602	69
MAKT	69,323	7,177 (6,623/ 554)	6,989	32
KTB	86,034	7,118 (6,733/ 385)	3,397	33

4.5. Experiments

This section gives experimental results of Chinese and Japanese word segmentation and Korean morphological analysis with the hybrid method. The following measures are used to evaluate performance of word segmentation:

R : Recall.

$$R = \frac{\langle \text{\# of correctly segmented words in system's output} \rangle}{\langle \text{\# of words in test data} \rangle}.$$

P : Precision.

$$P = \frac{\langle \text{\# of correctly segmented words in system's output} \rangle}{\langle \text{\# of words in system's output} \rangle}.$$

F : F-measure.

$$F = \frac{2RP}{R+P}.$$

R_{known} : Recall for known words.

$$R_{known} = \frac{\langle \text{\# of correctly segmented known words in system's output} \rangle}{\langle \text{\# of known words in test data} \rangle}.$$

$R_{unknown}$: Recall for unknown words.

$$R_{unknown} = \frac{\langle \text{\# of correctly segmented unknown words in system's output} \rangle}{\langle \text{\# of unknown words in test data} \rangle}.$$

P_{known} : Precision for known words.

$$P_{known} = \frac{\langle \text{\# of correctly segmented known words in system's output} \rangle}{\langle \text{\# of known words in system's output} \rangle}.$$

$P_{unknown}$: Precision for unknown words.

$$P_{unknown} = \frac{\langle \text{\# of correctly segmented unknown words in system's output} \rangle}{\langle \text{\# of unknown words in system's output} \rangle}.$$

4.5.1 Experiments on Chinese Word Segmentation

We use four Chinese word-segmented corpora; the Academia Sinica corpus (AS), the Penn Chinese Treebank corpus (CTB), the Hong Kong City University corpus (HK) and the Beijing University corpus (PK), all of which were used in the First International Chinese Word Segmentation Bakeoff (Sproat and Emerson, 2003) at ACL-SIGHAN 2003[7].

These four corpora are word-segmented corpora, but POS-tags are not attached, therefore we need to attach a POS-tag (a class) to each word which is necessary for the Markov model-based method. We attached a class to each word using the Baum-Welch algorithm (Manning and Schütze, 1999) which is used for hidden Markov models. The algorithm finds a locally optimal tag sequence which maximizes Equation (4.1) in an unsupervised way. The initial states are randomly assigned, and the number of classes is set to 64[8].

We use the following systems for comparison:

***Bakeoff-1, 2, 3*:** The top three systems participated in the SIGHAN Bakeoff (Sproat and Emerson, 2003).

[7]The AS, HK and PK corpora are available from the SIGHAN Bakeoff's Web page (`http://www.sighan.org/bakeoff2003/alldata.html`).
The CTB corpus is available from LDC (Catalog No. LDC2003E16).

[8]We set the number of classes to 64 because the number of POS-tags used in widely used corpora is about 40–70 (e.g. the Kyoto University corpus has 42 tags and the RWCP corpus has 69 tags).

***Maximum Matching*:** A word segmentation system using the (forward) maximum matching method.

***Character Tagging*:** A word segmentation system using the character-based tagging method. This system is almost the same as the one studied by Xue (2003). The following features are used to estimate the POC-tag of a character c_i, where c_x is the xth character in the sentence, and y_x and t_x is the character type and the POC-tag of c_x respectively:

(1) Characters (c_{i-2}, c_{i-1}, c_i, c_{i+1}, c_{i+2})

(2) Pairs of characters ($c_{i-2}c_{i-1}$, $c_{i-1}c_i$, $c_{i-1}c_{i+1}$, c_ic_{i+1}, $c_{i+1}c_{i+2}$)

(3) Character types (y_{i-2}, y_{i-1}, y_i, y_{i+1}, y_{i+2})

(4) Pairs of character types ($y_{i-2}y_{i-1}$, $y_{i-1}y_i$, $y_{i-1}y_{i+1}$, y_iy_{i+1}, $y_{i+1}y_{i+2}$)

(5) Unigram and bigram of previous POC-tags ($t_{i-1}, t_{i-2}t_{i-1}$)

The most likely POC-tag sequence is solved with the Viterbi search.

All these systems including ours do not use any other resources than the training data. In this experiment, word dictionaries used by the hybrid method and Maximum Matching are constructed from all the words in each of the training corpora[9]. Statistical information of these data is shown in Table 4.3. The calculated values of λ_i in Equation (4.4) are shown in Table 4.4.

The results are shown in Table 4.5. Our system achieved the best F-measure values for the AS, HK and PK corpora. Both the recall and the precision for the AS corpus, and only the precisions for the HK and PK corpora of our system have statistically significant difference at 95% confidence level[10]. Although recall values of the hybrid system for known words are not high compared to the participants of SIGHAN Bakeoff, the recall values for known words and unknown words

[9]We converted the character code of Arabic numerals in the PK training corpus (GBK code) to ASCII code, because the character code is different from that of the test data.

[10]The 95% confidence interval of the recall value R is given as $R \pm 2\sqrt{R(1-R)/n}$, where n is the number of words in the test data. If the values of two systems do not overlap, the difference of two system's recalls is regarded as statistically significant (Sproat and Emerson, 2003). The same goes for precisions (P).

Table 4.4. Calculated Values of λ_i

Corpus	λ_1	λ_2	λ_3	λ_4
AS	0.037	0.178	0.257	0.528
CTB	0.044	0.219	0.251	0.486
HK	0.048	0.251	0.313	0.388
PK	0.055	0.207	0.242	0.495
EDR	0.047	0.085	0.116	0.752
KY	0.080	0.126	0.237	0.556
RWCP	0.073	0.105	0.252	0.571
MAKT	0.043	0.158	0.251	0.548
KTB	0.018	0.109	0.173	0.700

are relatively well-balanced. The results of Maximum Matching and Character Tagging show the trade-off between the word-based approach and the character-based approach which was discussed in Section 4.3. Maximum Matching is word-based and has the higher recall values for known words than Character Tagging on the CTB, HK and PK corpora. Character Tagging is character-based and has the highest recall values for unknown words on the AS, HK and PK corpora (Bakeoff-2 for AS and Bakeoff-3 for HK are also based on the character-based tagging method).

4.5.2 Experiments on Japanese Word Segmentation

We use three Japanese word-segmented and POS tagged corpora; the EDR corpus version 1.0 (EDR), the Kyoto University corpus version 2.0 (KY) and the RWCP corpus (RWCP).

In the experiments with the EDR corpus, we use the first 100,000 sentences of the corpus as the training data, and the remaining 107,802 sentences as the test data. A word dictionary is constructed from all the words in the training data.

In the experiments with the KY corpus (a newspaper corpus), we use the articles of January 1 and ones from January 3 to January 8 as the training data, and the articles of January 9 as the test data. As a word dictionary, we use the dictionary of JUMAN version 3.61 (Kurohashi and Nagao, 1998).

Table 4.5. Performance of Chinese Word Segmentation

Corpus	System	R	P	F	R_{known}	$R_{unknown}$	P_{known}	$P_{unknown}$
AS	Hybrid method	**0.973***	**0.970***	**0.972**	0.979	0.717	**0.974**	**0.804**
	Bakeoff-1	0.966	0.956	0.961	**0.980**	0.364	0.961	0.584
	Bakeoff-2	0.961	0.958	0.959	0.966	0.729	0.967	0.614
	Bakeoff-3	0.944	0.945	0.945	0.952	0.574	0.957	0.490
	Maximum Matching	0.917	0.912	0.915	0.938	0.000	0.914	0.000
	Character Tagging	0.961	0.959	0.960	0.966	**0.744**	0.968	0.610
CTB	Hybrid method	0.877	0.872	0.874	0.927	0.647	**0.919**	0.655
	Bakeoff-1	0.886	**0.875**	**0.881**	0.927	**0.705**	0.913	**0.701**
	Bakeoff-2	**0.892**	0.856	0.874	0.947	0.644	N/A	N/A
	Bakeoff-3	0.867	0.797	0.831	**0.963**	0.431	0.834	0.551
	Maximum Matching	0.800	0.663	0.725	**0.963**	0.063	0.736	0.084
	Character Tagging	0.832	0.836	0.834	0.872	0.650	0.913	0.552
HK	Hybrid method	**0.951**	**0.948***	**0.950**	0.969	0.715	**0.965**	**0.718**
	Bakeoff-1	0.947	0.934	0.940	0.972	0.625	N/A	N/A
	Bakeoff-2	0.940	0.908	0.924	**0.980**	0.415	0.922	0.627
	Bakeoff-3	0.917	0.915	0.916	0.936	0.670	0.953	0.537
	Maximum Matching	0.908	0.830	0.867	0.974	0.037	0.867	0.052
	Character Tagging	0.917	0.917	0.917	0.932	**0.728**	0.957	0.545
PK	Hybrid method	0.957	**0.951***	**0.954**	0.970	0.774	**0.961**	0.811
	Bakeoff-1	**0.962**	0.940	0.951	**0.979**	0.724	0.943	**0.904**
	Bakeoff-2	0.955	0.938	0.947	0.976	0.680	0.942	0.867
	Bakeoff-3	0.955	0.938	0.946	0.977	0.647	0.946	0.815
	Maximum Matching	0.930	0.883	0.906	0.973	0.347	0.896	0.579
	Character Tagging	0.932	0.931	0.931	0.943	**0.786**	0.958	0.639

(* indicates significance at $p < 0.05$)

Table 4.6. Performance of Japanese Word Segmentation

Corpus	System	R	P	F	R_{known}	$R_{unknown}$	P_{known}	$P_{unknown}$
EDR	Hybrid method	**0.952***	**0.948***	**0.950**	**0.962**	0.446	0.951	**0.693**
	Maximum Matching	0.757	0.824	0.789	0.772	0.009	0.828	0.039
	Character Tagging	0.946	0.946	0.946	0.953	**0.591**	**0.956**	0.501
KY	Hybrid method	0.986	0.984	0.985	0.989	0.924	0.984	0.968
	JUMAN	**0.989**	**0.985**	**0.987**	**0.993**	0.889	**0.985**	**0.985**
	Maximum Matching	0.806	0.753	0.779	0.843	0.004	0.829	0.002
	Character Tagging	0.946	0.942	0.944	0.947	**0.940**	0.959	0.685
RWCP	Hybrid method	**0.993***	**0.994***	**0.993**	**0.993**	0.586	**0.994**	**0.820**
	ChaSen	0.991	0.992	0.991	0.991	0.243	0.992	0.515
	Maximum Matching	0.879	0.918	0.898	0.880	0.100	0.919	0.103
	Character Tagging	0.972	0.968	0.970	0.972	**0.629**	0.977	0.051

(* indicates significance at $p < 0.05$)

In the experiments with the RWCP corpus, we separated the corpus into training data and test data randomly. As a word dictionary, we use IPADIC version 2.4.4 (Matsumoto and Asahara, 2001) which is distributed with ChaSen (Matsumoto et al., 2001).

Statistical information of these data is shown in Table 4.3.

We use the following systems for comparison:

***ChaSen (version 2.2.8)*:** The word segmentation and POS tagging system (Matsumoto et al., 2001) based on extended Markov models (Asahara and Matsumoto, 2000). This system carries out unknown word processing using hand-crafted rules based on character types. This system is designed for the POS tag set of the RWCP corpus.

***JUMAN (version 3.61)*:** The word segmentation and POS tagging system (Kurohashi and Nagao, 1998) based on a rule-based method. This system carries out unknown word processing using hand-crafted rules based on character types. This system is designed for the POS tag set of the KY corpus.

***Maximum Matching*:** The same system used in the Chinese experiments.

Table 4.7. Performance of Korean Morphological Analysis

Corpus	System	$Acc.^{S.R.}$	$Acc.^{seg.}$	R	P	F	R_{known}	$R_{unknown}$
MAKT	Hybrid method	0.979	0.940	0.951	0.950	0.950	0.970	0.715
KTB	Hybrid method	0.984	0.964	0.971	0.962	0.967	0.984	0.743

***Character Tagging*:** The same system used in the Chinese experiments.

In the above systems, although JUMAN uses hand-crafted parameters, the other systems do not use any other resources than the training data and the dictionaries[11].

The calculated values of λ_i in Equation (4.4) are shown in Table 4.4.

The results are shown in Table 4.6[12]. Compared to ChaSen and JUMAN, the hybrid method has the comparable F-measure values and the higher recall values for unknown words. Character Tagging has the highest recall value for unknown words as in the Chinese experiments. Both the recall (R) and the precision (P) for the EDR and RWCP corpora of our system have statistically significant difference at 95% confidence level compared to the other systems.

In the word segmentation errors of the hybrid method for the KY corpus, about 25% of the mis-segmented words are unknown words, and unknown words are still major problems. About 20% of the remaining mis-segmented words are words that exist in the dictionary but not in the training corpus, and some errors are caused by inconsistencies of the corpus as mentioned by Uchimoto et al. (2001).

[11]In the experiments with the KY corpus and the RWCP corpus, Maximum Matching uses only the dictionaries because it cannot use information in training corpora directly, and Character Tagging uses only the training corpora because it cannot use information in dictionaries directly. Thus these system have handicaps in the experiments. In the experiments with the EDR corpus and the previous experiments with the Chinese corpora, these differences do not exist because dictionaries are constructed from the training corpora.

[12]In this evaluation, R_{known}, $R_{unknown}$, P_{known} and $P_{unknown}$ are calculated considering words in the dictionaries as known words. Words that are in the training corpora but not in the dictionaries are regarded as unknown words. The number of known/unknown words of the KY and RWCP corpus shown in Table 4.3 is also calculated in this way.

4.5.3 Experiments on Korean Morphological Analysis

We use two Korean morphologically annotated corpora; Morphologically Annotated Korean Text (MAKT) and Korean English Treebank Annotations (KTB).

In the experiments with the MAKT corpus, we use the first 1,417 sentences of the corpus as the training data, and the remaining 157 sentences as the test data. A dictionary is constructed from all the morphemes in the training data.

In the experiments with the KTB corpus, we use sections 05, 20 and 30 for the test data, and the remaining 30 sections as the training data. A dictionary is constructed from all the morphemes in the training data.

Statistical information of these data is shown in Table 4.3[13]. The calculated values of λ_i in Equation (4.4) are shown in Table 4.4.

The results are shown in Table 4.7. In the table, $Acc.^{S.R.}$ means accuracy of spelling recovery and $Acc.^{seg.}$ means accuracy of morpheme segmentation, calculated as follows:

$$Acc.^{S.R.} = \frac{\langle \text{\# of word phrases whose spellings are correctly recovered} \rangle}{\langle \text{\# of word phrases in test data} \rangle},$$

$$Acc.^{seg.} = \frac{\langle \text{\# of word phrases which are correctly segmented to morphemes} \rangle}{\langle \text{\# of word phrases in test data} \rangle}.$$

The values shown in the table are accuracies of morpheme segmentation, excluding POS tagging. The precisions tend to be lower than recalls. Han and Palmer (2004) achieved F-measure of 0.952 for both morpheme segmentation and POS tagging. However, it is difficult to compare the results with ours because experimental settings are different.

4.5.4 Experiments in Various Settings

Some techniques are adopted in the hybrid method to achieve high accuracy, i.e., mixture models with POS trigram and word bigram, ME models with character-level features, and unsupervised learning for word segmented corpora with no POS-tags. We conduct some experiments to investigate their effect.

[13]The numbers of words in this table indicate the numbers of morphemes for Korean corpora.

Effect of Mixture Models

The hybrid method uses combination of POS unigram, POS bigram, POS trigram and word bigram as shown in Equation (4.4) to calculate the probabilities of word and tag sequences (Section 4.3.2). To see the effect of each n-gram model, we conduct experiments using different probabilistic models. Table 4.8 shows the results. In the table, **POS bigram**, **POS trigram** and **word bigram** correspond to the cases where Equations (4.11), (4.12) and (4.4) are used in the calculation of the probability $P(W,T)$ in Equation (4.3).

$$
\begin{aligned}
P(W,T) &= \prod_{i=1}^{n} P(w_i t_i | w_0 t_0 \cdots w_{i-1} t_{i-1}), \\
&\simeq \prod_{i=1}^{n} \{\lambda'_1 P^{\text{POS unigram}}(w_i t_i | w_0 t_0 \cdots w_{i-1} t_{i-1}) + \lambda'_2 P^{\text{POS bigram}}(w_i t_i | w_0 t_0 \cdots w_{i-1} t_{i-1})\}, \\
&= \prod_{i=1}^{n} \{\lambda'_1 P(w_i|t_i) P(t_i) + \lambda'_2 P(w_i|t_i) P(t_i|t_{i-1})\}, \qquad (4.11) \\
&\quad (\lambda'_1 + \lambda'_2 = 1), \\
P(W,T) &= \prod_{i=1}^{n} P(w_i t_i | w_0 t_0 \cdots w_{i-1} t_{i-1}), \\
&\simeq \prod_{i=1}^{n} \{\lambda''_1 P^{\text{POS unigram}}(w_i t_i | w_0 t_0 \cdots w_{i-1} t_{i-1}) + \lambda''_2 P^{\text{POS bigram}}(w_i t_i | w_0 t_0 \cdots w_{i-1} t_{i-1}) \\
&\qquad + \lambda''_3 P^{\text{POS trigram}}(w_i t_i | w_0 t_0 \cdots w_{i-1} t_{i-1})\}, \\
&= \prod_{i=1}^{n} \{\lambda''_1 P(w_i|t_i) P(t_i) + \lambda''_2 P(w_i|t_i) P(t_i|t_{i-1}) + \lambda''_3 P(w_i|t_i) P(t_i|t_{i-2} t_{i-1})\}, \qquad (4.12) \\
&\quad (\lambda''_1 + \lambda''_2 + \lambda''_3 = 1).
\end{aligned}
$$

The results show that the word bigram, which utilizes lexicalized information, contributed to accuracy more than POS trigram. POS trigram has little effect for the Chinese corpora (AS, CTB, HK and PK). One reason of this may be the use of unsupervisedly tagged classes, or POS trigram may not be useful in Chinese word segmentation as previously reported by Asahara and Matsumoto (2002).

Table 4.8. Performance for Different Probabilistic Models

Corpus	$F(R_{known}/R_{unknown})$		
	POS bigram	POS trigram	word bigram
AS	0.967 (0.976/0.616)	0.967 (0.976/0.624)	0.972 (0.979/0.717)
CTB	0.868 (0.924/0.636)	0.868 (0.923/0.644)	0.874 (0.927/0.647)
HK	0.940 (0.966/0.660)	0.939 (0.964/0.675)	0.950 (0.969/0.715)
PK	0.948 (0.969/0.740)	0.948 (0.968/0.751)	0.954 (0.970/0.774)
EDR	0.922 (0.923/0.319)	0.923 (0.925/0.320)	0.950 (0.962/0.446)
KY	0.980 (0.983/0.898)	0.981 (0.984/0.907)	0.985 (0.989/0.924)
RWCP	0.989 (0.987/0.543)	0.991 (0.989/0.557)	0.993 (0.993/0.586)

Table 4.9. Effect of Character-level Features

Corpus	$F(R_{known}/R_{unknown})$	
	without character-level features	with character-level features
AS	0.969 (0.977/0.671)	0.972 (0.979/0.717)
CTB	0.854 (0.904/0.593)	0.874 (0.927/0.647)
HK	0.941 (0.958/0.671)	0.950 (0.969/0.715)
PK	0.952 (0.967/0.750)	0.954 (0.970/0.774)
EDR	0.948 (0.961/0.382)	0.950 (0.962/0.446)
KY	0.982 (0.988/0.885)	0.985 (0.989/0.924)
RWCP	0.993 (0.993/0.557)	0.993 (0.993/0.586)

Effect of Character-level Features

The hybrid method uses ME models to calculate word emission probabilities for POC-tags in order to utilize various character-level features (Section 4.3.3). To see the effect of the use of the character-level features, we conduct experiments with two settings: In the calculation of the word emission probabilities for POC-tags ($P(w_i|t_i)$ in Equation (4.5)), one uses simply the maximum likelihood method without character-level features, and the other uses ME models with character-level features as described in the previous section. The results are shown in Table 4.9. Recalls for unknown words are improved by using ME models with extra features, and F-measures are also improved for all the corpora except the RWCP corpus.

Effect of Unsupervised Learning

In the Chinese experiments in Section 4.5.1, we used unsupervisedly tagged classes instead of POS-tags as the states of Markov models because the Chinese corpora are not POS tagged. In order to investigate the effect of such unsupervised learning, we conduct experiments with two types of data; they have the same sentences but one has manually tagged POS-tags and the other has automatically attached POS-tags (classes).

We use three corpora, the EDR, KY and RWCP corpora, which have manually tagged POS-tags. We attached a class for each word in the corpora using the Baum-Welch algorithm. In the Baum-Welch training, the initial states are randomly assigned and the number of classes is set to 64. The sizes of the training and test data we use in this experiments are the same as in the experiments in Section 4.5.2, but dictionaries are constructed from the training data because the POS-tags in the existing dictionaries used in Section 4.5.2 are not compatible with the classes attached by unsupervised learning. The results are shown in Table 4.10.

The results show that there is little difference between the use of manually tagged POS-tags and automatically tagged ones in F-measure, and we can obtain enough word segmentation accuracy with the classes attached by the Baum-Welch algorithm instead of the manually tagged POS-tags. The recalls for unknown words with automatically tagged POS-tags are higher than those with manually tagged POS-tags, and the reason may be explained as follows: In the case with automatically tagged POS-tags, the average number of POS-tags per word is much larger than that in the manually tagged case (for example, in the experiment with KY corpus, the average number of possible POS-tags per word is 1.07 in the manually tagged case and 2.08 in the automatically tagged case). As a result, the number of word and POS-tag pairs appeared only once (which are handled as unknown words in the training phase) in the automatically tagged case is larger than that in the case with manually tagged POS tags. Then, the number of training examples for unknown words is relatively larger, and the recall values for unknown words are high.

Table 4.10. Effect of Unsupervised Learning

Corpus	F ($R_{known}/R_{unknown}$)	
	with POS-tags annotated by human	with classes attached by unsupervised learning
EDR	0.950 (0.962/0.446)	0.950 (0.959/0.520)
KY	0.959 (0.989/0.683)	0.960 (0.981/0.750)
RWCP	0.985 (0.994/0.668)	0.984 (0.991/0.719)

4.6. Related Work and Discussion

4.6.1 Previous Methods for Word Segmentation

Many studies have been conducted on word segmentation and unknown word processing. Xue (2003) studied Chinese word segmentation using the character-based tagging method. As seen in the previous section, this method handles known and unknown words in the same way basing on character-level information. Our experiments showed that the method has quite high accuracy for unknown words, but accuracy for known words tends to be lower than other methods.

Nagata (1999b) conducted Japanese word segmentation by using probabilistic models incorporating character type information. In his experiments with the EDR corpus, he reported that $R/P/F$ were 0.946/0.937/0.941 and $R_{unknown}/P_{unknown}$ were 0.420/0.664 respectively. Our method performed better than the results, but comparing his results with ours is difficult because the experimental settings are not the same.

Asahara et al. (2003) studied Chinese word segmentation based on a character-based tagging method with support vector machines. They preprocessed a given sentence using a word segmenter based on word-level Markov models, and the output is used as features for character-based tagging. Their method is a character-based method incorporating word-level information. They reported $R_{known}/R_{unknown}$ for AS, CTB, HK and PK are respectively 0.952/0.574, 0.949/0.412, 0.980/0.415, 0.975/0.357, relatively higher recalls for known words and lower recalls for unknown words. Ng and Low (2004) studied Chinese word segmentation based on a character-based tagging method. They conducted experiments on the SIGHAN

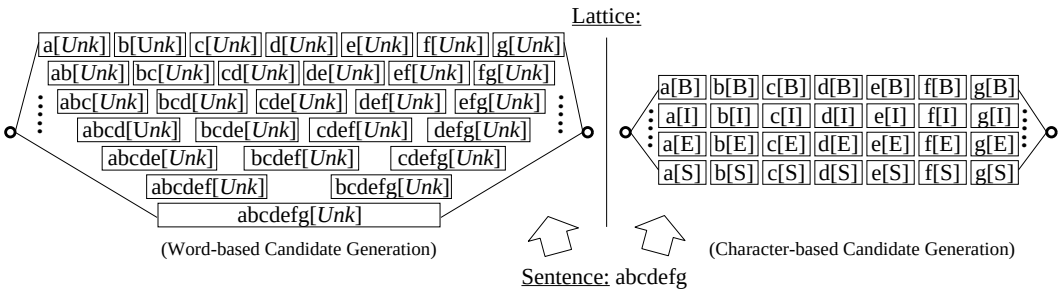

Figure 4.7. Generated Unknown Word Candidates in a Word-based Approach and a Character-based Approach

Bakeoff data and achieved the highest F-measures on the AS, HK and PK corpora compared to the participants of the SIGHAN Bakeoff. However, the values were not presented and cannot be compared to our results. They reported that 0.4% of F-measure increased by adding features which include word-level information. These methods studied by Asahara et al. (2003) and Ng and Low (2004) are character-based methods incorporating word-level information, and have high accuracy. They use both word-level and character-level information, but their approaches are different from ours.

Sarawagi and Cohen (2005) recently proposed semi-Markov conditional random fields (semi-CRFs), which are extension of conditional random fields to semi Markov models. Semi-CRFs will be applied to word segmentation with the character-based tagging approach. The resulting system will be able to use word-level and character-level features, since semi-CRFs can use features for chunks of elements as well as features for individual elements. However, the system will have an efficiency problem described in the following subsection.

4.6.2 Word-based Processing and Character-based Processing for Unknown Words

Uchimoto et al. (2001) studied Japanese word segmentation using ME models. Although their method is word-based, no word dictionaries are used directly in generation of hypotheses and both known and unknown words are handled in the same way. The method estimates how likely a string is to be a word by

using ME models. Given a sentence, the method estimates the probabilities for every substring in the sentence. Word segmentation is conducted by finding a segmentation of the sentence which maximizes the product of the probability that each segmented substring is a word. Compared to our method, their method can handle some types of features for unknown words such as "the word starts with an alphabet and ends with a numeral" or "the word consists of four characters". Our method cannot handle such word-level features easily because unknown words are handled by using characters as a unit. On the other hand, their method seems to have a computational cost problem. In their method, unknown words are processed using words as a unit, and the number of candidates for unknown words in a sentence consists of n characters is equal to $n(n+1)/2$ (Figure 4.7, left). Actually, they did not consider every substring in a sentence, and limited the length of substrings to be up to five characters. In our method and the character-based tagging method, the number of POC-tagged character candidates which are necessary for unknown word processing is equal to $4n$ (Figure 4.7, right), and there is no limitation for the length of unknown words.

4.6.3 Unknown Word Handling in the Hybrid Method

The hybrid method mixes word-level candidates with POS-tags and character-level candidates with POC-tags (Figure 4.3), and calculates probabilities of word and tag sequences on them. How unknown words are modeled in the method can be interpreted as follows: Unknown words can be handled as words with a special POS-tag *Unk*, and unknown words in a sentence can be identified by examining all the substrings in the sentence whether they appear as words with the *Unk* tag. In the hybrid method, given a string of length k, $w_i = c_j \cdots c_{j+k-1}$, its probability to have the *Unk* tag is calculated as:

$$P(w_i\, Unk|h) = \begin{cases} P(c_j\mathbf{S}|h) & (k=1), \\ P(c_j\mathbf{B}|h)\prod_{l=j+1}^{j+k-2} P(c_l\mathbf{I}|h) P(c_{j+k-1}\mathbf{E}|h) & (k>1), \end{cases} \quad (4.13)$$

where h is the history of the sequence. In other words, the probability of an unknown word is implicitly modeled by the product of the probabilities of the composing characters. Although this approach is similar to Nagata's method (Nagata, 1994) which calculates the probability of an unknown word using character-

trigram models, his method is word-based and therefore has the same issues mentioned in Section 4.6.2[14].

Equation (4.13) is calculated within the framework of the word-level Markov models. For example, suppose that the input sentence is "$\alpha/Noun$ β/(unknown word) $\gamma/Noun$", where α, β and γ are words, and β is an unknown word consists of three characters a, b, and c (β = abc). The hybrid method calculates the probability of the sentence as

$$\begin{aligned} & P(\alpha, Noun, \beta, Unk, \gamma, Noun) \\ \simeq\ & P(\alpha, Noun|h)P(\mathrm{a}, \mathbf{B}|h)P(\mathrm{b}, \mathbf{I}|h)P(\mathrm{c}, \mathbf{E}|h)P(\gamma, Noun|h). \end{aligned} \tag{4.14}$$

This probability is calculated using Equation (4.4).

4.7. Conclusion

In this chapter, we presented a hybrid method for word segmentation, which utilizes both word-level and character-level information to obtain high accuracy for known and unknown words. The method combines two existing methods, the Markov model-based method and the character-based tagging method. By handling POS-tags and POC-tags equally, hypotheses for known words and hypotheses for characters which compose unknown words are processed uniformly, and known words and unknown words are identified simultaneously. Experimental results showed that the method achieves high accuracy compared to the other state-of-the-art methods in both Chinese and Japanese word segmentation.

Our hybrid method is based on a standard Markov model-based method and POS-tags for known words are identified at the same time as the word boundaries are identified, but POS-tags for unknown words are not determined. Some approaches exist for predicting POS-tags of unknown words. One approach is to use subdivided POC-tags in order to identify not only the positions of characters but also parts-of-speech of the composing words (Ng and Low, 2004), and the other approach is to use statistical classifiers to predict POS-tags of unknown

[14]In his method, the number of unknown word candidates beginning at each character position in a given sentence is limited up to 10 (Nagata, 1994).

words (Chapter 2). These methods will be used in combination with the hybrid method if identification of POS-tags of unknown words is necessary.

Chapter 5

Revision Learning and its Application to Part-of-Speech Tagging

This chapter introduces a revision learning method that solves multi-class classification problems efficiently by combining a model with high generalization capacity and a model with small computational cost. This method uses the high generalization capacity model to revise the output of the small cost model to achieve high performance with small computational cost. Experimental results on word segmentation and POS tagging of Japanese and POS tagging of English with revision learning are reported.

5.1. Introduction

In Chapter 2, we applied SVMs to English POS tagging and achieved high accuracy. However, the method has some problems; it cannot be applied to Japanese word segmentation directly and its computational cost is very large. Although SVMs have good performance, their computational cost is large and this is a weakness of SVMs.

In general, a trade-off between capacity and computational efficiency of learning models exists. For example, SVMs have relatively high generalization capacity, but have high computational cost. Learning models with higher capacity may

not be of practical use because of their unreasonable computational cost. This problem becomes more serious when a large amount of training data is available. On the other hand, Markov models have lower computational cost, but they have lower capacity and difficulty in handling data with a large number of features.

To solve this problem, we propose a revision learning method which combines a model with high generalization capacity and a model with small computational cost to achieve high performance with small computational cost. This method is based on the idea that processing an entire target task using a model with higher capacity is wasteful and costly, that is, if a large portion of the task can be processed easily using a model with small computational cost, it should be processed so, and only difficult portions should be processed using the model with higher capacity.

Revision learning can handle general multi-class classification problems, which include POS tagging, text categorization and many other tasks in natural language processing. Furthermore, it can also be applied to word-based Japanese word segmentation which cannot be handled as a simple multi-class classification problem.

This chapter is organized as follows: Section 5.2 describes the general multi-class classification problem and the one-versus-rest method which is known as one of the solutions for the problem. Section 5.3 introduces revision learning, and discusses how to combine learning models. Section 5.4 describes one way to conduct Japanese word segmentation with revision learning. Section 5.5 shows experimental results on word segmentation and POS tagging with English and Japanese corpora. Section 5.6 discusses related work, and Section 5.7 gives conclusion.

5.2. Multi-Class Classification Problem and the One-versus-Rest Method

Let us consider the problem to decide the class of example $\mathbf{x}$ among multiple candidate classes. Such a problem is called a multi-class classification problem. Many tasks in natural language processing such as POS tagging can be regarded as a multi-class classification problem. When we only have binary classification

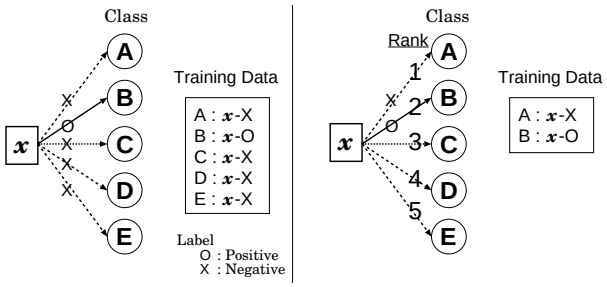

Figure 5.1. One-versus-Rest Method (left) and Revision Learning (right)

algorithm at hand, we have to reformulate a multi-class classification problem into a binary classification problem. We assume a binary classifier $f(\mathbf{x})$ that returns a positive or negative real value as a predicted class of $\mathbf{x}$, where the sign of the value indicates the class and the absolute value $|f(\mathbf{x})|$ reflects the confidence of the classification.

The one-versus-rest method is known as one of the solutions for this problem (Allwein et al., 2000). For one training example, this method creates a positive training example for the true class and negative training examples for the other classes. As a result, positive and negative examples for each class are generated. Suppose we have five candidate classes A, B, C, D and E, and the true class of $\mathbf{x}$ is B. Figure 5.1 (left hand side) shows the created training examples. Note that there are only binary classes (positive and negative) in contrast with the original problem (five classes). Then a binary classifier is trained for each class using the training examples, and five classifiers are created in this example. Given a test example $\mathbf{x}'$, all the classifiers classify the example whether it belongs to a specific class or not. Its class is decided by the classifier that gives the largest value of $f(\mathbf{x}')$. The algorithm is shown in Figure 5.2 in a pseudo-code.

This method has the problem of being computationally costly in training, because the negative examples are created for all the classes other than the true class, and the total number of the training examples becomes large (which is equal to the number of original training examples multiplied by the number of classes). The computational cost in testing is also large, because all the classifiers have to work on each test example.

```
# Training Procedure of One-versus-Rest
# This procedure is given training examples {(x_i, y_i)}, and creates
# classifiers.
#   C = {c_0, ..., c_{k-1}}:  the set of classes,
#   x_i:  the ith training example,
#   y_i ∈ C:  the class of x_i,
#   k:  the number of classes,
#   l:  the number of training examples,
#   f_c(·):  the binary classifier for the class c (see the text).
procedure Train_OVR({(x_0, y_0), ..., (x_{l-1}, y_{l-1})})
begin
  # Create the training data with binary label
  for i := 0 to l - 1
  begin
    for j := 0 to k - 1
    begin
      if c_j ≠ y_i then
        Add x_i to the training data of the class c_j as a negative example.
      else
        Add x_i to the training data of the class c_j as a positive example.
    end
  end
  # Train the binary classifiers
  for j := 0 to k - 1
    Train the classifier f_{c_j}(·) using the training data.
end

# Test Function of One-versus-Rest
# This function is given a test example and returns the predicted
# class of it.
#   C = {c_0, ..., c_{k-1}}:  the set of classes,
#   x:  the test example,
#   k:  the number of classes,
#   f_c(·):  binary classifier trained with the above algorithm.
function Test_OVR(x)
begin
  for j := 0 to k - 1
    confidence_j := f_{c_j}(x)
  return c_{argmax_j confidence_j}
end
```

Figure 5.2. One-versus-Rest Algorithm

5.3. Revision Learning

As discussed in the previous section, the one-versus-rest method has the problem of computational cost. This problem becomes more serious when computationally expensive binary classifiers are used or when a large amount of data is used. To cope with this problem, let us consider the task of POS tagging. Most portions of POS tagging are not so difficult, and a simple POS-based Markov models achieve more than 95% accuracy simply using the POS context (Brants, 2000). This means that the low capacity model is enough to do most portions of the task, and we need not use a highly accurate but computationally expensive algorithm in every portion of the task. This is the base motivation of the revision learning method we are proposing here.

Revision learning uses a binary classifier with higher capacity to revise the errors made by a stochastic model with lower capacity as follows: During the training phase, firstly, a ranking is assigned to the candidate classes of a training example by the stochastic model, that is, the candidate classes are sorted in descending order of its conditional probability given the example. Next, training data for revisers (binary classifiers) are created by checking the ranked classes in the order, as follows. If the highest ranked class is incorrect (i.e. it is not equal to the true class of the example), the example is added to the training data for the class as a negative example, and the next ranked class is checked recursively. If the class is correct, the example is added to the training data for the class as a positive example, and the remaining lower-ranked classes are not taken into consideration (Figure 5.1, right hand side). Thus, training data of the binary classifiers is created for each class. Binary classifiers are trained using these training data. Note that each classifier is a pure binary classifier regardless of the number of classes in the original problem. The binary classifiers are trained just for answering whether the outputs from the stochastic model are correct or not.

During the testing phase, firstly the ranking of the candidate classes for a given example is assigned by the stochastic model as in the training. Then the binary classifiers classify the example according to the ranking. If the classifier associated with the highest ranked class answers the example as incorrect, the next highest ranked class becomes the next candidate for checking. But if the

example is classified as correct, the class of the classifier is returned as the answer for the example. The algorithm is shown in Figure 5.3.

The amount of training data generated in the revision learning can be much smaller than that in one-versus-rest. Since, in revision learning, negative examples are created only when the stochastic model fails to assign the highest probability to the correct POS tag, whereas negative examples are created for all but one class in the one-versus-rest method. Moreover, testing time in revision learning is shorter, because only one classifier is called as far as it answers as correct, but all the classifiers are called in the one-versus-rest method.

5.4. Word Segmentation and POS Tagging with Revision Learning

We introduced revision learning for multi-class classification in the previous section. However, word-based Japanese word segmentation cannot be handled as a multi-class classification problem, because words in a sentence are not separated by spaces in Japanese and Japanese lexical analyzers have to not only decide the POS tag of the words but also segment the sentence into words. So in this section, we describe how to apply the revision learning to word segmentation and POS tagging of Japanese.

For a given sentence, a lattice consisting of all possible words can be built using a word dictionary as in Figure 5.4[1]. Word segmentation and POS tagging are conducted by choosing the most likely path on it. We adopt Markov models as the stochastic model and SVMs as the binary classifier for revision learning. For any sub-paths from the beginning of the sentence (BOS) in the lattice, its generative probability can be calculated using Markov models (Nagata, 1999a). We first pick up the end node of the sentence as the current state node, and repeat the following revision learning process backward until the beginning of the sentence: Rankings are calculated by Markov models for all the nodes connected to the current state node, and the best of these nodes is identified by the revision learning using the SVM classifiers. The selected node then becomes the current

[1]Although only a POS tag of a word is shown in each node of the lattice, each node can have other attributes of the word such as inflection forms if the dictionary has such information.

```
# Training Procedure of Revision Learning
# This procedure is given training examples {(x_i, y_i)}, and creates
# classifiers.
#   C = {c_0, ..., c_{k-1}}:  the set of classes,
#   x_i:  the ith training example,
#   y_i ∈ C:  the class of x_i,
#   k:  the number of classes,
#   l:  the number of training examples,
#   n_i:  the ordered indexes of C (see the following code),
#   f_c(·):  the binary classifier for the class c (see the text).
procedure Train_RL({(x_0, y_0), ..., (x_{l-1}, y_{l-1})})
begin
  # Create the training data with binary label
  for i := 0 to l − 1
  begin
    Call the stochastic model to obtain the ordered indexes {n_0, ..., n_{k-1}}
    such that P(c_{n_0}|x_i) ≥ ··· ≥ P(c_{n_{k-1}}|x_i).
    for j := 0 to k − 1
    begin
      if c_{n_j} ≠ y_i then
        Add x_i to the training data of the class c_{n_j} as a negative example for f_{c_{n_j}}(·).
      else
      begin
        Add x_i to the training data of the class c_{n_j} as a positive example for f_{c_{n_j}}(·).
        break
      end
    end
  end
  # Train the binary classifiers
  for j := 0 to k − 1
    Train the classifier f_{c_j}(·) using the training data.
end

# Test Function of Revision Learning
# This function is given a test example and returns the predicted
# class of it.
#   C = {c_0, ..., c_{k-1}}:  the set of classes,
#   x:  the test example,
#   k:  the number of classes,
#   n_i:  the ordered indexes of C (see the following code),
#   f_c(·):  binary classifier trained with the above algorithm.
function Test_RL(x)
begin
  Call the stochastic model to obtain the ordered indexes {n_0, ..., n_{k-1}}
  such that P(c_{n_0}|x) ≥ ··· ≥ P(c_{n_{k-1}}|x).
  for j := 0 to k − 1
    if f_{c_{n_j}}(x) > 0 then
      return c_{n_j}
  # Return the class with the highest likelihood if no solutions
  # are found
  return c_{n_0}
end
```

Figure 5.3. Revision Learning Algorithm

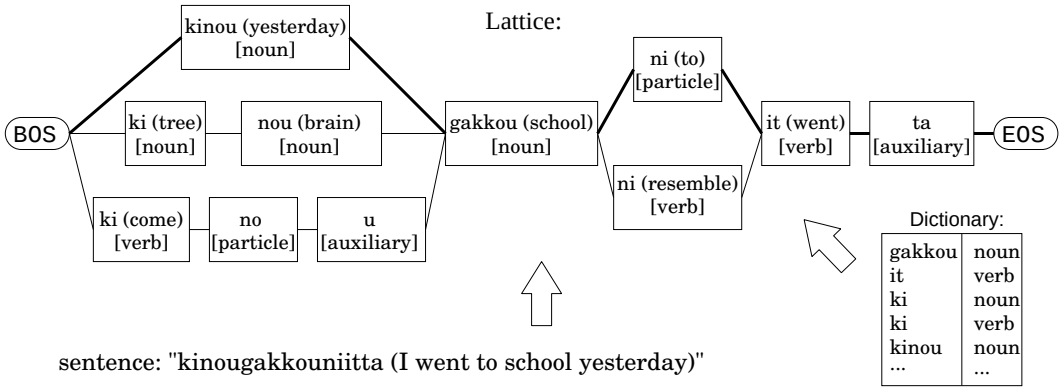

Figure 5.4. Example of Lattice for Japanese Word Segmentation and POS Tagging

state node for the next round. This can be seen as SVMs deciding whether two adjoining nodes in the lattice are connected or not according to the ranking assigned by Markov models.

In word segmentation and POS tagging of Japanese, for any given word w, we use the following features for the SVMs:

1. the POS tags, the lexical forms and the inflection forms of the two words preceding w;
2. the POS tags and the lexical forms of the two words following w;[2]
3. the lexical form and the inflection form of w.

The preceding words and their attributes (POS tags and inflection forms) are unknown because the processing is conducted from the end of the sentence, but Markov models can predict the most likely preceding words, and we use them as features for the SVMs.

English POS tagging is regarded as a special case of word segmentation and POS tagging of Japanese where the segmentation is done in advance, and can be conducted in the same way. In English POS tagging, given a word w, we use the following features for the SVMs:

[2]The inflection forms of the words following w are not used because inflection forms generally affect only their following words in Japanese.

1. the POS tags and the lexical forms of the two words preceding w, which are given by Markov models;

2. the POS tags and the lexical forms of the two words following w;

3. the lexical form of w, and the prefixes and suffixes of up to four characters, the existence of numerals, capital letters and hyphens in w.

5.5. Experiments

This section gives experimental results of POS tagging of English and word segmentation and POS tagging of Japanese with revision learning. Experiments of English are performed on the Penn Treebank WSJ corpus. Experiments of Japanese are performed on the RWCP corpus and the Kyoto University corpus. The following measures are used for evaluation of Japanese word segmentation and POS tagging:

$$\text{recall} = \frac{\langle \text{number of correct words in system's output} \rangle}{\langle \text{number of words in test data} \rangle}.$$

$$\text{precision} = \frac{\langle \text{number of correct words in system's output} \rangle}{\langle \text{number of words in system's output} \rangle}$$

$$\text{F-measure} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}.$$

5.5.1 Experiments on the Penn Treebank WSJ Corpus (English)

Experiments on English POS tagging are performed with the Penn Treebank WSJ corpus which has 50 POS tags. The corpus is randomly separated into training data of 41,342 sentences (999,984 tokens) and test data of 11,771 sentences (284,808 tokens). The dictionary for Markov models is constructed from all the words in the training data.

T3 of ICOPOST release 0.9.0 (Schröder, 2001) is used as the stochastic model for the ranking stage. This is equivalent to POS-based second order Markov models. SVMs with the second order polynomial kernel are used as the binary classifiers.

The results are compared to two systems; TnT (Brants, 2000) which is based on second order Markov models, and the POS tagger discussed in Chapter 2 which uses SVMs with one-versus-rest.

The accuracies of those systems for known words, unknown words and all the words are shown in Table 5.1. The accuracies for both known words and unknown words are improved through revision learning. However, revision learning could not surpass the one-versus-rest. The main difference in the accuracies stems from those for unknown words. The reason for that seems to be that the dictionary of Markov models for POS tagging is obtained from the training data, as a result, no unknown words exist in the training data, and the Markov models never make mistake for unknown words during the training of the binary classifiers. Therefore, no examples of unknown words are available in the training data for the SVMs reviser. This is problematic: Though the Markov models handle unknown words with an exceptional method, SVMs cannot learn about errors made by the unknown word processing of the Markov models. To cope with this problem, we force the Markov models to make mistakes for unknown words by eliminating low frequency words from the dictionary. We eliminated the words appearing only once in the training data from the dictionary in the training phase, so as to make SVMs learn about unknown words. The results are shown in Table 5.1 (row "cutoff-1"). Such procedure improves the accuracies for unknown words.

One advantage of revision learning is its small computational cost. We compare the computation time with the Markov models and the one-versus-rest. We also use SVMs with a linear kernel function that has lower capacity but lower computational cost compared to the second order polynomial kernel SVMs. The experiments are performed on an Alpha 21164A 500MHz processor. Table 5.2 shows the total number of training examples, training time, testing time and accuracy for each of the five systems. The training time and the testing time of revision learning are considerably smaller than those of the one-versus-rest. Using linear kernel, the accuracy decreases a little, but the computational cost is much lower than that with the second order polynomial kernel.

Table 5.1. Results of English POS Tagging

	Accuracy (Known / Unknown)	Number of Errors
T3 (Original)	96.59% (96.90% / 82.74%)	9,720
T3 with RL	96.93% (97.23% / 83.55%)	8,734
T3 with RL (cutoff-1)	96.98% (97.25% / 85.11%)	8,588
TnT	96.62% (96.90% / 84.19%)	9,626
SVMs (one-versus-rest)	97.11% (97.34% / 86.80%)	8,245

Table 5.2. Computational Cost of English POS Tagging

	Total Number of Examples for SVMs	Training Time (hour)	Testing Time (second)	Accuracy
T3 (original)	—	0.004	89	96.59%
T3 with RL (polynomial kernel)	1,027,840	16	2,089	96.98%
T3 with RL (linear kernel)	1,027,840	2	129	96.94%
TnT	—	0.002	4	96.62%
SVMs (one-versus-rest)	49,999,200	625	55,239	97.11%

5.5.2 Experiments on the RWCP Corpus (Japanese)

We use the RWCP corpus with some additional spoken language data for the experiments on Japanese word segmentation and POS tagging. The corpus has 88 POS tags. The corpus was randomly separated into training data of 33,831 sentences (840,879 tokens) and test data of 3,758 sentences (93,155 tokens). As the dictionary for Markov models, we used IPADIC version 2.4.4 with 366,878 words (Matsumoto and Asahara, 2001) which is originally compiled for the Japanese morphological analyzer ChaSen.

We use two models, a POS bigram model (Nagata, 1999a) and ChaSen version 2.2.8 (Matsumoto et al., 2001) which is based on variable length Markov models, as the stochastic model for the ranking stage, and we use SVMs with the second order polynomial kernel as the binary classifier.

The results of the original systems and those with revision learning (RL) are shown in Table 5.3, which provides recalls, precisions and F-measures for two cases, namely segmentation (i.e. only segmentation of the sentences into words) and tagging (i.e. both segmentation and POS tagging). Results obtained by SVMs with one-versus-rest are not shown in the table, because the method cannot be applied to Japanese word segmentation directly.

When revision learning is used, all the measures are improved for both the POS bigram and ChaSen. Improvement is particularly clear for the tagging task.

The numbers of correctly tagged words (tokens) for each POS category (group of POS tags) in the output of ChaSen with and without revision learning are shown in Table 5.4. Many particles are correctly revised with revision learning. The reason is that the POS tags for particles are often affected by the following words in Japanese, and SVMs can revise such particles because it uses the lexical forms of the following words as the features. This is the advantage of our method compared to simple Markov models, because Markov models have difficulty in handling a lot of features such as the lexical forms of words. The examples of successful revisions are shown in Figure 5.5. The words with asterisks are the revised words. The particle to express a change of state followed by the verb 「なる」(“naru”) are defined not as a particle of the 副詞化 type but a particle of the 格助詞-一般 type, and the particle 「と」(“to”) followed by the verb 「思う」(“omou”; think, believe) is defined as a particle of the 格助詞-引用 type,

Table 5.3. Results on the RWCP Corpus

		Segmentation			Segmentation & Tagging		
		Recall	Precision	F-measure	Recall	Precision	F-measure
POS bigram	Original	98.06%	98.77%	98.42%	95.61%	96.30%	95.96%
	with RL	99.06%	99.27%	99.16%	98.13%	98.33%	98.23%
ChaSen	Original	99.06%	99.20%	99.13%	97.67%	97.81%	97.74%
	with RL	99.22%	99.34%	99.28%	98.26%	98.37%	98.32%

and they were correctly revised. The reason for these correct revisions seems to be that the SVMs properly handled following word features like「なる」and「思う」.

5.5.3 Experiments on the Kyoto University Corpus (Japanese)

Experiments are performed with the Kyoto University corpus version 2.0. We construct two training sets; one consisting of the articles of January 1 and from January 3 to January 8 (total of 7,958 sentences), and the other consisting of all the entire corpus except the articles of January 9 (total of 18,710 sentences). We use the articles of January 9 (total of 1,246 sentences) as the test data. We use the dictionary of Japanese morphological analysis system JUMAN version 3.61 (Kurohashi and Nagao, 1998).

POS bigram is used as the stochastic model, and SVMs with second order polynomial kernel are used as the binary classifiers.

The results are compared to the results of JUMAN, with post-processing using Japanese dependency/case structure analyzer KNP (Kurohashi, 1998) to resolve ambiguities.

The results are shown in Table 5.6. All the measures are improved, especially for tagging similarly to the case on the RWCP corpus. Compared to JUMAN with KNP, all the measures obtained by POS bigram with revision learning are lower in segmentation, but higher in POS tagging.

Table 5.4. The Number of Correctly Tagged Words for Each POS Category in the RWCP Corpus

POS Category	# in Test Data	Original	with RL	Difference
Noun	41,512	40,355	40,556	+201
Prefix	817	781	784	+3
Verb	8,205	8,076	8,115	+39
Adjective	678	632	655	+23
Adverb	779	735	750	+15
Adnominal	378	373	373	0
Conjunction	258	243	243	0
Particle	20,298	19,686	19,942	+256
Auxiliary	4,419	4,333	4,336	+3
Interjection	94	90	91	+1
Symbol	15,665	15,647	15,651	+4
Others	1	1	1	0
Filler	43	36	36	0

Table 5.5. Examples of Revised Words on the RWCP Corpus

	Word	POS Tag	
		Original	Revised
Sentence 1 (“tsuikitto natteshimau”)	つい (tsui)	副詞-一般	
	キッ (kit)	副詞-助詞類接続	
	と (to)	* 助詞-副詞化	助詞-格助詞-一般
	なっ (nat)	動詞-自立	
	て (te)	助詞-接続助詞	
	しまう (shimau)	動詞-非自立	
Sentence 2 (“daredarou toomoi”)	誰 (dare)	名詞-代名詞-一般	
	だろ (daro)	助動詞	
	う (u)	助動詞	
	と (to)	* 助詞-格助詞-一般	助詞-格助詞-引用
	思い (omoi)	動詞-自立	

Table 5.6. Results on the Kyoto University Corpus

		Segmentation			Segmentation & Tagging		
		Recall	Precision	F-measure	Recall	Precision	F-measure
POS Bigram	Original	97.72%	97.13%	97.43%	93.21%	92.65%	92.93%
(7,958 sentences)	with RL	98.40%	97.77%	98.08%	95.62%	95.01%	95.31%
POS Bigram	Original	97.86%	97.20%	97.53%	93.31%	92.67%	92.99%
(18,710 sentences)	with RL	98.67%	97.99%	98.33%	96.00%	95.34%	95.67%
JUMAN		98.88%	98.52%	98.70%	93.93%	93.58%	93.75%
JUMAN+KNP		98.89%	98.53%	98.71%	95.10%	94.75%	94.93%

Table 5.7. Side Effects of Revision Learning

Corpus	WSJ Corpus	RWCP Corpus	
Stochastic Model	T3	POS bigram	ChaSen
(a) # of Successfully Revised Tokens	2,927	2,572	684
(b) # of Wrongly Revised Tokens	1,795	229	139
(c) # of Not Successfully Revised Tokens	6,793	1,515	1,483

5.5.4 Side Effects of Revision Learning

In the previous experiments, the total accuracies were always improved using revision learning. However, in some cases, the SVM reviser may have wrongly revised correct outputs of the Markov models. We examine errors made in the previous experiments, and classify them into three cases:

(a) a Markov models' output was false and SVMs revised it successfully,

(b) a Markov models' output was true but SVMs wrongly revised it,

(c) a Markov models' output was false and SVMs couldn't revise it successfully.

The results are shown in Table 5.7. The ratio of successfully revised tokens (a) for wrongly revised tokens (b) varies in different corpora and stochastic models, but the number of wrongly revised tokens is not small. One of the causes may be robustness of the models for noisy data. Table 5.8 shows one example of the wrong revisions in English POS tagging and inconsistent data found in the

Table 5.8. Example of Side Effects

Test Data	
Correct Tagging	Esso/NNP said/VBD the/DT Whiting/NNP field/NN ...
Output of the MMs	Esso/NNP said/VBD the/DT Whiting/VBG field/NN ...
Output of the SVMs	*Esso/NN* said/VBD the/DT Whiting/VBG field/NN ...
Training Data	
Sentence 1 (error)	*Esso/NN* said/VBD the/DT fields/NNS were/VBD ...
Sentence 2	... Group/NNP ;/: Esso/NNP Resources/NNP Canada/NNP ...
Sentence 3	Three/CD companies/NNS ,/, Esso/NNP Resources/NNPS ...

training corpus which might cause the error. In this case, Markov models (MMs) outputted the correct tag "NNP (proper noun)" for the word "Esso", but SVMs wrongly revised it as "NN (common noun)". Only three sentences in the training data contained the word "Esso", which is shown in Table 5.8 as Sentence 1, Sentence 2 and Sentence 3. In Sentence 1, the word "Esso" is tagged as "NN", which seems to be an error of the corpus, and this noise probably caused the SVMs' wrong revision because the test sentence is quite similar to Sentence 1 and features for the word "Esso" are the same in both the test sentence and Sentence 1. Markov model-based POS taggers decide their outputs based on state transition probabilities and symbol emission probabilities, and are relatively not affected by exceptional noises. On the other hand, SVMs are example-based learning algorithm, and can learn exceptional events which occur rarely in training data, but are more easily affected by these noises.

5.6. Related Work

Our proposal is to revise the outputs of a stochastic model using a binary classifier. Brill (1995) studied transformation-based error-driven learning (TBL), which conducts POS tagging by applying the transformation rules to the POS tags of a given sentence, and has a resemblance to revision learning in that the second model revises the output of the first model. However, our method differs from TBL in two ways. First, our revision learner simply answers whether a given pattern is correct or not, and any types of binary classifiers are applicable. Second,

in our model, the second learner is applied to the output of the first learner only once. In contrast, rewriting rules are applied repeatedly in TBL.

Collins conducted parsing and named-entity tagging based on a reranking approach (Collins, 2000; Collins, 2002; Collins and Duffy, 2002). For a given sentence, he obtained candidate parse sequences or tag sequences for the sentence using a generative model, and reranked them using a discriminative model to improve the results. He reported the method achieved 13% and 17.7% relative decrease in error rate for WSJ Penn Treebank parsing and named-entity tagging of web data respectively over state-of-the-art methods. Revision learning resembles his method in that a stochastic (generative) model is used to reduce search space in the first step, and a discriminative model with rich features are used to obtain the final answer in the second step. One advantage of his method is that global features for a whole sentence can be incorporated easily into the discriminative model. However, all the candidates are examined in the second step. Revision learning examines the candidates using a discriminative model until one answer found, and solves multi-class classification problems with lower computational cost.

Recently, combinations of multiple learners have been studied to achieve high performance (Alpaydm, 1998). Such methodologies to combine multiple learners can be distinguished into two approaches; one is the multi-expert method and the other is the multi-stage method. In the former, each learner is trained and answers independently, and the final decision is made based on those answers. In the latter, the multiple learners are ordered in series, and each learner is trained and answers only if the previous learner rejects the examples. Revision learning belongs to the latter approach. In POS tagging, some studies using the multi-expert method were conducted (van Halteren et al., 2001; Màrquez et al., 1999), and Brill and Wu (1998) combined three models; maximum entropy models, TBL and trigram, and achieved higher accuracy than any of the three learners. Regarding the multi-stage methods, cascading (Alpaydin and Kaynak, 1998) is a well known method, and Even-Zohar and Roth (2001) proposed the sequential learning model and applied it to POS tagging. Their methods differ from revision learning in that each learner behaves in the same way and more than one learner is used in their method, while in revision learning, the stochastic model assigns

rankings to candidates and the binary classifier selects the output. Furthermore, mistakes made by a former learner are fatal in their methods, but is not so in revision learning because the binary classifier works to revise them.

The advantage of the multi-expert method is that each learner can help each other even if it has some weakness, and generalization errors can be decreased. On the other hand, the computational cost becomes large because each learner is trained using entire training data and answers for every test data. In contrast, multi-stage methods can decrease the computational cost, and seem to be effective when a large amount of data is used or when learners with high computational cost such as SVMs are used.

5.7. Conclusion

In this chapter, we proposed a revision learning method which combines a stochastic model and a binary classifier to achieve higher performance with low computational cost for large-scale multi-class classification problems. We applied it to word segmentation and POS tagging of Japanese and POS tagging of English, in which a large number of examples and classes are handled, and showed improvement of accuracy with small computational cost.

Compared to the conventional one-versus-rest method, revision learning has lower computational cost with comparable accuracy. Furthermore, it can be applied not only to simple multi-class classification, but also to a wider variety of problems such as Japanese word segmentation.

Chapter 6

Corpus Error Detection Using Support Vector Machines

While the corpus-based research relies on human annotated corpora, it is often said that a non-negligible amount of errors remain even in frequently used corpora such as Penn Treebank. Detecting errors in annotated corpora is important for corpus-based natural language processing in order to improve quality of the corpora. In this chapter, we propose a method for detecting errors in corpora using SVMs. This method is based on the idea of extracting exceptional elements that violate consistency of corpora. We propose a method using SVMs which assigns a weight to each element in a POS tagged corpus and finds errors. We apply the method to English and Japanese POS-tagged corpora and achieve high precision in error detection.

6.1. Introduction

Corpora are widely used in natural language processing today. For example, many statistical part-of-speech (POS) taggers have been developed and they use corpora as the training data to obtain statistical information or rules (Brill, 1995; Ratnaparkhi, 1996). For natural language processing systems based on a corpus, the quantity and quality of the corpus largely affect their performance. In general, corpora are annotated by hand, and therefore error-prone. These errors are problematic for corpus-based systems. The errors become false training ex-

amples and deteriorate the performance of the systems. Furthermore, incorrect instances may be used as testing examples and prevent the accurate measurement of performance. Many studies and improvements have been conducted for POS tagging, and major methods of POS tagging achieve accuracy of 96–97% on the Penn Treebank WSJ corpus, but obtaining higher accuracies is difficult (Ratnaparkhi, 1996). It has been mentioned that the limitation is largely caused by inconsistencies in the corpus (Ratnaparkhi, 1996; Padró and Màrquez, 1998; van Halteren et al., 2001). Therefore, correcting errors in a corpus and improving its quality is important. However, to find and correct errors in corpora by hand is costly, since the size of corpora is usually very large. Hence, automatic detection of errors in corpora is necessary.

One of the approaches for corpus error detection is use of machine learning techniques (Abney et al., 1999; Matsumoto and Yamashita, 2000; Ma et al., 2001). These methods regard difficult elements to be learned for learning models (e.g. boosting or neural networks) as corpus errors. Abney et al. (1999) studied corpus error detection using boosting. Boosting assigns weights to training examples, and the weights are large for examples that are difficult to be classified. Mislabeled examples caused by annotators tend to be difficult examples to be classified and the authors conducted error detection of POS tags and PP attachment information in a corpus by extracting examples with large weights.

Some probabilistic approaches for corpus error detection have also been studied (Eskin, 2000; Murata et al., 2000). Eskin (2000) conducted corpus error detection using an anomaly detection technique. He supposed that all the elements in a corpus are generated by a mixture model consisting of two distributions, a majority distribution (typically a structured distribution) and an anomalous distribution (a uniform random distribution), and erroneous elements are generated from the anomalous distribution. For each element in a corpus, likelihood is calculated in two cases when the element is generated from the majority distribution and from the anomalous one. The element is regarded as an error if the likelihood in the latter case is large enough.

In this chapter, we focus on detection of errors in corpora annotated with POS tags, and propose a method for corpus error detection using SVMs. SVMs are one of machine learning models and applied to many natural language processing

tasks with success recently. In the next section, we explain a method for corpus error detection using SVMs.

6.2. Corpus Error Detection Using Support Vector Machines

Training data for corpus error detection is usually not available, so we cannot solve it with supervised learning. We consider in the following way: In general, a corpus is built according to a set of guidelines, thus it should be consistent in some sense. If there is an exceptional element in the corpus that jeopardizes consistency, it is likely to be an error. Therefore, corpus error detection can be conducted by detecting exceptional elements that causes inconsistency. Such exceptional elements can be detected using a machine learning algorithm, by extracting elements that the learning algorithm hardly classify.

While this is a simple and straightforward approach and any machine learning method is applicable to this task, we use SVMs as the learning algorithm. The advantage of using SVMs is the following: In our setting, each element in an annotated corpus receives a weight value according to the SVM algorithm, and these weights can be used as the confidence level of erroneous examples. By effectively using those weights the inspection of the erroneous parts can be undertaken in the order of the confidence level, so that an efficient checking of the corpus is possible. We believe this is a particular advantage of our method compared with the methods that use other machine learning methods.

6.2.1 Detecting Exceptional Elements with Support Vector Machines

Support Vector Machines (SVMs) are supervised machine learning models for binary classification. Given l training examples consists of feature vectors $\mathbf{x}_i \in \mathbf{R}^L$ and their labels $y_i \in \{+1, -1\}$, SVMs map them into a high dimensional space by a nonlinear function $\mathbf{\Phi}(\mathbf{x})$ and linearly separate them. The optimal hyperplane to

separate them is found by solving the following quadratic programming problem:

$$\begin{aligned} \underset{\alpha_1,\cdots,\alpha_l}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{l}\alpha_i, \\ \text{subject to} \quad & 0 \le \alpha_i \le C \quad (1 \le i \le l), \\ & \sum_{i=1}^{l}\alpha_i y_i = 0, \end{aligned}$$

where the function $K(\mathbf{x}_i, \mathbf{x}_j)$ is the inner product of the nonlinear function ($K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j)$) called a kernel function. The constant C controls the training and generalization errors, and is the upper bound of α_i. Given a test example $\mathbf{x}$, its label y is decided by summing the inner products of the test example and the training examples weighted by α_i:

$$y \;=\; \mathrm{sgn}\Big(\sum_{i=1}^{l}\alpha_i y_i K(\mathbf{x}_i,\mathbf{x}) + b\Big),$$

where b is a threshold value. Thus, SVMs assign a weight α_i to each training example. The weights are large for examples that are hard for SVMs to classify, that is, exceptional examples in the training data have large weights. We conduct corpus error detection using the weights. To detect exceptional elements in a corpus annotated with POS tags, we first construct an SVM model for POS tagging using all the elements in the corpus as training examples. Note that each example corresponds to a word in the corpus. Then SVMs assign weights to the examples, and large weights are assigned to difficult examples. Finally, we extract examples with large weights greater than or equal to a threshold value θ.

We use a revision learning method (described in Chapter 5) for POS tagging with SVMs. This method creates training examples of SVMs with binary labels for each POS tag class using a stochastic model (e.g. Markov models) as follows: Each word in a corpus becomes a positive example of its POS tag class. We then build a simple stochastic POS tagger based on Markov models (e.g., POS bigram or trigram), and words in the corpus that the stochastic model fails to attach the correct POS tags are collected as negative examples of the POS tag class. In such way, revision learning makes a model of SVMs to revise outputs of the stochastic model. Let us consider the following example sentence:

```
"11/CD million/CD yen/NNS are/VBP expected/VBN"
```

We suppose that a stochastic model attaches POS tags incorrectly; the following incorrect tags are returned as the best answer and the above correct tags are returned as the second best answer by the stochastic model:

```
"11/CD million/CD yen/NN are/VBP expected/VBN"
```

In this case, the following training examples are created for SVMs (each line corresponds to an example):

```
<Class> <Label> <Feature Vector>
CD      +1      (word:11, word-1:BOS, ...)
CD      +1      (word:million, word-1:11, ...)
NN      -1      (word:yen, word-1:million, ...)
NNS     +1      (word:yen, word-1:million, ...)
VBP     +1      (word:are, word-1:yen, ...)
VBN     +1      (word:expected, word-1:are, ...)
```

Thus, the positive and negative examples are created for each class (POS tag), and a model of SVMs is trained for each class using the training examples. We use the same features as in Chapter 5.

6.2.2 Extracting Inconsistencies

So far, we discussed how to detect exceptional elements in a corpus. However, it is insufficient and inconvenient for corpus error detection, because an exceptional element is not always an error, that is, an exceptional element may be a correct exceptional element. Furthermore, it is often difficult to judge whether an element in corpora is a true error or not when only the exceptional element is shown alone. To solve these problems, we extract not only an exceptional element but also another similar element that is inconsistent with the exceptional element. If the exceptional element is correct, the second element is likely to be an error, and vice versa. It is relatively easy to judge whether a given element contains errors or not when these conflicting elements are shown at the same time.

We assume that an inconsistency occurs when two examples have similar features but have different labels. The similarity between two examples $\mathbf{x}_i$ and

$\mathbf{x}_j$ in SVMs is measured by the following distance:

$$\begin{aligned} d(\mathbf{x}_i, \mathbf{x}_j) &= \sqrt{\|\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{\Phi}(\mathbf{x}_j)\|^2}, \\ &= \sqrt{K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)}. \end{aligned}$$

We can extract inconsistencies from a corpus as follows: Given an example $\mathbf{x}$ which was detected as an exceptional example (following the proposal in the previous subsection), we extract an example $\mathbf{z}$ with the smallest values of the distance $d(\mathbf{x}, \mathbf{z})$ from the examples whose labels are different from $\mathbf{x}$. Intuitively, $\mathbf{z}$ is the closest but opposite class example to $\mathbf{x}$ in the higher dimensional space of the SVMs, and may be the cause for $\mathbf{x}$ to be attached a large weight.

6.3. Experiments

We conduct experiments of corpus error detection using the Penn Treebank WSJ corpus (in English), the RWCP corpus (in Japanese) and the Kyoto University Corpus (in Japanese). In the following experiments, we use SVMs with the second order polynomial kernel, and the upper bound value C is set to 1.

6.3.1 Experiments on the Penn Treebank WSJ Corpus (English)

Experiments are performed on the Penn Treebank WSJ corpus, which consists of 53,113 sentences (1,284,792 tokens).

We create models of SVMs for POS tagging using the corpus with revision learning. The distribution of the obtained weights α_i is shown in Figure 6.1. The values of α_i concentrate near the lower bound zero and the upper bound $C(= 1.0)$. The examples with α_i near the upper bound seem to be exceptional. Therefore, we regarded the examples with $\alpha_i \geq 0.5$ as exceptional examples (i.e. $\theta = 0.5$). As a result, 1,740 elements were detected as errors. Figure 6.2 shows an example of detected errors with a browsing tool we made. A detected inconsistency pair is shown in the lower part of the screen. We examined by hand whether the detected errors are true errors or not for the first 200 elements in the corpus from the detected 1,740 elements, and 199 were actual errors and 1 was not. The

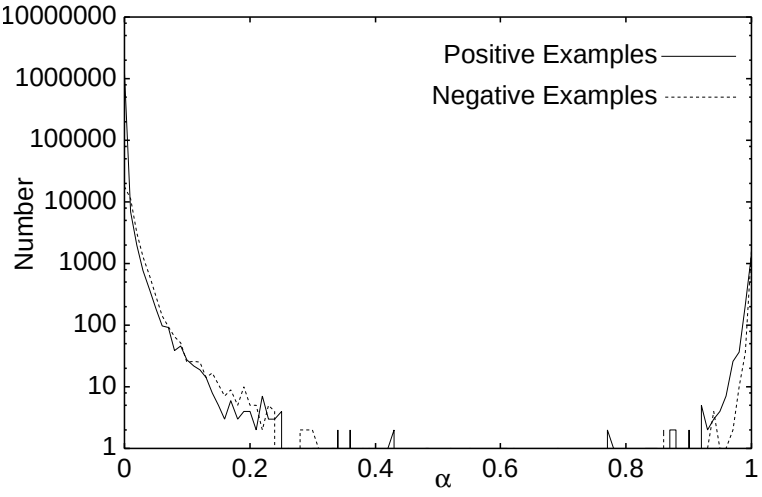

Figure 6.1. Distribution of the Value α_i on the WSJ Corpus

precision (the ratio of correctly detected errors for all of the detected errors) is 99.5%. Examples of correctly detected errors and incorrectly detected errors from the corpus are shown in Table 6.1. The underlined words were detected as errors. To judge whether they are true errors or not is easy by comparing the pairs of examples that contradict each other.

To examine the recall (the ratio of the correctly detected errors for all of actual errors existing in the corpus), we conduct another experiments on an artificial data. We made the artificial data by randomly changing the POS tags of

Table 6.1. Examples of Correctly Detected Errors and Incorrectly Detected Errors in the WSJ Corpus

Correctly Detected Errors	
pay about 11 million yen/NNS ($ 77,000	budgeted about 11 million yen/NN ($ 77,500
, president and chief/JJ executive officer of	named president and chief/NN executive officer
for its fiscal first quarter ended/VBN Sept. 30	its first quarter ended/VBD Sept. 30 was
Incorrectly Detected Errors	
EOS 3/LS . EOS Send your child to	Nov. 1-Dec . EOS 3/CD . EOS

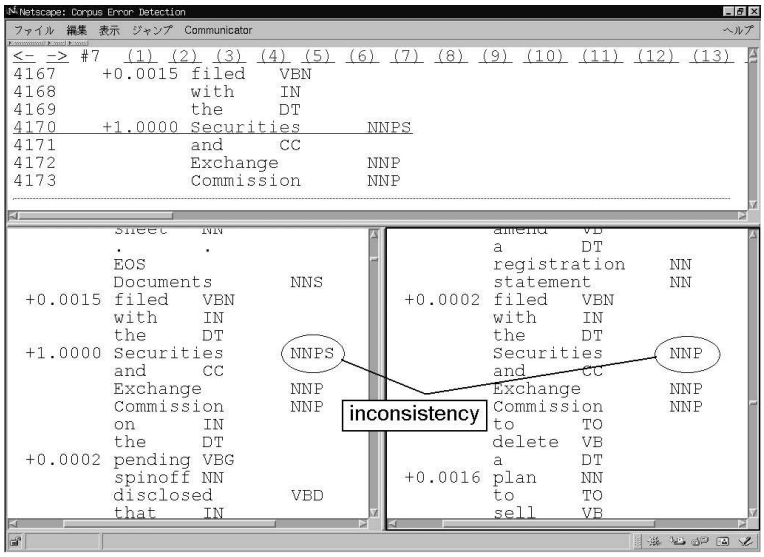

Figure 6.2. Example of Corpus Error Detection

Table 6.2. Recall for the Artificial Data

θ_α	# of Correctly Detected Errors	Recall
1.0	607	4.7%
0.5	1520	11.8%
0.2	1555	12.1%
0.1	1749	13.6%
0.05	2381	18.5%

randomly selected ambiguous tokens (those words which have multiple possible POS tags in the corpus) in the WSJ corpus. The POS tags of 12,848 tokens (1% of the whole corpus) are changed, and the results are shown in Table 6.2 for various values of θ[1]. For the smaller threshold θ, the larger recall was obtained, but the values are not high.

[1]Precisions cannot be measured automatically because actual errors as well as the artificially mixed errors are also detected.

Table 6.3. Number of Detected Errors in the RWCP Corpus

θ_α	Correct Detection	(Segmentation / POS Tag)	Incorrect Detection	Precision
1.0	110	(30 / 80)	8	93.2%
0.5	165	(43 / 122)	11	93.8%
0.2	171	(45 / 126)	12	93.4%
0.1	188	(51 / 137)	31	85.8%
0.05	300	(73 / 227)	73	80.4%

6.3.2 Experiments on the RWCP Corpus (Japanese)

We conduct experiments with the RWCP corpus, which is a Japanese corpus consists of 35,743 sentences (921,946 words).

The distribution of the weights α_i is shown in Figure 6.3. The distribution of α_i shows the same tendency as in the case of the WSJ corpus.

We conducted corpus error detection for various values of θ, and examined by hand whether the detected errors are true errors or not. The results are shown in Table 6.3, where the correctly detected errors are distinguished into two types; one is the word segmentation error and the other is the POS tag error, since Japanese has two kinds of ambiguities, word segmentation and POS tagging. Precisions of more than 80% are obtained, and the number of POS tag errors is larger than that of segmentation errors.

Examples of correctly detected errors and incorrectly detected errors from the corpus are shown in Table 6.4. The underlined words were detected as errors. In the examples of correctly detected errors, both segmentation errors (upper) and POS tag errors (lower) are shown. The example of incorrectly detected errors shows a limitation of our method. We use the two words on each side of the current word as features for SVMs. In the examples, the two words on each side are the same and only the POS tag of the current word is different, so that SVMs cannot distinguish the difference and regard them as errors (inconsistencies).

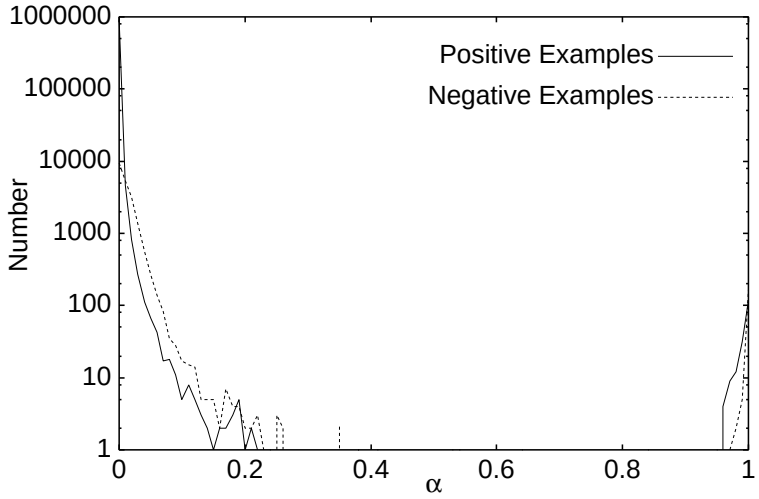

Figure 6.3. Distribution of the Value α_i on the RWCP Corpus

Table 6.4. Examples of Correctly Detected Errors and Incorrectly Detected Errors in the RWCP Corpus

Correctly Detected Errors	
用 |多目的/Common-Noun| 車	用 | 多/Prefix|目的/Common-Noun| 車
大阪 | 市 |中央/ProperNoun-Place| 区 | の	大阪 | 市 |中央/Common-Noun| 区 | の
Incorrectly Detected Errors	
容疑 | 者 |と/Coord-Particle| 二 | 人 | の | 間	容疑 | 者 |と/Case-Particle| 二 | 人 | で

Table 6.5. Number of Detected Errors on the Kyoto University Corpus in the Repeated Experiment

Round	1	2	3	4
Correct Detection	85	11	2	0
(Segmentation Error)	(21)	(2)	(0)	(0)
(POS Tag Error)	(64)	(9)	(2)	(0)
Incorrect Detection	0	0	0	0
Total	85	11	2	0

6.3.3 Experiments on the Kyoto University Corpus (Japanese)

Experiments are performed on a portion of the Kyoto University corpus version 2.0, consisting of the articles of January 1, and from January 3 to January 9 (total of 9,204 sentences, 229,816 words). We set the value of θ to 0.5.

By repeating corpus error detection and correction of the detected errors by hand, new errors that are not detected previously may be detected. To examine this, we repeated corpus error detection and correction by hand. Table 6.5 shows the result. All the detected errors in all rounds were true errors, that is, the precision is 100%. By applying the corpus error detection repeatedly, the number of detected errors decreases rapidly, and no errors are detected in the fourth round. In short, even if we repeat corpus error detection with feedback, few new errors were detected in this experiment.

6.4. Related Work

Compared to conventional probabilistic approaches for corpus error detection, although precise comparison is difficult, our approach achieves relatively high precision. By using a probabilistic approach, Murata et al. (2000) detected errors of words in a corpus with a precision of 70−80%, and Eskin (2000) detected errors with a precision of 69%, but our approach achieved more than 80%. The probabilistic methods cannot handle infrequent events nor compare events with similar probabilities, since the probabilities cannot be calculated or compared with

enough confidence. However, our method uses example-based machine learning method, and can handle such infrequent events.

SVMs are similar to boosting, and our approach uses the weights attached by SVMs in a similar manner to what Abney et al. (1999) studied. However, we introduced a post-processing step to extract inconsistent similar examples, and this improves the precision of detection and usability. Ma et al. (2001) studied corpus error detection by finding conflicting elements using min-max modular neural networks. Compared to their method, our method is useful in the point that the detected errors can be sorted by the attached weights so that human can check more suspicious elements first.

In the experiments, our method had high precisions but low recalls. The value will be controlled by tuning the features for SVMs as well as the threshold value θ, and detecting more errors in corpora remains as future work.

6.5. Conclusion

In this chapter, we proposed a method for corpus error detection using SVMs. This method extracts inconsistencies in corpora. We achieved precision of more than 80%, and the performance seems to be high enough for practical use in corpus refinement.

Chapter 7

Conclusion

In this dissertation, we studied statistical word segmentation and POS tagging for Chinese, Japanese, Korean and English. We proposed two methods for guessing POS tags of unknown words, and a method for word segmentation. We also proposed a method which remedy an inefficiency problem occurred in POS tagging with SVMs, and proposed a method for corpus error detection. These methods incorporate diverse features and have high accuracy compared to existing methods.

In Chapter 2, we presented methods for unknown word guessing and POS tagging using SVMs. Compared to a previous Markov model based method, the methods use a number of features effectively, and obtained high accuracy in experiments on English unknown word guessing and POS tagging.

In Chapter 3, we presented a method for guessing POS tags of unknown words using both local and global information. The method takes into consideration interactions between the POS tags of all the unknown words in a document by using Gibbs sampling. We conducted experiments on Chinese, English and Japanese unknown word guessing, and the method obtained higher accuracy compared to a method with only local information.

In Chapter 4, we presented a hybrid method for word segmentation, which utilizes both word-level and character-level information. The method combines two existing methods, the Markov model-based method and the character-based tagging method in order to obtain high accuracy for both known and unknown words. We conducted experiments on Chinese and Japanese word segmentation

and Korean morphological analysis, and the method obtained higher performance than most of previous methods.

In Chapter 5, we presented a revision learning method which combines a stochastic model and a binary classifier to achieve higher performance with low computational cost for large-scale multi-class classification problems. We applied it to word segmentation and POS tagging of Japanese and POS tagging of English, in which a large number of examples and classes are handled, and obtained comparable accuracy with small computational cost compared to the conventional one-versus-rest method.

In Chapter 6, we proposed a method for corpus error detection using SVMs. This method extracts inconsistencies in corpora by finding exceptional elements from the corpora. We applied the method to English and Japanese corpora, and obtained high precisions.

Although we attempted to use unlabeled data in Chapter 3, the results were not consistent. Semi-supervised learning is useful especially when a word segmenter or a POS tagger is applied to languages or domains in which there are few tagged corpora, and our future work will include semi-supervised learning for word segmentation and POS tagging.

In this dissertation, we concentrated on modeling given corpora as accurately as possible. However, the design of the corpora itself is a large issue. There are several POS tagsets, and also several word segmentation standards in Chinese and Japanese. Word segmentation and POS tagging are not the final goals in general, and finding optimal ways which maximize performance of the whole systems for individual applications, such as information retrieval and machine translation, is an open problem.

Bibliography

Steven Abney, Robert E. Schapire, and Yoram Singer. 1999. Boosting Applied to Tagging and PP Attachment. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 38–45.

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. 2000. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. In *Proceedings of 17th International Conference on Machine Learning*, pages 9–16.

Ethem Alpaydin and Cenk Kaynak. 1998. Cascading Classifiers. *Kybernetika*, 34(4):369–374.

Ethem Alpaydm. 1998. Techniques for Combining Multiple Learners. In *Proceedings of Engineering of Intelligent Systems '98 Conference*.

Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. 2003. An introduction to MCMC for Machine Learning. *Machine Learning*, 50:5–43.

Masayuki Asahara and Yuji Matsumoto. 2000. Extended Models and Tools for High-performance Part-of-Speech Tagger. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 21–27.

Masayuki Asahara and Yuji Matsumoto. 2002. Extended Statistical Models for Morphological Analysis. In *Transaction of Information Processing Society of Japan*, pages 685–695. (in Japanese).

Masayuki Asahara, Chooi Ling Goh, Xiaojie Wang, and Yuji Matsumoto. 2003. Combining Segmenter and Chunker for Chinese Word Segmentation. In *Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing*, pages 144–147.

Masayuki Asahara. 2003. *Corpus-based Japanese morphological analysis*. Nara Institute of Science and Technology, Doctor's Thesis.

Harald Baayen and Richard Sproat. 1996. Estimating Lexical Priors for Low-Frequency Morphologically Ambiguous Forms. *Computational Linguistics*, 22(2):155–166.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

Thorsten Brants. 2000. TnT — A Statistical Part-of-Speech Tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference and the 1st Meeting of the North American Chapter of the Association of Computational Linguistics*, pages 224–231.

Eric Brill and Jun Wu. 1998. Classifier Combination for Improved Lexical Disambiguation. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pages 191–195.

Eric Brill. 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics*, 21(4):543–565.

Eugene Charniak, Curtis Hendrickson, Neil Jacobson, and Mike Perkowitz. 1993. Equations for Part-of-Speech Tagging. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 784–789.

Stanley Chen and Ronald Rosenfeld. 1999. A Gaussian Prior for Smoothing Maximum Entropy Models. Technical Report CMUCS-99-108, Carnegie Mellon University.

Chao-jan Chen, Ming-hong Bai, and Keh-Jiann Chen. 1997. Category Guessing for Chinese Unknown Words. In *Proceedings of the 4th Natural Language Processing Pacific Rim Symposium 1997*, pages 35–40.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named Entity Recognition: A Maximum Entropy Approach Using Global Information. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 190–196.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 263–270.

Michael Collins. 2000. Discriminative Reranking for Natural Language Parsing. In *Proceedings of the 17th International Conference on Machine Learning*, pages 175–182.

Michael Collins. 2002. Ranking Algorithms for Named-Entity Extraction: Boosting and the Voted Perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 489–496.

Corinna Cortes and Vladimir Vapnik. 1995. Support Vector Networks. *Machine Learning*, 20:273–297.

Silviu Cucerzan and David Yarowsky. 2000. Language Independent, Minimally Supervised Induction of Lexical Probabilities. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 270–277.

Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A Practical Part-of-Speech Tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 133–140.

J. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The annuals of Mathematical Statistics*, 43(5):1470–1480.

Saso Džeroski, Tomaz Erjavec, and Jakub Zavrel. 2000. Morphosyntactic Tagging of Slovene: Evaluating Taggers and Tagsets. In *Proceedings of the Second*

International Conference on Language Resources and Evaluation, pages 1099–1104.

Eleazar Eskin. 2000. Detecting Errors within a Corpus using Anomaly Detection. In *Proceedings of the 6th Applied Natural Language Processing Conference and the 1st Meeting of the North American Chapter of the Association of Computational Linguistics*, pages 148–153.

Yair Even-Zohar and Dan Roth. 2001. A Sequential Model for Multi-Class Classification. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 10–19.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 363–370.

Chooi-Ling Goh, Masayuki Asahara, and Yuji Matsumoto. 2004. Chinese Word Segmentation by Classification of Characters. In *Proceedings of the 3rd ACL SIGHAN Workshop*, pages 57–64.

Chung-hye Han and Martha Palmer. 2004. A Morphological Tagger for Korean: Statistical Tagging Combined with Corpus-Based Morphological Rule Application. *Machine Translation*, 18(4):275–297.

Thorsten Joachims. 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142.

D. Klakow. 1998. Log-linear interpolation of language models. In *Proceedings of the 5-th International Conference on Spoken Language Processing*, pages 1695–1699.

Taku Kudoh and Yuji Matsumoto. 2000. Use of Support Vector Learning for Chunk Identification. In *Proceedings of the Fourth Conference on Computational Natural Language Learning*, pages 142–144.

Sadao Kurohashi and Makoto Nagao. 1998. *Japanese Morphological Analysis System JUMAN version 3.61*. Department of Informatics, Kyoto University. (in Japanese).

Sadao Kurohashi. 1998. *Japanese Dependency/Case Structure Analyzer KNP version 2.0b6*. Department of Informatics, Kyoto University. (in Japanese).

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289.

Gary Geunbae Lee, Jeongwon Cha, and Jong-Hyeok Lee. 2002. Syllable-Pattern-Based Unknown-Morpheme Segmentation and Estimation for Hybrid Part-of-Speech Tagging of Korean. *Computational Linguistics*, 28(1):53–70.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528.

Zhiyong Luo and Rou Song. 2004. An Integrated Method for Chinese Unknown Word Extraction. In *Proceedings of the 3rd ACL SIGHAN Workshop*, pages 148–154.

Qing Ma, Bao-Liang Lu, Masaki Murata, Michinori Ichikawa, and Hitoshi Isahara. 2001. On-Line Error Detection of Annotated Corpus Using Modular Neural Networks. In *Proceedings of International Conference on Artificial Neural Networks (ICANN 2001)*, pages 1185–1192.

David J. C. MacKay. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press.

Lluiís Màrquez, Horacio Rodríguez, Josep Carmona, and Josep Montolio. 1999. Improving POS Tagging Using Machine-Learning Techniques. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 53–62.

Jose. L. Marroquin. 1985. Optimal Bayesian Estimators for Image Segmentation and Surface Reconstruction. A.I. Memo 839, MIT.

Yuji Matsumoto and Masayuki Asahara. 2001. *IPADIC User's Manual version 2.2.4*. Nara Institute of Science and Technology. (in Japanese).

Yuji Matsumoto and Tatsuo Yamashita. 2000. Using Machine Learning Methods to Improve Quality of Tagged Corpora and Learning Models. In *Proceedings of the Second International Conference on Language Resource and Evaluation*, pages 11–16.

Yuji Matsumoto, Akira Kitauchi, Tatsuo Yamashita, Yoshitaka Hirano, Hiroshi Matsuda, Kazuma Takaoka, and Masayuki Asahara. 2001. *Morphological Analysis System ChaSen version 2.2.8 Manual*. Nara Institute of Science and Technology.

Beáta Megyesi. 2001. Comparing Data-Driven Learning Algorithms for PoS Tagging of Swedish. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*.

Andrei Mikheev. 1997. Automatic Rule Induction for Unknown-Word Guessing. *Computational Linguistics*, 23(3):405–423.

Shinsuke Mori and Makoto Nagao. 1996. Word Extraction from Corpora and Its Part-of-Speech Estimation Using Distributional Analysis. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 1119–1122.

Masaki Murata, Masao Utiyama, Kiyotaka Uchimoto, Qing Ma, and Hitoshi Isahara. 2000. Corpus Error Detection and Correction Using the Decision-List and Example-Based Methods. In *IPSJ SIGNotes Natural Language No.136*, pages 49–56. (in Japanese).

Masaki Nagata. 1994. A Stochastic Japanese Morphological Analyzer Using a Forward-DP Backward-A* N-Best Search Algorithm. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 201–207.

Masaaki Nagata. 1999a. *Japanese Language Processing Based on Stochastic Models*. Kyoto University, Doctoral Thesis. (in Japanese).

Masaki Nagata. 1999b. A Part of Speech Estimation Method for Japanese Unknown Words using a Statistical Model of Morphology and Context. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 277–284.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese Part-of-Speech Tagging: One-at-a-Time or All-at-Once? Word-Based or Character-Based? In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 277–284.

Giorgos S. Orphanos and Dimitris N. Christodoulakis. 1999. POS Disambiguation and Unknown Word Guessing with Decision Trees. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 134–141.

Lluiís Padró and Lluiís Màrquez. 1998. On the Evaluation and Comparison of Taggers: The Effect of Noise in Testing Corpora. In *Proceedings of the joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, pages 997–1002.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese Segmentation and New Word Detection using Conditional Random Fields. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 23–27.

John C. Platt. 1999. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classifiers*. MIT Press.

Lawrence R. Rabiner and Biing-Hwang Juang. 1993. *Fundamentals of Speech Recognition*. PTR Prentice-Hall, Inc.

Lance Ramshaw and Mitch Marcus. 1995. Text Chunking using Transformation-Based Learning. In *Proceedings of the 3rd Workshop on Very Large Corpora*, pages 88–94.

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.

Ronald Rosenfeld, Stanley F. Chen, and Xiaojin Zhu. 2001. Whole-Sentence Exponential Language Models: A Vehicle For Linguistic-Statistical Integration. *Computers Speech and Language*, 15(1):55–73.

Dan Roth and Dmitry Zelenko. 1998. Part of Speech Tagging Using a Network of Linear Separators. In *Proceedings of the joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, pages 1136–1142.

Sunita Sarawagi and William W. Cohen. 2005. Semi-Markov Conditional Random Fields for Information Extraction. In *Advances in Neural Information Processing Systems 17*, pages 1185–1192.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49.

Ingo Schröder. 2001. ICOPOST — Ingo's Collection Of POS Taggers. `http://nats-www.informatik.uni-hamburg.de/~ingo/icopost/`.

Satoshi Sekine, Ralph Grishman, and Hiroyuki Shinnou. 1998. A Decision Tree Method for Finding and Classifying Names in Japanese Texts. In *Proceedings of the 6th Workshop on Very Large Corpora*, pages 171–177.

Richard Sproat and Thomas Emerson. 2003. The First International Chinese Word Segmentation Bakeoff. In *Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing*, pages 133–143.

Richard Sproat, Chilin Shih, William Gale, and Nancy Chang. 1996. A Stochastic Finite-State Word-Segmentation Algorithm for Chinese. *Computational Linguistics*, 22(3):377–404.

Maosong Sun, Dayang Shen, and Changning Huang. 1997. CSeg&Tag1.0: A Practical Word Segmenter and POS Tagger for Chinese Texts. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 119–126.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting Semantic Orientations of Words using Spin Model. In *Proceedings of the 43rd*

Annual Meeting of the Association for Computational Linguistics, pages 133–140.

Scott M. Thede. 1998. Predicting Part-of-Speech Information about Unknown Words using Statistical Methods. In *Proceedings of the joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, pages 1505–1507.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing Text Chunks. In *Proceedings of 9th Conference of the European Chapter of the Association for Computational Linguistics*, pages 173–179.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 2001. The Unknown Word Problem: a Morphological Analysis of Japanese Using Maximum Entropy Aided by a Dictionary. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 91–99.

Hans van Halteren, Jakub Zavrel, and Walter Daelemans. 2001. Improving Accuracy in Wordclass Tagging through Combination of Machine Learning Systems. *Computational Linguistics*, 27(2):199–230.

Vladimir Vapnik. 1998. *Statistical Learning Theory*. Springer.

Ralph Weischedel, Marie Meteer, Richard Schwartz, Lance Ramshaw, and Jeoe Palmucci. 1993. Coping with Ambiguity and Unknown Words through Probabilistic Models. *Computational Linguistics*, 19(2):359–382.

Jason Weston and Chris Watkins. 1999. Support Vector Machines for Multi-Class Pattern Recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*.

Pak-Kwong Wong and Chorkin Chan. 1996. Chinese Word Segmentation based on Maximum Matching and Word Binding Force. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 200–203.

Nianwen Xue. 2003. Chinese Word Segmentation as Character Tagging. *International Journal of Computational Linguistics and Chinese*, 8(1):29–48.

Mikio Yamamoto and Masakazu Masuyama. 1997. Japanese Morphological Analysis using Markov Chain Probabilities of Extended Characters with Part-of-Speech and Word Segmentation Information. In *Proceedings of the 3rd Annual Meeting of the Association for Natural Language Processing*, pages 421–424. (in Japanese).

David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33th Annual Meeting of the Association for Computational Linguistics*, pages 189–196.

Tatsumi Yoshida, Kiyonori Ohtake, and Kazuhide Yamamoto. 2003. Performance Evaluation of Chinese Analyzers with Support Vector Machines. *Journal of Natural Language Processing*, 10(1):109–131. (in Japanese).

Hua-Ping Zhang, Qun Liu, Xue-Qi Cheng, Hao Zhang, and Hong-Kui Yu. 2003. Chinese Lexical Analysis Using Hierarchical Hidden Markov Model. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pages 63–70.

List of Publications

Journal Papers

[1] Tetsuji Nakagawa, Taku Kudo and Yuji Matsumoto: Morphological Analysis Using Support Vector Machines and Proposal of Revision Learning, *IPSJ Journal*, Vol.44, No.5, pp.1354–1367, May 2003. (in Japanese).

[2] Tetsuji Nakagawa and Yuji Matsumoto: Chinese and Japanese Word Segmentation Using Word-level and Character-level Information, *IPSJ Journal*, Vol.46, No.11, pp.2714–2727, Nov 2005. (in Japanese).

Conference Papers

[1] Tetsuji Nakagawa, Taku Kudo and Yuji Matsumoto: Unknown Word Guessing and Part-of-Speech Tagging Using Support Vector Machines, In *Proceedings of the 6th Natural Language Processing Pacific Rim Symposium (NLPRS2001)*, Tokyo, Japan, pp.325–331, November 2001.

[2] Tetsuji Nakagawa, Taku Kudo and Yuji Matsumoto: Revision Learning and its Application to Part-of-Speech Tagging, In *Proceedings of the 40th Annual Meeting of Association for Computational Linguistics (ACL-02)*, Philadelphia, USA, pp.497–504, July 2002.

[3] Tetsuji Nakagawa and Yuji Matsumoto: Detecting Errors in Corpora Using Support Vector Machines, In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, Taiwan, pp.709–715, August 2002.

[4] Mihoko Kitamura, Tetsuji Nakagawa, Seika Kim and Toshiki Murata: Development of Chinese-Japanese MT System Based on Language-Independent Translation Engine, In *Proceedings of Asian Symposium on Natural Language Processing to Overcome Language Barriers*, Hainan Island, China, pp.39–45, March 2004.

[5] Tetsuji Nakagawa: Chinese and Japanese Word Segmentation Using Word-Level and Character-Level Information, In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland, pp.466–472, August 2004.

Other Publications

[1] Tetsuji Nakagawa, Taku Kudo and Yuji Matsumoto: Unknown Word Guessing with Support Vector Machines, *IPSJ SIG Notes*, 2001-NL-141, pp.77–82, January 2001. (in Japanese).

[2] Taku Kudo, Hiroyasu Yamada, Tetsuji Nakagawa and Yuji Matsumoto: Automatic Feature Selection for Chunking with SVMs, In *Proceedings of the 7th Annual Meeting of the Association for Natural Language Processing (NLP-2001)*, pp.257–260, March 2001. (in Japanese).

[3] Yuji Matsumoto, Taku Kudo, Hiroya Takamura, Hiroyasu Yamada and Tetsuji Nakagawa: System Ensemble Methods in Natural Language Processing, In *Proceedings of 2001 Workshop on Information-Based Induction Sciences (IBIS 2001)*, pp.19–24, July 2001. (in Japanese).

[4] Tetsuji Nakagawa, Taku Kudo and Yuji Matsumoto: *Revision Learning Applied to Morphological Analysis*, *IPSJ SIG Notes*, 2001-NL-146, pp.1–8, November 2001. (in Japanese).

[5] Tetsuji Nakagawa and Yuji Matsumoto: Corpus Error Detection Using Support Vector Machines, In *Proceedings of the 8th Annual Meeting of the Association for Natural Language Processing (NLP-2002)*, pp.563–566, March 2002. (in Japanese).

[6] Hideki Kawai, Hiroyasu Yamada, Tetsuji Nakagawa, Hiroshi Sasaki, Susumu Akamine, Yuji Matsumoto and Toshikazu Fukushima: Automatic Classification of Location Information Web Pages using Support Vector Machines, In *Proceedings of the 1st Forum on Information Technology (FIT 2002)*, D-2, Vol.2, pp.3–4, September 2002. (in Japanese).

[7] Mihoko Kitamura, Toshiki Murata, Tatsuya Sukehiro, Sayori Shimohata, Miki Sasaki, Toshihiko Matsunaga and Tetsuji Nakagawa: Basic Technology and Development of Community Type Machine Translation Sites Yakushite-Net, In *Proceedings of the 65th National Convention of IPSJ*, Vol.5, pp.319–322, March 2003. (in Japanese).

[8] Miki Sasaki, Mihoko Kitamura, Sayori Shimohata and Tetsuji Nakagawa: Core Word based Category Judging for Dictionary Construction, In *Proceedings of the 2nd Forum on Information Technology (FIT 2003)*, September 2003. (in Japanese).

[9] Tetsuji Nakagawa and Yuji Matsumoto: Chinese and Japanese Word Segmentation Using Word-Level and Character-Level Information, *IPSJ SIG Technical Reports*, 2004-NL-162, pp.197–204, July 2004. (in Japanese).

[10] Tetsuji Nakagawa and Mihoko Kitamura: NTCIR-4 CLIR Experiments at Oki, In *Proceedings of the 4th NTCIR Workshop*, March 2005.

[11] Tetsuji Nakagawa: NTCIR-5 CLIR Experiments at Oki, In *Proceedings of the 5th NTCIR Workshop*, pp.104–109, December 2005.

[12] Tetsuji Nakagawa and Yuji Matsumoto: Guessing Parts-of-Speech of Unknown Words Using Global Information, *IPSJ SIG Technical Reports*, 2006-NL-172, March 2006. (in Japanese).