Chunking with Support Vector Machines

Conference Paper · June 2001

DOI: 10.3115/1073336.1073361 · Source: DBLP

Yuji Matsumoto
Nara Institute of Science and Technology
362 PUBLICATIONS **6,496** CITATIONS

Chunking with Support Vector Machines

Taku Kudo and **Yuji Matsumoto**
Graduate School of Information Science,
Nara Institute of Science and Technology
{taku-ku,matsu}@is.aist-nara.ac.jp

Abstract

We apply Support Vector Machines (SVMs) to identify English base phrases (chunks). SVMs are known to achieve high generalization performance even with input data of high dimensional feature spaces. Furthermore, by the Kernel principle, SVMs can carry out training with smaller computational overhead independent of their dimensionality. We apply weighted voting of 8 SVMs-based systems trained with distinct chunk representations. Experimental results show that our approach achieves higher accuracy than previous approaches.

1 Introduction

Chunking is recognized as series of processes — first identifying proper *chunks* from a sequence of *tokens* (such as words), and second classifying these chunks into some grammatical classes. Various NLP tasks can be seen as a chunking task. Examples include English base noun phrase identification (base NP chunking), English base phrase identification (chunking), Japanese chunk (*bunsetsu*) identification and named entity extraction. Tokenization and part-of-speech tagging can also be regarded as a chunking task, if we assume each character as a *token*.

Machine learning techniques are often applied to chunking, since the task is formulated as estimating an identifying function from the information (features) available in the surrounding context. Various machine learning approaches have been proposed for chunking (Ramshaw and Marcus, 1995; Tjong Kim Sang, 2000a; Tjong Kim Sang et al., 2000; Tjong Kim Sang, 2000b; Sassano and Utsuro, 2000; van Halteren, 2000).

Conventional machine learning techniques, such as Hidden Markov Model (HMM) and Maximum Entropy Model (ME), normally require a careful feature selection in order to achieve high accuracy. They do not provide a method for automatic selection of given feature sets. Usually, heuristics are used for selecting effective features and their combinations.

New statistical learning techniques such as Support Vector Machines (SVMs) (Cortes and Vapnik, 1995; Vapnik, 1998) and Boosting(Freund and Schapire, 1996) have been proposed. These techniques take a strategy that maximizes the margin between critical samples and the separating hyperplane. In particular, SVMs achieve high generalization even with training data of a very high dimension. Furthermore, by introducing the Kernel function, SVMs handle non-linear feature spaces, and carry out the training considering combinations of more than one feature.

In the field of natural language processing, SVMs are applied to text categorization and syntactic dependency structure analysis, and are reported to have achieved higher accuracy than previous approaches.(Joachims, 1998; Taira and Haruno, 1999; Kudo and Matsumoto, 2000a).

In this paper, we apply Support Vector Machines to the chunking task. In addition, in order to achieve higher accuracy, we apply weighted voting of 8 SVM-based systems which are trained using distinct chunk representations. For the weighted voting systems, we introduce a new type of weighting strategy which are derived from the theoretical basis of the SVMs.

2 Support Vector Machines

2.1 Optimal Hyperplane

Let us define the training samples each of which belongs either to positive or negative class as: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$ $(\mathbf{x}_i \in \mathbf{R}^n,\ y_i \in \{+1, -1\})$. $\mathbf{x}_i$ is a feature vector of the i-th sample represented by an n dimensional vector. y_i is the class (positive($+1$) or negative(-1) class) label of the i-th sample. l is the number of the given training sam-

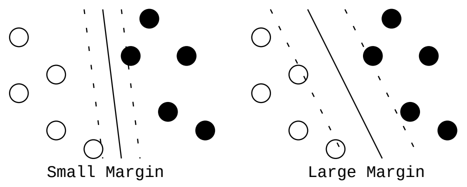

Figure 1: Two possible separating hyperplanes

ples. In the basic SVMs framework, we try to separate the positive and negative samples by a hyperplane expressed as: $(\mathbf{w} \cdot \mathbf{x}) + b = 0 \;\; (\mathbf{w} \in \mathbf{R}^n, b \in \mathbf{R})$. SVMs find an "optimal" hyperplane (i.e. an optimal parameter set for $\mathbf{w}, b$) which separates the training data into two classes. What does "optimal" mean? In order to define it, we need to consider the **margin** between two classes. Figure 1 illustrates this idea. Solid lines show two possible hyperplanes, each of which correctly separates the training data into two classes. Two dashed lines parallel to the separating hyperplane indicate the boundaries in which one can move the separating hyperplane without any misclassification. We call the distance between those parallel dashed lines as **margin**. SVMs find the separating hyperplane which maximizes its margin. Precisely, two dashed lines and margin (M) can be expressed as: $\mathbf{w} \cdot \mathbf{x} + b = \pm 1, \quad M = 2/\|\mathbf{w}\|$.

To maximize this margin, we should minimize $\|\mathbf{w}\|$. In other words, this problem becomes equivalent to solving the following optimization problem:

$$\begin{array}{ll} \text{Minimize :} & L(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{Subject to :} & y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq 1 \; (i = 1, \ldots, l). \end{array}$$

The training samples which lie on either of two dashed lines are called support vectors. It is known that only the support vectors in given training data matter. This implies that we can obtain the same decision function even if we remove all training samples except for the extracted support vectors.

In practice, even in the case where we cannot separate training data linearly because of some noise in the training data, etc, we can build the separating linear hyperplane by allowing some misclassifications. Though we omit the details here, we can build an optimal hyperplane by introducing a soft margin parameter C, which trades off between the training error and the magnitude of the margin.

Furthermore, SVMs have a potential to carry out the non-linear classification. Though we leave the details to (Vapnik, 1998), the optimization problem can be rewritten into a dual form, where all feature vectors appear in their dot products. By simply substituting every dot product of $\mathbf{x}_i$ and $\mathbf{x}_j$ in dual form with a certain Kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$, SVMs can handle non-linear hypotheses. Among many kinds of Kernel functions available, we will focus on the d-th polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$. Use of d-th polynomial kernel functions allows us to build an optimal separating hyperplane which takes into account all combinations of features up to d.

2.2 Generalization Ability of SVMs

Statistical Learning Theory(Vapnik, 1998) states that training error (*empirical risk*) E_t and test error (*risk*) E_g hold the following theorem.

Theorem 1 (Vapnik) *If $h(h < l)$ is the VC dimension of the class functions implemented by some machine learning algorithms, then for all functions of that class, with a probability of at least $1 - \eta$, the risk is bounded by*

$$E_g \leq E_t + \sqrt{\frac{h(\ln \frac{2l}{h} + 1) - \ln \frac{\eta}{4}}{l}}, \tag{1}$$

where h is a non-negative integer called the Vapnik Chervonenkis (VC) dimension, and is a measure of the complexity of the given decision function. The r.h.s. term of (1) is called **VC bound**. In order to minimize the *risk*, we have to minimize the *empirical risk* as well as VC dimension. It is known that the following theorem holds for VC dimension h and margin M(Vapnik, 1998).

Theorem 2 (Vapnik) *Suppose n as the dimension of given training samples M as the margin, and D as the smallest diameter which encloses all training sample, then VC dimension h of the SVMs are bounded by*

$$h \leq \min(D^2/M^2, n) + 1. \tag{2}$$

In order to minimize the VC dimension h, we have to maximize the margin M, which is exactly the strategy that SVMs take.

Vapnik gives an alternative bound for the *risk*.

Theorem 3 (Vapnik) *Suppose E_l is an error rate estimated by Leave-One-Out procedure, E_l is bounded as*

$$E_l \leq \frac{Number\ of\ Support\ Vectors}{Number\ of\ Training\ Samples}. \tag{3}$$

Leave-One-Out procedure is a simple method to examine the *risk* of the decision function — first by removing a single sample from the training data, we construct the decision function on the basis of the remaining training data, and then test the removed sample. In this fashion, we test all l samples of the training data using l different decision functions. (3) is a natural consequence bearing in mind that support vectors are the only factors contributing to the final decision function. Namely, when the every removed support vector becomes error in *Leave-One-Out* procedure, E_l becomes the r.h.s. term of (3). In practice, it is known that this bound is less predictive than the VC bound.

3 Chunking

3.1 Chunk representation

There are mainly two types of representations for proper chunks. One is **Inside/Outside** representation, and the other is **Start/End** representation.

1. **Inside/Outside**
   This representation was first introduced in (Ramshaw and Marcus, 1995), and has been applied for base NP chunking. This method uses the following set of three tags for representing proper chunks.

I	Current token is inside of a chunk.
O	Current token is outside of any chunk.
B	Current token is the beginning of a chunk which immediately follows another chunk.

   Tjong Kim Sang calls this method as IOB1 representation, and introduces three alternative versions — IOB2,IOE1 and IOE2 (Tjong Kim Sang and Veenstra, 1999).

IOB2	A B tag is given for every token which exists at the beginning of a chunk. Other tokens are the same as IOB1.
IOE1	An E tag is used to mark the last token of a chunk immediately preceding another chunk.
IOE2	An E tag is given for every token which exists at the end of a chunk.

2. **Start/End**
   This method has been used for the Japanese named entity extraction task, and requires the following five tags for representing proper chunks(Uchimoto et al., 2000) [1].

B	Current token is the start of a chunk consisting of more than one token.
E	Current token is the end of a chunk consisting of more than one token.
I	Current token is a middle of a chunk consisting of more than two tokens.
S	Current token is a chunk consisting of only one token.
O	Current token is outside of any chunk.

	IOB1	IOB2	IOE1	IOE2	Start/End
In	O	O	O	O	O
early	I	B	I	I	B
trading	I	I	I	E	E
in	O	O	O	O	O
busy	I	B	I	I	B
Hong	I	I	I	I	I
Kong	I	I	E	E	E
Monday	B	B	I	E	S
,	O	O	O	O	O
gold	I	B	I	E	S
was	O	O	O	O	O

Table 1: Example for each chunk representation

Examples of these five representations are shown in Table 1.

If we have to identify the grammatical class of each chunk, we represent them by a pair of an I/O/B/E/S label and a class label. For example, in IOB2 representation, B-VP label is given to a token which represents the beginning of a verb base phrase (VP).

3.2 Chunking with SVMs

Basically, SVMs are binary classifiers, thus we must extend SVMs to multi-class classifiers in order to classify three (B,I,O) or more (B,I,O,E,S) classes. There are two popular methods to extend a binary classification task to that of K classes. One is *one class vs. all others*. The idea is to build K classifiers so as to separate one class from all others. The other is *pairwise* classification. The idea is to build $K \times (K-1)/2$ classifiers considering all pairs of classes, and final decision is given by their weighted voting. There are a number of other methods to extend SVMs to multiclass classifiers. For example, Dietterich and Bakiri(Dietterich and Bakiri, 1995) and Allwein(Allwein et al., 2000) introduce a unifying framework for solving the multiclass problem

[1] Originally, Uchimoto uses C/E/U/O/S representation. However we rename them as B/I/O/E/S for our purpose, since we want to keep consistency with Inside/Start (B/I/O) representation.

by reducing them into binary models. However, we employ the simple *pairwise* classifiers because of the following reasons:

(1) In general, SVMs require $O(n^2) \sim O(n^3)$ training cost (where n is the size of training data). Thus, if the size of training data for individual binary classifiers is small, we can significantly reduce the training cost. Although *pairwise* classifiers tend to build a larger number of binary classifiers, the training cost required for *pairwise* method is much more tractable compared to the *one vs. all others*.

(2) Some experiments (Kreßel, 1999) report that a combination of *pairwise* classifiers performs better than the *one vs. all others*.

For the feature sets for actual training and classification of SVMs, we use all the information available in the surrounding context, such as the words, their part-of-speech tags as well as the chunk labels. More precisely, we give the following features to identify the chunk label c_i for the i-th word:

		$\rightarrow$	Direction	$\rightarrow$	
Word:	w_{i-2}	w_{i-1}	w_i	w_{i+1}	w_{i+1}
POS:	t_{i-2}	t_{i-1}	t_i	t_{i+1}	t_{i+1}
Chunk:	c_{i-2}	c_{i-1}	$\boxed{c_i}$		

Here, w_i is the word appearing at i-th position, t_i is the POS tag of w_i, and c_i is the (extended) chunk label for i-th word. In addition, we can reverse the parsing direction (from right to left) by using two chunk tags which appear to the r.h.s. of the current token (c_{i+1}, c_{i+2}). In this paper, we call the method which parses from left to right as **forward parsing**, and the method which parses from right to left as **backward parsing**.

Since the preceding chunk labels (c_{i-1}, c_{i-2} for forward parsing , c_{i+1}, c_{i+2} for backward parsing) are not given in the test data, they are decided dynamically during the tagging of chunk labels. The technique can be regarded as a sort of Dynamic Programming (DP) matching, in which the best answer is searched by maximizing the total certainty score for the combination of tags. In using DP matching, we limit a number of ambiguities by applying beam search with width N. In CoNLL 2000 shared task, the number of votes for the class obtained through the pairwise voting is used as the certain score for beam search with width 5 (Kudo and Matsumoto, 2000a). In this paper, however, we apply deterministic method instead of applying beam search with keeping some ambiguities. The reason we apply deterministic method is that our further experiments and investigation for the selection of beam width shows that larger beam width dose not always give a significant improvement in the accuracy. Given our experiments, we conclude that satisfying accuracies can be obtained even with the deterministic parsing. Another reason for selecting the simpler setting is that the major purpose of this paper is to compare weighted voting schemes and to show an effective weighting method with the help of empirical risk estimation frameworks.

3.3 Weighted Voting

Tjong Kim Sang et al. report that they achieve higher accuracy by applying weighted voting of systems which are trained using distinct chunk representations and different machine learning algorithms, such as MBL, ME and IGTree(Tjong Kim Sang, 2000a; Tjong Kim Sang et al., 2000). It is well-known that weighted voting scheme has a potential to maximize the margin between critical samples and the separating hyperplane, and produces a decision function with high generalization performance(Schapire et al., 1997). The boosting technique is a type of weighted voting scheme, and has been applied to many NLP problems such as parsing, part-of-speech tagging and text categorization.

In our experiments, in order to obtain higher accuracy, we also apply weighted voting of 8 SVM-based systems which are trained using distinct chunk representations. Before applying weighted voting method, first we need to decide the weights to be given to individual systems. We can obtain the best weights if we could obtain the accuracy for the “true” test data. However, it is impossible to estimate them. In boosting technique, the voting weights are given by the accuracy of the training data during the iteration of changing the frequency (distribution) of training data. However, we cannot use the accuracy of the training data for voting weights, since SVMs do not depend on the frequency (distribution) of training data, and can separate the training data without any mis-classification by selecting the appropriate kernel function and the soft margin parameter. In this paper, we introduce the following four weighting methods in our experiments:

1. Uniform weights
   We give the same voting weight to all systems. This method is taken as the baseline for other weighting methods.

2. Cross validation
Dividing training data into N portions, we employ the training by using $N-1$ portions, and then evaluate the remaining portion. In this fashion, we will have N individual accuracy. Final voting weights are given by the average of these N accuracies.

3. VC-bound
By applying (1) and (2), we estimate the lower bound of accuracy for each system, and use the accuracy as a voting weight. The voting weight is calculated as: $w = 1 - VCbound$. The value of D, which represents the smallest diameter enclosing all of the training data, is approximated by the maximum distance from the origin.

4. Leave-One-Out bound
By using (3), we estimate the lower bound of the accuracy of a system. The voting weight is calculated as: $w = 1 - E_l$.

The procedure of our experiments is summarized as follows:

1. We convert the training data into 4 representations (IOB1/IOB2/IOE1/IOE2).

2. We consider two parsing directions (Forward/Backward) for each representation, i.e. $4 \times 2 = 8$ systems for a single training data set. Then, we employ SVMs training using these independent chunk representations.

3. After training, we examine the VC bound and Leave-One-Out bound for each of 8 systems. As for cross validation, we employ the steps 1 and 2 for each divided training data, and obtain the weights.

4. We test these 8 systems with a separated test data set. Before employing weighted voting, we have to convert them into a uniform representation, since the tag sets used in individual 8 systems are different. For this purpose, we re-convert each of the estimated results into 4 representations (IOB1/IOB2/IOE2/IOE1).

5. We employ weighted voting of 8 systems with respect to the converted 4 uniform representations and the 4 voting schemes respectively. Finally, we have 4 (types of uniform representations) $\times$ 4 (types of weights) $= 16$ results for our experiments.

Although we can use models with IOBES-F or IOBES-B representations for the committees for the weighted voting, we do not use them in our voting experiments. The reason is that the number of classes are different (3 vs. 5) and the estimated VC and LOO bound cannot straightforwardly be compared with other models that have three classes (IOB1/IOB2/IOE1/IOE2) under the same condition. We conduct experiments with IOBES-F and IOBES-B representations only to investigate how far the difference of various chunk representations would affect the actual chunking accuracies.

4 Experiments

4.1 Experiment Setting

We use the following three annotated corpora for our experiments.

- Base NP standard data set (**baseNP-S**)
This data set was first introduced by (Ramshaw and Marcus, 1995), and taken as the standard data set for baseNP identification task[2]. This data set consists of four sections (15-18) of the Wall Street Journal (WSJ) part of the Penn Treebank for the training data, and one section (20) for the test data. The data has part-of-speech (POS) tags annotated by the Brill tagger(Brill, 1995).

- Base NP large data set (**baseNP-L**)
This data set consists of 20 sections (02-21) of the WSJ part of the Penn Treebank for the training data, and one section (00) for the test data. POS tags in this data sets are also annotated by the Brill tagger. We omit the experiments IOB1 and IOE1 representations for this training data since the data size is too large for our current SVMs learning program. In case of IOB1 and IOE1, the size of training data for one classifier which estimates the class I and O becomes much larger compared with IOB2 and IOE2 models. In addition, we also omit to estimate the voting weights using cross validation method due to a large amount of training cost.

- Chunking data set (**chunking**)
This data set was used for CoNLL-2000 shared task(Tjong Kim Sang and Buchholz, 2000). In this data set, the total of 10 base phrase classes (NP,VP,PP,ADJP,ADVP,CONJP,

[2]ftp://ftp.cis.upenn.edu/pub/chunker/

INITJ,LST,PTR,SBAR) are annotated. This data set consists of 4 sections (15-18) of the WSJ part of the Penn Treebank for the training data, and one section (20) for the test data [3].

All the experiments are carried out with our software package *TinySVM*[4], which is designed and optimized to handle large sparse feature vectors and large number of training samples. This package can estimate the VC bound and Leave-One-Out bound automatically. For the kernel function, we use the 2-nd polynomial function and set the soft margin parameter C to be 1.

In the baseNP identification task, the performance of the systems is usually measured with three rates: precision, recall and $F_{\beta=1}(= 2 \cdot precision \cdot recall/(precision + recall))$. In this paper, we refer to $F_{\beta=1}$ as *accuracy*.

4.2 Results of Experiments

Table 2 shows results of our SVMs based chunking with individual chunk representations. This table also lists the voting weights estimated by different approaches (B:Cross Validation, C:VC-bound, D:Leave-one-out). We also show the results of Start/End representation in Table 2.

Table 3 shows the results of the weighted voting of four different voting methods: A: Uniform, B: Cross Validation ($N = 5$), C: VC bound, D: Leave-One-Out Bound.

Table 4 shows the precision, recall and $F_{\beta=1}$ of the best result for each data set.

4.3 Accuracy vs Chunk Representation

We obtain the best accuracy when we apply IOE2-B representation for baseNP-S and chunking data set. In fact, we cannot find a significant difference in the performance between Inside/Outside(IOB1/IOB2/IOE1/IOE2) and Start/End(IOBES) representations.

Sassano and Utsuro evaluate how the difference of the chunk representation would affect the performance of the systems based on different machine learning algorithms(Sassano and Utsuro, 2000). They report that Decision List system performs better with Start/End representation than with Inside/Outside, since Decision List considers the specific combination of features. As for Maximum Entropy, they report that it performs better with Inside/Outside representation than with Start/End,

Training Condition		Acc.	Estimated Weights		
data	rep.	$F_{\beta=1}$	B	C	D
baseNP-S	IOB1-F	93.76	.9394	.4310	.9193
	IOB1-B	93.93	.9422	.4351	.9184
	IOB2-F	93.84	.9410	.4415	.9172
	IOB2-B	93.70	.9407	.4300	.9166
	IOE1-F	93.73	.9386	.4274	.9183
	IOE1-B	93.98	.9425	.4400	**.9217**
	IOE2-F	93.98	.9409	.4350	.9180
	IOE2-B	**94.11**	**.9426**	**.4510**	.9193
baseNP-L	IOB2-F	**95.34**	-	.4500	.9497
	IOB2-B	95.28	-	.4362	.9487
	IOE2-F	95.32	-	.4467	.9496
	IOE2-B	95.29	-	**.4556**	**.9503**
chunking	IOB1-F	93.48	.9342	.6585	.9605
	IOB1-B	93.74	.9346	.6614	.9596
	IOB2-F	93.46	.9341	.6809	.9586
	IOB2-B	93.47	.9355	.6722	.9594
	IOE1-F	93.45	.9335	.6533	.9589
	IOE1-B	93.72	.9358	.6669	**.9611**
	IOE2-F	93.45	.9341	.6740	.9606
	IOE2-B	**93.85**	**.9361**	**.6913**	.9597
baseNP-S	IOBES-F	93.96			
	IOBES-B	93.58			
chunking	IOBES-F	93.31			
	IOBES-B	93.41			

B:Cross Validation, C:VC bound, D:LOO bound

Table 2: Accuracy of individual representations

Training Condition		Accuracy $F_{\beta=1}$			
data	rep.	A	B	C	D
baseNP-S	IOB1	94.14	94.20	94.20	94.16
	IOB2	94.16	**94.22**	**94.22**	94.18
	IOE1	94.14	94.19	94.19	94.16
	IOE2	94.16	94.20	94.21	94.17
baseNP-L	IOB2	**95.77**	-	95.66	95.66
	IOE2	**95.77**	-	95.66	95.66
chunking	IOB1	93.77	93.87	93.89	93.87
	IOB2	93.72	93.87	93.90	93.88
	IOE1	93.76	93.86	93.88	93.86
	IOE2	93.77	93.89	**93.91**	93.85

A:Uniform Weights, B:Cross Validation
C:VC bound, D:LOO bound

Table 3: Results of weighted voting

data set	precision	recall	$F_{\beta=1}$
baseNP-S	94.15%	94.29%	94.22
baseNP-L	95.62%	95.93%	95.77
chunking	93.89%	93.92%	93.91

Table 4: Best results for each data set

[3] http://lcg-www.uia.ac.be/conll2000/chunking/

[4] http://cl.aist-nara.ac.jp/ taku-ku/software/TinySVM/

since Maximum Entropy model regards all features as independent and tries to catch the more general feature sets.

We believe that SVMs perform well regardless of the chunk representation, since SVMs have a high generalization performance and a potential to select the optimal features for the given task.

4.4 Effects of Weighted Voting

By applying weighted voting, we achieve higher accuracy than any of single representation system regardless of the voting weights. Furthermore, we achieve higher accuracy by applying Cross validation and VC-bound and Leave-One-Out methods than the baseline method.

By using VC bound for each weight, we achieve nearly the same accuracy as that of Cross validation. This result suggests that the VC bound has a potential to predict the error rate for the "true" test data accurately. Focusing on the relationship between the accuracy of the test data and the estimated weights, we find that VC bound can predict the accuracy for the test data precisely. Even if we have no room for applying the voting schemes because of some real-world constraints (limited computation and memory capacity), the use of VC bound may allow to obtain the best accuracy. On the other hand, we find that the prediction ability of Leave-One-Out is worse than that of VC bound.

Cross validation is the standard method to estimate the voting weights for different systems. However, Cross validation requires a larger amount of computational overhead as the training data is divided and is repeatedly used to obtain the voting weights. We believe that VC bound is more effective than Cross validation, since it can obtain the comparable results to Cross validation without increasing computational overhead.

4.5 Comparison with Related Works

Tjong Kim Sang et al. report that they achieve accuracy of 93.86 for baseNP-S data set, and 94.90 for baseNP-L data set. They apply weighted voting of the systems which are trained using distinct chunk representations and different machine learning algorithms such as MBL, ME and IGTree(Tjong Kim Sang, 2000a; Tjong Kim Sang et al., 2000).

Our experiments achieve the accuracy of 93.76 - 94.11 for baseNP-S, and 95.29 - 95.34 for baseNP-L even with a single chunk representation. In addition, by applying the weighted voting framework, we achieve accuracy of **94.22** for baseNP-S, and **95.77** for baseNP-L data set. As far as accuracies are concerned, our model outperforms Tjong Kim Sang's model.

In the CoNLL-2000 shared task, we achieved the accuracy of 93.48 using IOB2-F representation (Kudo and Matsumoto, 2000b) [5]. By combining weighted voting schemes, we achieve accuracy of **93.91**. In addition, our method also outperforms other methods based on the weighted voting(van Halteren, 2000; Tjong Kim Sang, 2000b).

4.6 Future Work

- Applying to other chunking tasks
  Our chunking method can be equally applicable to other chunking task, such as English POS tagging, Japanese chunk(*bunsetsu*) identification and named entity extraction. For future, we will apply our method to those chunking tasks and examine the performance of the method.

- Incorporating variable context length model
  In our experiments, we simply use the so-called fixed context length model. We believe that we can achieve higher accuracy by selecting appropriate context length which is actually needed for identifying individual chunk tags. Sassano and Utsuro(Sassano and Utsuro, 2000) introduce a variable context length model for Japanese named entity identification task and perform better results. We will incorporate the variable context length model into our system.

- Considering more predictable bound
  In our experiments, we introduce new types of voting methods which stem from the theorems of SVMs — VC bound and Leave-One-Out bound. On the other hand, Chapelle and Vapnik introduce an alternative and more predictable bound for the *risk* and report their proposed bound is quite useful for selecting the kernel function and soft margin parameter(Chapelle and Vapnik, 2000). We believe that we can obtain higher accuracy using this more predictable bound for the voting weights in our experiments.

[5] In our experiments, the accuracy of 93.46 is obtained with IOB2-F representation, which was the exactly the same representation we applied for CoNLL 2000 shared task. This slight difference of accuracy arises from the following two reason : (1) The difference of beam width for parsing (N=1 vs. N=5), (2) The difference of applied SVMs package (*TinySVM* vs. SVM^{light}.

5 Summary

In this paper, we introduce a uniform framework for chunking task based on Support Vector Machines (SVMs). Experimental results on WSJ corpus show that our method outperforms other conventional machine learning frameworks such MBL and Maximum Entropy Models. The results are due to the good characteristics of generalization and non-overfitting of SVMs even with a high dimensional vector space. In addition, we achieve higher accuracy by applying weighted voting of 8-SVM based systems which are trained using distinct chunk representations.

References

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. 2000. Reducing multiclass to binary: A unifying approach for margin classifiers. In *International Conf. on Machine Learning (ICML)*, pages 9–16.

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

Oliver Chapelle and Vladimir Vapnik. 2000. Model selection for support vector machines. In *Advances in Neural Information Processing Systems 12*. Cambridge, Mass: MIT Press.

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

T. G. Dietterich and G. Bakiri. 1995. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286.

Yoav Freund and Robert E. Schapire. 1996. Experiments with a new boosting algorithm. In *International Conference on Machine Learning (ICML)*, pages 148–146.

Thorsten Joachims. 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *European Conference on Machine Learning (ECML)*.

Ulrich H.-G Kreßel. 1999. Pairwise Classification and Support Vector Machines. In *Advances in Kernel Mathods*. MIT Press.

Taku Kudo and Yuji Matsumoto. 2000a. Japanese Dependency Structure Analysis Based on Support Vector Machines. In *Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 18–25.

Taku Kudo and Yuji Matsumoto. 2000b. Use of Support Vector Learning for Chunk Identification. In *Proceedings of the 4th Conference on CoNLL-2000 and LLL-2000*, pages 142–144.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd Workshop on Very Large Corpora*, pages 88–94.

Manabu Sassano and Takehito Utsuro. 2000. Named Entity Chunking Techniques in Supervised Learning for Japanese Named Entity Recognition. In *Proceedings of COLING 2000*, pages 705–711.

Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. 1997. Boosting the margin: a new explanation for the effectiveness of voting methods. In *International Conference on Machine Learning (ICML)*, pages 322–330.

Hirotoshi Taira and Masahiko Haruno. 1999. Feature Selection in SVM Text Categorization. In *AAAI-99*.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of EACL'99*, pages 173–179.

Erik F. Tjong Kim Sang, Walter Daelemans, Hervé Déjean, Rob Koeling, Yuval Krymolowski, Vasin Punyakanok, and Dan Roth. 2000. Applying system combination to base noun phrase identification. In *Proceedings of COLING 2000*, pages 857–863.

Erik F. Tjong Kim Sang. 2000a. Noun phrase recognition by system combination. In *Proceedings of ANLP-NAACL 2000*, pages 50–55.

Erik F. Tjong Kim Sang. 2000b. Text Chunking by System Combination. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 151–153.

Kiyotaka Uchimoto, Qing Ma, Masaki Murata, Hiromi Ozaku, and Hitoshi Isahara. 2000. Named Entity Extraction Based on A Maximum Entropy Model and Transformation Rules. In *Processing of the ACL 2000*.

Hans van Halteren. 2000. Chunking with WPDV Models. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 154–156.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.