NAIST-IS-MT9951036

Master's Thesis

Japanese Dependency Structure Analysis Based on Support Vector Machines

Taku Kudo

February 9, 2000

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

Master's Thesis
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
MASTER of ENGINEERING

Taku Kudo

Thesis Committee: Yuji Matsumoto, Professor
Kiyohiro Shikano, Professor
Yasuharu Den, Associate Professor

Japanese Dependency Structure Analysis Based on Support Vector Machines*

Taku Kudo

Abstract

This thesis presents a method of Japanese dependency structure analysis based on Support Vector Machines (SVMs). Conventional parsing techniques based on Machine Learning framework, such as Decision Trees and Maximum Entropy Models, have difficulty in selecting useful features as well as finding appropriate combination of selected features. On the other hand, it is well-known that SVMs achieve high generalization performance even with input data of very high dimensional feature space. Furthermore, by introducing the Kernel principle, SVMs can carry out the training in high-dimensional spaces with a smaller computational cost independent of their dimensionality. We apply SVMs to Japanese dependency structure identification problem. In our experiments, we consider two methods for applying SVMs to dependency structure analysis. One is probabilistic model wildly used in statistical dependency structure analysis. The other is our proposed new approach: deterministic parsing only estimating whether current segment modifies immediately right-hand side segment or not. Experimental results on Kyoto University corpus show that our system achieves the accuracy of 89.29% even with small training data (7958 sentences).

Keywords:

Parsing, Dependency Structure Analysis, Chunking, Machine Learning, Support Vector Machine

*Master's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT9951036, February 9, 2000.

Support Vector Machine を用いた 日本語係り受け解析*

工藤 拓

内容梗概

本稿では, Support Vector Machine (SVM) に基づく日本語係り受け解析手法を提案し, その評価を行なう. 既に, 決定木や最大エントロピー法等が統計的係り受け解析の学習モデルとして提案されているが, これらの手法は慎重な素性選択が要求されたり, 素性の組み合わせを効率良く学習できないといった問題がある. その一方で, SVM は従来からある学習モデルと比較して, 入力次元数に依存することなく極めて高い汎化能力を持ち, さらに Kernel 関数という概念を導入することで効率良く素性の組み合わせを考慮しながら学習することが可能である. 本稿では, SVM を係り受け解析に適用しその評価を行なう. その際, 任意の二文節間の係りやすさを数値化した行列を作成し, そこから動的計画法を用いて文全体を最適にする係り受け関係を求めるという従来法に基づく手法と, 直後の文節に係るか係らないかという観点のみで決定的に解析する提案手法の二つの適用方法の比較を行なう. 京大コーパスを用いて実験評価を行なった結果, 7958 文という非常に少ない学習データにもかかわらず 89.29% の高い精度を示した.

キーワード

構文解析, 係り受け解析, チャンキング, 機械学習, Support Vector Machine

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 修士論文, NAIST-IS-MT9951036, 2000 年 2 月 9 日.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Dependency structure analysis has been recognized as a basic technique in Japanese sentence analysis, and a number of studies have been proposed for years. Japanese dependency structure is usually defined in terms of the relationship between phrasal units called *'bunsetsu'* segments (hereafter "segments"). Generally, dependency structure analysis consists of two steps. In the first step, dependency matrix is constructed, in which each element corresponds to a pair of chunks and represents the probability of a dependency relation between them. The second step is to find the optimal combination of dependencies to form the entire sentence.

In previous approaches, these dependencies are given by manually constructed rules. However, rule-based approaches have problems in coverage and consistency, since there are a number of features that affect the accuracy of the final results, and these features usually relate to one another.

On the other hand, as large-scale tagged corpora have become available these days, a number of statistical parsing techniques which estimate the dependency probabilities using such tagged corpora have been developed[2, 5]. These approaches have overcome the systems based on the rule-based approaches. In addition, there are a number of machine learning algorithms has been applied to the dependency structure analysis, such as Decision Trees[6] and Maximum Entropy models[1, 9, 18, 23, 24, 22] have been applied to dependency or syntactic structure analysis. However, these models require an appropriate feature selection in order to achieve a high performance. In addition, acquisition of an efficient combination of features is difficult in these models.

In recent years, new statistical learning techniques such as Support Vector Ma-

chines (SVMs) [3, 25, 26] and Boosting[4] are proposed. These techniques take a strategy that maximize the margin between critical examples and the separating hyperplane. In particular, compared with other conventional statistical learning algorithms, SVMs achieve high generalization even with training data of a very high dimension. Furthermore, by optimizing the Kernel function, SVMs can handle non-linear feature spaces, and carry out the training with considering combinations of more than one feature.

Thanks to such predominant nature, SVMs deliver state-of-the-art performance in real-world applications such as recognition of hand-written letters, or of three dimensional images. In the field of natural language processing, SVMs are also applied to dependency structure analysis[13], chunking[14, 12] and text categorization[7, 8, 21], and are reported to have achieved high accuracy without falling into over-fitting even with a large number of words taken as the features.

In this thesis, we propose an application of SVMs to Japanese dependency structure analysis. Precisely, we propose two methods for applyining SVMs to Japanese dependency structure analysis. One is probabilistic model which has been widely applied for Japanese dependency structure analysis. The other is our proposed new approach: deterministic parsing only estimating whether current segment modifies immediately right-hand side segment. We use the features that have been studied in conventional statistical dependency analysis with a little modification on them.

Chapter 2

Support Vector Machines

In this chapter, we describe an algorithm and theoretical backgrounds of Support Vector Machines (SVMs). SVMs is powerful new type of learning algorithm based on recent advances in statistical learning theory. SVMs is now applied to large number of real-world applications such as text categorization, hand-written character recognition, etc., and delivers a state-of-the-art performance.

2.1. Optimal Hyperplane

Suppose the training data which belong either to positive or negative class as follows.

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_l, y_l)$$
$$\mathbf{x}_i \in \mathbf{R}^n,\ y_i \in \{+1, -1\}.$$

$\mathbf{x}_i$ is a feature vector of i-th sample, which is represented by an n dimensional vector ($\mathbf{x}_i = (f_1, \ldots, f_n) \in \mathbf{R}^n$). y_i is a scalar value that specifies the class (positive(+1) or negative(-1) class) of i-th data. Formally, one can define the pattern recognition problem as a learning and building process of the decision function $f : \mathbf{R}^n \to \{\pm 1\}$.

In basic SVMs framework, one tries to separate the positive and negative examples in the training data by a linear hyperplane:

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

SVMs finds the "optimal" hyperplane (optimal parameter $\mathbf{w}, b$) which separates the training data into two classes accurately. What dose "optimal" mean? In order to de-

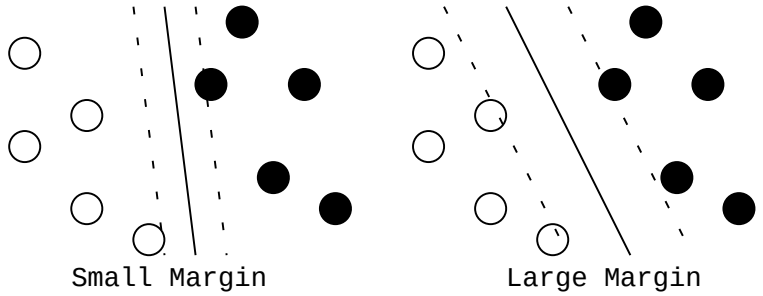


Figure 2.1. Two possible separating hyperplanes

fine it, we need to consider the **margin** between two classes. Figure 2.1 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 misclassification. We call the distance between those parallel dashed lines as **margin**. SVMs takes a direct strategy that finds the separating hyperplane which maximizes its margin.

In order to describe the separating hyperplane, we introduce the following form:

$$(\mathbf{w} \cdot \mathbf{x}_i) + b \geq \quad 1 \qquad \text{if } (y_i = 1) \tag{2.2}$$

$$(\mathbf{w} \cdot \mathbf{x}_i) + b \leq -1 \qquad \text{if } (y_i = -1). \tag{2.3}$$

(2.2) (2.3) can be written in one formula as:

$$y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq 1 \qquad (i = 1, \ldots, l). \tag{2.4}$$

Distance from the separating hyperplane to the point $\mathbf{x}_i$ can be written as:

$$d(\mathbf{w}, b; \mathbf{x}_i) = \frac{|\mathbf{w} \cdot \mathbf{x}_i + b|}{\|\mathbf{w}\|}.$$

Thus, the margin between two separating hyperplanes can be written as:

$$\begin{aligned} & \min_{\mathbf{x}_i; y_i = 1} d(\mathbf{w}, b; \mathbf{x}_i) + \min_{\mathbf{x}_i; y_i = -1} d(\mathbf{w}, b; \mathbf{x}_i) \\ = & \min_{\mathbf{x}_i; y_i = 1} \frac{|\mathbf{w} \cdot \mathbf{x}_i + b|}{\|\mathbf{w}\|} + \min_{\mathbf{x}_i; y_i = -1} \frac{|\mathbf{w} \cdot \mathbf{x}_i + b|}{\|\mathbf{w}\|} \\ = & \frac{2}{\|\mathbf{w}\|}. \end{aligned}$$

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

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

The solution of this optimization problem can be obtained by considering the following primal Lagrangian:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i \{y_i[(\mathbf{x}_i \cdot \mathbf{w}) + b] - 1\} \tag{2.5}$$

where the α_i are Lagrange multipliers. We must now minimize this Lagrangian with respect to $\mathbf{w}$ and b under the constraints $\alpha_i \geq 0$. At the saddle point, the solution $\mathbf{w}, b$ must satisify the following conditions:

$$\begin{aligned} \frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} &= 0 \quad \rightarrow \quad \sum_{i=1}^{l} \alpha_i y_i = 0 \\ \frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} &= 0 \quad \rightarrow \quad \mathbf{w} = \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}_i. \end{aligned} \tag{2.6}$$

Substituting these conditions into (2.5), we can obtain the following dual Lagrangian:

$$\begin{aligned} \text{Maximize :} \quad & L(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ \text{Subject to :} \quad & \alpha_i \geq 0, \; \sum_{i=1}^{l} \alpha_i y_i = 0 \qquad (i = 1, \ldots, l) \end{aligned} \tag{2.7}$$

(2.6) indicates that the optimal hyperplane (vector $\mathbf{w}$) is a liner combinations of the vectors of the training data. Namely, there is a Lagrange multiplier α_i for every training data $\mathbf{x}_i$. In this dual form problem, the training data $\mathbf{x}_i$ with non-zero α_i is called a Support Vector. Support Vectors can be considered as the minimal and critical elements which represent the all other training data. If all other training data were removed, one could obtain the same separating hyperplane.

By using the Support Vectors, $\mathbf{w}$ and b can thus be expressed as follows:

$$\mathbf{w} = \sum_{i;\mathbf{x}_i \in SVs} \alpha_i y_i \, \mathbf{x}_i \quad b = \mathbf{w} \cdot \mathbf{x}_i - y_i.$$

Finally, the decision function $f : \mathbf{R}^n \to \{\pm 1\}$ can be written as:

$$\begin{aligned} f(\mathbf{x}) &= \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x} + b) \\ &= \operatorname{sgn}\left(\sum_{i;\mathbf{x}_i \in SVs} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b \right). \end{aligned} \tag{2.8}$$

2.2. Generalization for the Nonseparable Case — Soft Margin Constraints

In the case where we cannot separate training examples linearly, "Soft Margin" method allows some classification errors that may be caused by some noise in the training examples. This can be done by introducing positive slack variables $\xi_i \geq 0$ in the constraints (2.2),(2.3).

$$\begin{aligned} (\mathbf{w} \cdot \mathbf{x}_i) + b &\geq 1 - \xi_i && \text{if } (y_i = 1) \\ (\mathbf{w} \cdot \mathbf{x}_i) + b &\geq -1 + \xi_i && \text{if } (y_i = -1) \end{aligned}$$

In this case, we minimize the following value instead of $\frac{1}{2}\|\mathbf{w}\|^2$.

$$L(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{l} \xi_i \tag{2.9}$$

The first term in (2.9) specifies the size of margin and the second term evaluates how far the training data are away from the optimal separating hyperplane. C is the parameter that defines the balance of two quantities. If one makes C larger, separating hyperplane becomes to evaluate classification error large, and if we make C small, the separating hyperplane becomes evaluate whole margin more significant, permitting some classification error.

Though we omit the details here, minimization of (2.9) is reduced to the following optimization problem:

$$\text{Maximize :} \qquad L(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{Subject to :} \qquad 0 \leq \alpha_i \leq C, \qquad \sum_{i=1}^{l} \alpha_i y_i = 0 \qquad (i = 1, \ldots, l).$$

Only the difference from the separable case is that the α_i now has an upper bound of C. Usually, the value of C is estimated experimentally.

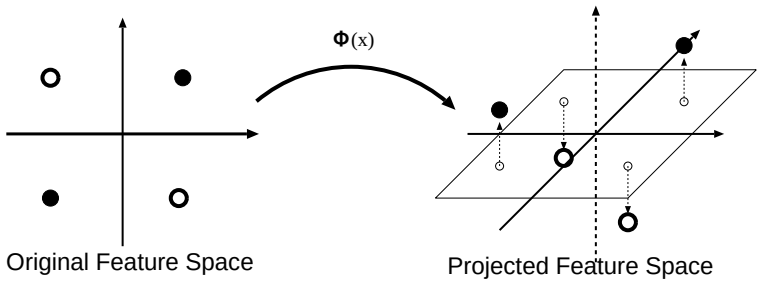


Figure 2.2. Projecting 2-D feature space onto 3-D space

2.3. High-dimensional mapping and Generalized Inner Products — Kernel function

In general classification problems, there are cases in which it is unable to separate the training data linearly.

Suppose the exclusive-or (XOR) problem in two dimensional feature space (Figure 2.2, left-hand side). It is not possible to construct the separable liner hyperplane. However, one may obtain separable hyperplane if one projects the original two dimensional feature space into three dimensional feature space by giving some projecting function like (Figure 2.2):

$$\Phi(\mathbf{x}): \ (x_1, x_2) \rightarrow (x_1, x_2, x_1 x_2).$$

More generally, the linearly unseparable training data could be separated linearly by expanding all combinations of features as new ones, and projecting them onto a higher-dimensional space. However, such a naive approach requires enormous computational overhead, since one must carry out vector operations in higher dimensional space. For example, if one tries to construct polynomials of degree $d < n$ in n-dimensional feature space, one needs more than $(n/d)^d$ features.

Let us consider the case where we project the training data $\mathbf{x}$ onto a higher-dimensional space by using projection function Φ chosen *a priori* [1]. As we pay attention to the objective function (2.7) and the decision function (2.8), these functions depend only on the dot products of the input training vectors. If we could calculate the dot products

[1] In general, $\Phi(\mathbf{x})$ is a mapping into Hilbert space.

from $\mathbf{x}_1$ and $\mathbf{x}_2$ directly without considering the vectors $\Phi(\mathbf{x}_1)$ and $\Phi(\mathbf{x}_2)$ projected onto the higher-dimensional space, we can reduce the computational complexity considerably. Namely, we can reduce the computational overhead if we could find the function K that satisfies:

$$\Phi(\mathbf{x}_1) \cdot \Phi(\mathbf{x}_2) = K(\mathbf{x}_1, \mathbf{x}_2). \tag{2.10}$$

On the other hand, since we do not need Φ itself for actual learning and classification, all we have to do is to prove the existence of Φ that satisfies (2.10) provided the function K is selected properly. It is known that (2.10) holds if and only if the function K satisfies the *Mercer condition* [26]. In this way, instead of projecting the training data onto the high-dimensional space, we can decrease the computational overhead by replacing the dot products, which is calculated in optimization and classification steps, with the function K. Such a function K is called a **Kernel function**. Kernel function can be considered as a generalized inner products of given two vectors. Some representative examples of Kernel functions are: [2]

$$K(\mathbf{x}, \mathbf{y}) = \tanh(a\ \mathbf{x} \cdot \mathbf{y} - b) \tag{2.11}$$

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{-\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right) \tag{2.12}$$

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^d. \tag{2.13}$$

Using a Kernel function, we can rewrite the dual form Lagrangian and decision function as:

$$L(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{2.14}$$

$$y = \operatorname{sgn}\left(\sum_{i;\mathbf{x}_i \in SVs} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right). \tag{2.15}$$

Substituting the Kernel function K in the decision function (2.15) with each of the above examples, (2.11) represents the so-called two layered neural networks, (2.12) represents Radial Basis Function (RBF) network models. (2.13) is called as d-th polynomial kernel. Use of d-th polynomial kernel function allows us to build an optimal separating hyperplane which takes into account all combination of features up to d.

[2] $\tanh(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function

It is easy to prove the existence of actual projecting function $\Phi(\mathbf{x})$, when one applies polynomial kernel function with second degree ($d = 2$) to the previous exclusive-or (XOR) problem in two dimensional feature space.

$$\begin{aligned} K(\mathbf{x}, \mathbf{y}) &= (\mathbf{x} \cdot \mathbf{y} + 1)^2 \\ &= (x_1 y_1 + x_2 y_2 + 1)^2 \\ &= (x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 y_1 x_2 y_2 + 1) \\ &= (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, 1) \cdot (y_1^2, y_2^2, \sqrt{2}y_1, \sqrt{2}y_2, \sqrt{2}y_1 y_2, 1)^T \end{aligned}$$

$$\Phi(\mathbf{x}): \ (x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, 1)$$

By using second polyonimal kernel function, the original two dimensional feature space is projected onto six dimensional feature space. This means that the use of polynomial kernel function allows us to build a linearly separable hyperplane even in the case of exclusive-or (XOR) problem.

2.4. Generalization ability of SVMs

In this section, we introduce a uniform generalization theory for machine leaning algorithms. Suppose that training data and test data are generated from the same underlying i.i.d[3] probability distribution $P(\mathbf{x}, y)$. Then the classification problems consists of finding a mapping function $f : X \rightarrow Y$ that minimizes the *risk* of misclassification

$$R[f] = \frac{1}{2} \int |f(\mathbf{x}) - y| dP(\mathbf{x}, y).$$

The problem is that one cannot estimate $R[f]$ directory since the distribution $P(\mathbf{x}, y)$ is unknown. Instead of minimize the true *risk*, usually the following *empirical risk* is used.

$$R_{emp}[f] = \frac{1}{2l} \sum_{i=1}^{l} |f(\mathbf{x}_i) - y_i|$$

However, it is know that these *Empirical Risk Minimization* principle dose not always guarantee a small actual risk. Therefore we do have to find a novel method to estimate the true *risk* indeed.

[3]independently, identically, distribution

Statistical Learning Theory [26] states that *empirical risk* and *risk* 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 as*

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{h(\ln \frac{2l}{h} + 1) - \ln \frac{\eta}{4}}{l}} \tag{2.16}$$

where h is a non-negative integer called the Vapnik Cheroveniks (VC) dimension, and is a measure of the capacity of the given decision function. The right side term of (2.16) is called as **VC bound**.

Actually, almost all previous machine learning techniques are based on *Empirical Risk Minimization* principle, then try to only minimize the *empirical risk* $R_{emp}[f]$ under the fixed VC dimension. However, it is difficult to estimate an appropriate VC dimension for individual classification tasks. In other words, one can not estimate precisely the complexity and capacity of the given tasks.

On the other hand, *Structural Risk Minimization* principle tries to choose the function f which minimizes the guaranteed VC bound. (2.16) shows that we must minimize the VC dimension in order to minimize the VC bound. It is known that the following theorem holds for VC dimension h and margin M[26].

Theorem 2 (Vapnik) *If we suppose n as the dimension of given training examples, M as the margin, and D as the smallest diameter which enclose all training data, then VC dimension h of the SVMs is bounded as*

$$h \leq \min(D^2/M^2, n) + 1. \tag{2.17}$$

In order to minimize the VC dimension h, we have to maximize the margin M, which is exactly the strategy that SVMs takes. In addition, since D is decided by the given Kernel function, (2.17) also gives some criteria for selecting the appropriate kernel function and the soft margin parameter .

Vapnik gives an alternative bound for the *Risk* of SVMs:

Theorem 3 (Vapnik) *If we suppose $E_l[f]$ is an error rate estimated by Leave-One-Out procedure, $E_l[f]$ is bounded as*

$$E_l[f] \leq \frac{Number\ of\ Support\ Vectors}{Number\ of\ training\ samples}. \tag{2.18}$$

Leave-One-Out procedure is a simple method to examine the capacity of the decision function — first by removing one element from the training data, we constructs the decision function on the basis of the remaining training data, and then tests the removed element. In this fashion, we tests all l elements of the training data using l different decision functions.

(2.18) 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, $E_l[f]$ becomes the right side term of (2.18). Although *Leave-One-Out* bound is elegant and allows us to estimate the rough bound of the *Risk*, there seems to be many situations where the actual error increases even though the number of support vectors decreases. Actually, it is known that this bound is less predictive than the VC bound.

Chapter 3

Statistical Dependency Structure Analysis

This chapter presents the statistical dependency analysis of Japanese dependency structure analysis and is organized as follows:

First, we introduce the probabilistic model which has been widely applied, and describe how we integrate SVMs into this probabilistic model. Second, we describe the proposed new approach, in which we try to parse a sentence determinately only estimating the current segment modifies the immediately right-hand side segment. Finally, we describe the advantage of applying SVMs to dependency structure resolution.

3.1. The Probablistic Model

This section describes a general formulation of the probabilistic model and parsing techniques which have been applied for Japanese statistical dependency analysis.

First of all, we define a sequence of segments $\{b_1, b_2 \ldots, b_m\}$ as B, and the sequence dependency pattern $\{Dep(1), Dep(2), \ldots, Dep(m-1)\}$ as D, where $Dep(i) = j$ means that the segment b_i depends on (modifies) the segment b_j. In this framework, we suppose that the dependency sequence D satisfies the following constraints.

1. Except for the rightmost one, each segment depends on (modifies) exactly one of the segments appearing to the right.

2. Dependencies do not cross one another.

Statistical dependency structure analysis is defined as a searching problem for the dependency pattern D that maximizes the conditional probability $P(D|B)$ of the input sequence under the above-mentioned constraints.

$$D_{best} = \underset{D}{\operatorname{argmax}}\, P(D|B)$$

If we assume that the dependency probabilities are mutually independent, $P(D|B)$ could be rewritten as:

$$P(D|B) = \prod_{i=1}^{m-1} P(Dep(i)=j \mid \mathbf{f}_{ij})$$
$$\mathbf{f}_{ij} = \{f_1, \ldots, f_n\} \in \mathrm{R}^n.$$

$P(Dep(i) = j \mid \mathbf{f}_{ij})$ represents the probability that b_i depends on (modifies) b_j. $\mathbf{f}_{ij}$ is an n dimensional feature vector that represents various kinds of linguistic features related with the segments b_i and b_j. Usually, the matrix which holds the probabilities $M[i,j] = P(Dep(i)=j \mid \mathbf{f}_{ij})$ is called as **dependency matrix**.

We obtain D_{best} taking into all the combination of these probabilities. Generally, the optimal solution D_{best} can be identified by using bottom-up parsing algorithm such as CYK algorithm[10]. For Japanese dependency structure analysis, Sekine suggests an efficient parsing technique which parses from the end of a sentence and employs beam search[20].

Briefly, the parsing process with probabilistic model consists of the following two steps:

1. Build the dependency matrix — estimate the probabilities of dependency
2. Search the best dependency relations which maximize the conditional probability $P(D|B)$ of the input sequence.

Figure 3.1 shows the typical example of the parsing process with probabilistic model.

The problem in the dependency structure analysis is how one can estimate the dependency probabilities accurately and build a good dependency matrix. There are mainly two approaches to solve this problem. One is the rule-based approach. In this approach, dependency matrix is given by a hand-crafted rules. This approach had been used in the early studies of dependency structure analysis. However, rule-based approaches have problems in coverage and consistency, since there are a number of

features that affect the accuracy of the final results, and these features usually relate to one another. In addition, it is too difficult to incorporate a scoring scheme in rule-based systems.

The other is corpus-based statistical approach. As large-scale tagged corpora have become available these days, a number of statistical parsing techniques which estimate the dependency probabilities using such tagged corpora have been developed[2, 5]. These approaches have overcome the systems based on the rule-based approaches. Decision Trees[6] and Maximum Entropy models[18, 23, 22, 1] have been applied to dependency or syntactic structure analysis.

3.2. Integration of SVMs into probabilistic model

In this thesis, we simply integrate SVMs into above mentioned probabilistic model. In order to apply SVMs, the following two problems arise.

- What are the positive and negative examples for the classifiers of SVMs? We have to prepare positive and negative examples since SVMs is a binary classifier.
- How one can estimate the probabilities of dependencies? SVMs have no potential to output the probabilities.

For the first problem, we adopt a simple and effective method: Out of all combination of two segments in the training data, we take a pair of segments that are in a dependency relation as a positive example, and two segments that appear in a sentence but are not in a dependency relation as a negative example.

$$\bigcup_{\substack{1 \le i \le m-1 \\ i+1 \le j \le m}} (\mathbf{f}_{ij}, y_{ij}) = \{(\mathbf{f}_{12}, y_{12}), (\mathbf{f}_{23}, y_{23}), \ldots, (\mathbf{f}_{m-1\,m}, y_{m-1\,m})\}$$

$$\mathbf{f}_{ij} = \{f_1, \ldots, f_n\} \in \mathbf{R}^n$$
$$y_{ij} \in \{\mathrm{Depend}(+1),\ \ \mathrm{Not\text{–}Depend}(-1)\}$$

For the second problem, we define dependency probability as:

$$P(Dep(i)=j \mid \mathbf{f'}_{ij}) = \tanh\left(\sum_{k,l;\mathbf{f}_{kl} \in SVs} \alpha_{kl} y_{kl} K(\mathbf{f}_{kl}, \mathbf{f'}_{ij}) + b\right). \tag{3.1}$$

1st step: Build the Dependency Matrix

Modifiee

Modifier	彼は	彼女の	温かい	真心に	感動した。
彼は	0.0	0.1	0.2	0.1	0.6
彼女の	0.0	0.0	0.3	0.5	0.2
温かい	0.0	0.0	0.0	0.8	0.2
真心に	0.0	0.0	0.0	0.0	1.0
感動した。	0.0	0.0	0.0	0.0	0.0

$$M [i, j] = P (D(i) = j \mid f_{ij})$$

2nd step: Find the Best Dependency Relation

Sekine's backward beam search method (beam width=3)

ID	1	2	3	4	5	
	彼は	彼女の	温かい	真心に	感動した。	
Cand1				5(1.0)		Prob. 1.0
Cand1			4(0.8)	5(1.0)		Prob. 0.8
Cand2			5(0.2)	5(1.0)		Prob. 0.2
Cand1		4(0.5)	4(0.8)	5(1.0)		Prob. 0.4
Cand2		3(0.3)	4(0.8)	5(1.0)		Prob. 0.24
Cand3		3(0.2)	4(0.8)	5(1.0)		Prob. 0.16
Cand1	5(0.6)	4(0.5)	4(0.8)	5(1.0)		Prob. 0.24 ◀ Best
Cand2	3(0.6)	3(0.3)	4(0.8)	5(1.0)		Prob. 0.14
Cand3	5(0.6)	3(0.2)	4(0.8)	5(1.0)		Prob. 0.08

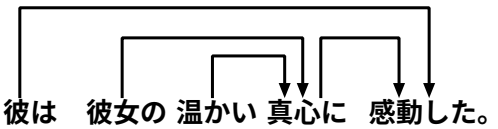


Figure 3.1. Example of the parsing process with probabilistic model

(3.1) indicates that the distance between test data $\mathbf{f}'_{ij}$ and the separating hyperplane is put into the sigmoid function, assuming it represents the probability value of the dependency relation. Precisely, this transformation dose not give us a true probability however, there is a report which states that sigmoid function experimentally gives a good approximation of probability function from the decision function of SVMs[17]. We adopt this method in our experiment to transform the distance measure obtained in SVMs into a probability function.

By giving dependency probabilities, we can analyze dependency structure with a framework of conventional probabilistic model.

3.3. Cascaded Chunking Model

In the probabilistic model, one has to estimate the *probabilities* of each dependency relations. For this purpose, we introduce some kind of heuristics — transforming the distance from the separating hypeplane into *pseudo* probabilities using sigmoid function. Experimentally, it is known that this transformation will give us a good estimation however, there are no theoretical analysis which states that use of sigmoid function can guarantee the *true* probability of dependency relation.

In this thesis, we introduce a new method for Japanese dependency structure analysis, which dose not require the probabilities of dependencies and parses a sentence determinately. Proposed new method can be combined with not only SVMs but also any type of machine learning algorithms which have no potential to output the probabilities for the estimated class.

The original idea of our proposed new method stems form the cascaded chucking method which has been applied in English parsing[19, 18]. Let us introduce the basic framework of cascaded chunking parsing method:

1. Give the series of base phrases as a input for this algorithm.
2. Scanning from the beginning of given sentence, chunk some arbitrary series of base phrases into single non-terminal node. (phase 1.)
3. Leave only the head phrase from the chunked phrases, and delete the all other phrases. (phase 2.)

4. Finish the algorithm if the number of remaining non-terminal node reaches one, otherwise return to step 2.

This process can be recognized as cascaded tagging, since one can regard the chunking as the process of annotation of tags, which represents the status of the individual chunks, to the each base phrases.

We apply this cascaded chunking parsing technique to the Japanese dependency structure analysis. Considering Japanese is head final language and more than three segments do not consists of the single non-terminal node, we can simplify the process of Japanese dependency structure analysis as follows:

1. Give **O** tag to all segments . **O** tag represents the dependency relation of the current segment is undecided.

2. Estimate whether the each segment with **O** tag modifies the immediately right hand side segment or not. The case of modifying, the **O** tag is replaced with **D** tag.

3. Delete the all segments with **D** tag appearing to immediately right hand side of the segment with **O** tag. (phase 2.)

4. Finish the algorithm if the number of remaining segment reaches one, otherwise, return to the step 2.

Figure 3.2 shows the example of the parsing process with proposed cascaded chunking model. In addition, figure 3.3 represents the pseudo code of this algorithm.

We think this proposed cascaded chunking model has the following advantages compared with the probabilistic model.

- **Simplification and Efficiency**
  The probabilistic model is not efficient since one has to estimate the probabilities of every candidates which may have dependency relation. The algorithm of our proposed cascaded chunking model is simpler, and easier to implement than that of the probabilistic model. In addition, our proposed cascaded chunking model is quite efficient since it requires minimal estimation of dependency relation.

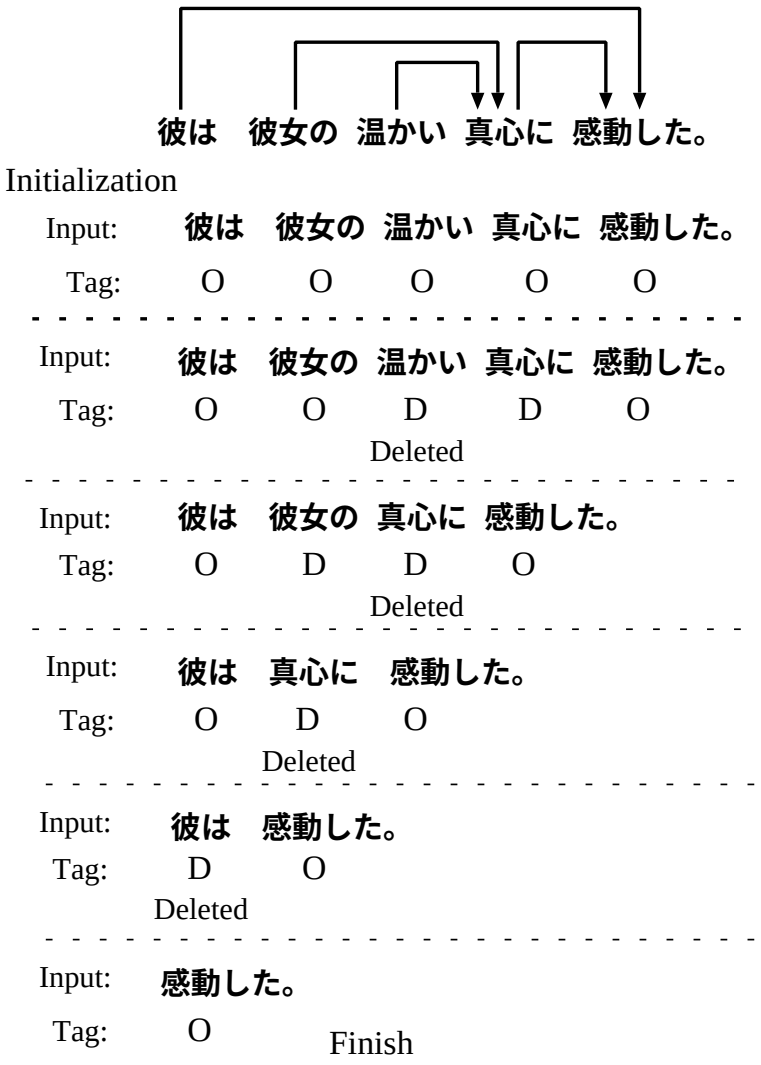


Figure 3.2. Example of the parsing process with cascaded chunking model

$Input := \{Segment[1], .., Segment[n]\}$ // Input segments
$Tag := \{O, O, .., O\}$ // Tags
$Del := \{0, 0, , .., 0\}$ // delete flags
$Tag[0] := O$ // dummy
$Output := \phi$ // Hash, Key:mofier, Value:modifiee

function $estimate(src, dst)$
begin
    $feature\_set := \{src, dst\}$
    $estimate := any\_classifer(feature\_set)$
end

while $(length(Input) > 1)$
begin
    // Phase 1.
    for $i := 1$ **to** $length(Input) - 1$
    begin
        $Del[i] := 0$

        if $(i = (length(Input) - 1))$ **then**
            $Tag[i] := D$
        else if $(Tag[i] = O)$ **then**
            $Tag[i] := estimate(Input[i], Input[i + 1])$

        if $(Tag[i] = D)$ **then**
        begin
            $Output[Input[i]] := Input[i + 1]$
            if $(Tag[i - 1] = O)$ **then**
                $Del[i] := 1$
        end
    end

    // Phase 2.
    for $i := length(Input) - 1$ **downto** 1
    begin
        if $(Del[i] = 1)$ **then**
        begin
            $delete(Input[i])$
            $delete(Tag[i])$
        end
    end
end

Figure 3.3. Pseudo code of cascaded chunking algorithm

function $estimate(src, dst)$
begin
    if (src modifies dst in the training data) **then**
        $estimate := D$
    else
        $estimate := O$

    // Storing as a new training data
    $feature\_set := \{src, dst\}$
    $push(TrainigData, \{estimate, feature\_set\})$
end

Figure 3.4. The function *estimate* in the case of extracting trainin data

- **Automatic detection of cross dependency**
  Previous probabilistic model cannot detect the cross dependency relation by itself. We have to eliminate the case of cross dependency relation at any stage of actual parsing, and this process is usually implemented in the individual parser. On the other hand, our proposed method has a ability to detect the case of cross dependency relation even though the algorithm is simple enough.

- **Not assuming the independence constraints in dependency relation**
  The probabilistic model assumes that each dependency structure are mutually independent. However, there are cases in which one cannot parse correctly with constraints of independence. For examples, coordinates structure cannot be always parsed with the constraints of independence. Our proposed cascaded chunking model will parse and estimate relations simultaneously. This means that one can use all dependency relations, which have narrower scope than that of current estimating relation, as feature sets. We describe the details about it in Chapter 3.4.

- **Parsing from beginning of given sentence**
  In the step of actual parsing, there are some techniques which parse form the end of sentence, since the nearer to the end of sentence, the fewer candidates of dependency should be estimated[20]. We feel a gap about these techniques since our human beings useally parse from the beggining of the sentence. Our

proposed method parses from the beggining of sentence.

- **Identitication parsing with extraction of training data**
  Replacing the function *estimate* with the function in figure 3.4, one can carry out the extraction of training data sets. Namely, the process of the extraction of training data sets can be recocnized as a one of parsing process. One point which differs from the parsing is that dependency relation is estimated by refereinig the training data set itself. Considering a number of segments modify the immideately right hand side segment, one can reduce the training data (pairs of positive and negative examples) considerebly. This implies that one can deal with larger size of training data using any machine learning algorithms which requires huge number of computational overhead.

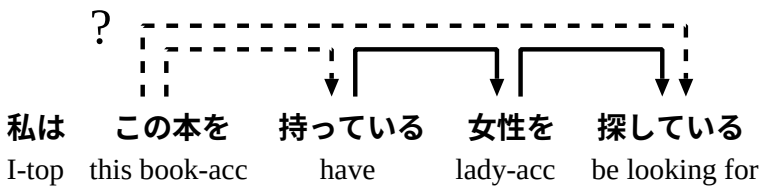
- **Independency on machine learning algorithms**
  In the cascaded chunking model, we can apply any machine learning algorithms only with an ability of binary classification, since proposed method will parse sentence deteministetly only estimating wheter current segment modfies the immideately right hand side segment or not. Probabilities of dependency is not always necessary for our proposed method.

- **Ability to annotate the type of dependency relation**
  We can say that probabalistic model is specialized to detect only the dependency relation. Namely, It is difficult to extend probabalistic model to annotate the type of dependency relation such as normal dependency relation, coordinate structure or apposition structure etc. Our proposed method can deal with these types of dependency relations without remodeling the algorithm itself. All we have to do is just to change the tag sets which represent the type of dependency relations.

3.4. Static and Dynamic Features

Features that are supposed to be effective in Japanese dependency analysis are: head words and their parts-of-speech, particles and inflection forms of the words that appear at the end of chunks, distance between two chunks, existence of punctuation marks. As those are solely defined by the pair of chunks, we refer to them as **static features**.

Japanese dependency relations are heavily constrained by such static features since the inflection forms and postpositional particles constrain the dependency relation. However, when a sentence is long and there are more than one possible dependents, static features, by themselves cannot determine the correct dependency. Let us look at the following example.



In this example, “この本を (this book-acc)” may modify either of “持っている (have)” or “探している (be looking for)” and cannot be determined only with the static features. However, “女性を (lady-acc)” can modify the only the verb “探している,”. Knowing such information is quite useful for resolving syntactic ambiguity, since two accusative noun phrses hardly modify the same verb. It is possible to use such information if we add new features related to other modifiers. In the above case, the chunk “探している” can receive a new feature of accusative modification (by “女性を”) during the parsing process, which precludes the chunk “この本を” from modifying “探している” since there is a strict constraint about double-accusative modification that will be learned from training examples. We decided to take into consideration all such modification information by using functional words or inflection forms of modifiers.

Using such information about modifiers in the training phase has no difficulty since they are clearly available in a tree-bank. On the other hand, they are not known in the parsing phase of the test data. This problem can be easily solved if we adopt a bottom-up parsing algorithm and attach the modification information dynamically to the newly constructed phrases (the chunks that become the head of the phrases). We refer to the features that are added incrementally during the parsing process as **dynamic features**.

More specificly, we take the follwoing three type of dynamic features in our experiments. (figure 3.5)

1. The segments which modifiy the current estimating modifiee. (A)

2. The segments which modifiy the current estimating modifier. (B)

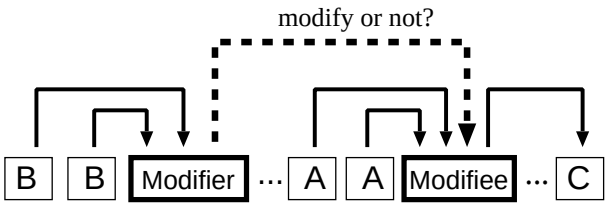


Figure 3.5. Three types of Dynamic Features

3. The segment which is modified by the current estimating modifiee. (C)

In this thesis, we try to apply these dynamic features to both of two type of parsing model — probablistic model and cascaded chunking model. Actually, the methods to deal with these dynamic features sligthly differ between these two models.

In the probablistic model, we use Sekine's bacward parsing method as actual parsing algorithm. Use of this alogritm dose not allow us to commit the segments, whch appear to left hand side of the current estimating modifier, as dynamic features. Namely, we can only use dynamic features whose type are A or C. In the cascaded chunking model, we can use all the type of dynamic features – A, B or C. However, these dynamic features must have narrower scope than that of current estimating two segments.

3.5. Why SVMs give us good estimation?

In order to obtain good estimation for dependency structure analysis, one must consider the combinations of features — modifier and modifiee. In other words, it will be hard to confirm that a dependency relation is constructed with only the features of the modifier or the modfiee. It is quite natural that dependency relation is decided by at least the information from both of these two segments. The problem is how one can extract or select these effective combinations of features from the given two segments.

In order to overcome this problem, most of all previous systems select the approaches based on **parametric probabilistic** framework. In this framework, the combinations of features are decided by manually or heuristically. Then, from these manually given parametric space, one tries to estimate the probabilities using some well-known estimator such as Maximum likelihood or Maximum Entropy model. We believe these manual selection dose not necessarily cover all relevant combinations that

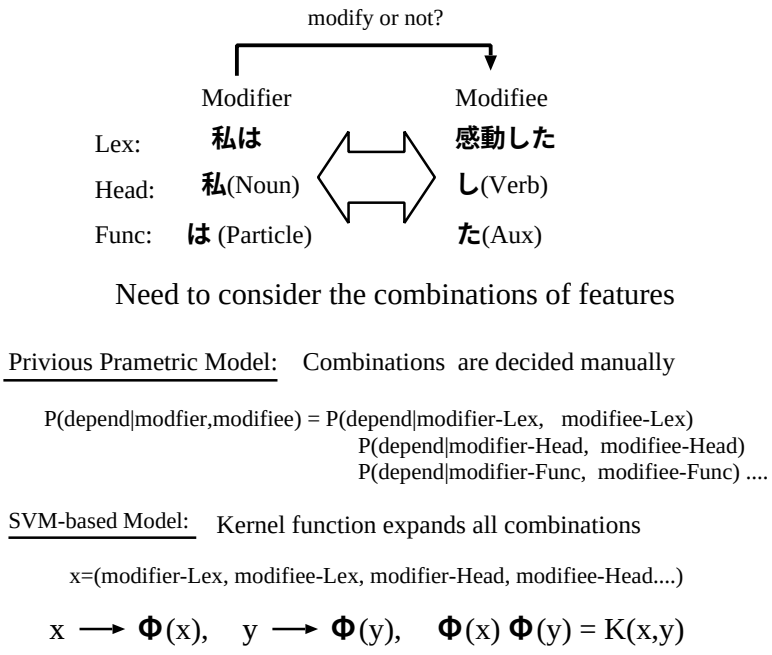


Figure 3.6. Previous Parametric Model vs SVM-based Model

are important in the determination of dependency relation. Of course we can optimize these effective combinations using statistical model selection scheme such as Cross-validation, AIC and MDL. However, it is difficult to select effective combinations even with these statistical model selection schemes, since these estimator usually require greedy optimization, and the large number of computational overhead will be required, considering the number of combinations is quite large.

On the other hand, in our proposed SVMs-based approach, one needs not to take these manual and heuristic selection of effective features. The main advantage of SVMs-based approach is that one can use Kernel functions. For example, by using polynomial Kernel function, combinations of given features can be automatically expanded as new feature sets without increasing computational overheads. In addition, one can avoid overfitting and have robust estimation even with these high dimensional feature space, since SVMs takes *Structural Risk Minimization* principle, and has a generalization ability which dose not depend on the given feature dimension. This implies that SVMs has a ability to select effective features sets, including their combinations, even with high dimensional feature space expanded by some Kernel function.

We believe that our proposed model is better than previous parametric probability model from the viewpoints of coverage and consistency, since our model learns the combination of features without increasing the computational complexity. If we want to reconsider them, all we have to do is just to change the Kernel function. The computational complexity depends on the number of support vectors not on the dimension of the Kernel function.

Chapter 4

Experiments and Discussion

This chapter summarizes the experiments and the results of the proposed dependency analysis using a tree-bank data.

4.1. Experiments Setting

We use Kyoto University text corpus (Version 2.0) consisting of articles of Mainichi newspaper annotated with dependency structure[16]. 7,958 sentences from the articles on January 1st to January 7th are used for the training data, and 1,246 sentences from the articles on January 9th are used for the test data. For the kernel function, we used the polynomial function (2.13). We set the soft margin parameter C to be 1.

The feature set used in the experiments are shown in Table 4.1. The static features are basically taken from Uchimoto's list[23] with little modification. In Table 4.1, 'Head' means the rightmost content word in a segment whose part-of-speech is not a functional category. 'Type' means the rightmost functional word or the inflectional form of the rightmost predicate if there is no functional word in the segment. The static features include the information on existence of brackets, question marks and punctuation marks etc. Besides, there are features that show the relative relation of two segments, such as distance, and existence of brackets, quotation marks and punctuation marks between them.

Here, we describe about the details how we introduce dynamic features. Generally, when we try to determine one dependency relation, there are more than one candidates for dynamic features with type A or B. If we commit their all information such as POS

Static Features	Modifier/Modifiee segments	**Head** (surface-form, POS, POS-subcategory, inflection-type, inflection-form), **Type** (surface-form, POS, POS-subcategory, inflection-type, inflection-form), brackets, quotation-marks, punctuation-marks, position in sentence (beginning, end)
	Between two segments	distance(1,2-5,6-), case-particles, brackets, quotation-marks, punctuation-marks
Dynamic Features	Type A,B	Form of inflection represented with *functional representation*
	Type C	POS and POS-subcategory of Head word

Table 4.1. Features used in our experiments

tags, functional words and inflection forms independently as new dynamic feature sets, we could not distinguish which POS tag corresponds to functional words or inflection forms. In order to avoid this problem, we introduce single representation which express POS tag, functional word and inflection form simultaneously. We call this single representation as *functional representation*. More specifically, we use the lexical form if the word's POS is particle, adverb, adnominal or conjunction. We use the inflection form if the word has inflection. We use the POS tags for others. For the dynamic features with type C, we use their POS tag and POS-subcategory.

For the basic evaluation measure, we use dependency accuracy and sentence accuracy. dependency accuracy means the percentage of correct dependencies out of all dependencies relation. The sentence accuracy represents the percentage of sentences in which all dependencies are determined correctly.

Training data size	Dependency Accuracy	Sentence Accuracy
1172	86.52%	39.31%
1917	87.21%	40.06%
3032	87.67%	42.94%
4318	88.35%	44.15%
5540	88.66%	45.20%
6756	88.77%	45.36%
7958	89.09%	46.17%

Table 4.2. Result of probabilistic model with dynamic feature, A ($d = 3, k = 5$)

Training data size	Dependency Accuracy	Sentence Accuracy
1172	86.12%	38.50%
1917	86.81%	39.80%
3032	87.62%	42.45%
4318	88.33%	44.47%
5540	88.40%	43.66%
6756	88.55%	45.04%
7958	88.77%	45.04%

Table 4.3. Result of probabilistic model without dynamic features ($d = 3, k = 5$)

4.2. Results of Experiments

4.2.1 Probabilistic model

Table 4.2 shows the result of parsing accuracy based on probabilistic model under the condition $k = 5$ (beam width), and $d = 3$ (dimension of the polynomial functions used for the kernel function). We can achieve 89.09% when we used all training data (7958 sentences).

Training data size	Dependency Accuracy	Sentence Accuracy
1172	86.66%	42.29%
1917	87.23%	42.94%
3032	87.87%	44.63%
4318	88.48%	44.63%
5540	88.64%	45.36%
6756	88.84%	47.05%
7958	89.29%	47.53%

Table 4.4. Result of cascaded chunking model with all dynamic feature ($d = 3$)

Training data size	Dependency Accuracy	Sentence Accuracy
1172	86.72%	41.32%
1917	87.43%	42.53%
3032	87.49%	43.26%
4318	88.16%	44.31%
5540	88.17%	43.83%
6756	88.22%	44.15%
7958	88.72%	45.20%

Table 4.5. Result of cascaded chunking model without dynamic features ($d = 3$)

4.2.2 Cascaded chunking model

Table 4.4 shows the result of parsing accuracy based on cascaded chunking model under the condition $d = 3$ (dimension of the polynomial functions used for the kernel function). We can achieve 89.29% when we used all training data (7958 sentences).

4.3. Effects of Dynamic Features

In this section, we describe how much the proposed dynamic features contribute to improve accuracy. Table 4.3 and Table 4.4 show the accuracy when only static features are used. Generally, the results with dynamic feature set is better than the results

Deleted type of dynamic feature	Difference of accuracy without each feature	
	Dependency Accuracy	Sentence Accuracy
A	-0.28%	-0.89%
B	-0.10%	-0.89%
C	-0.28%	-0.56%
AB	-0.33%	-1.21%
AC	-0.55%	-0.97%
BC	-0.54%	-1.61%
ABC	-0.58%	-2.34%

Table 4.6. Effects of dynamic features with cascaded chunking model

without them. The results with dynamic features constantly outperform that with static features only. In most of cases, the improvements is significant.

In addition, we investigate how much each type of dynamic feature contributes to improve accuracy. Table 4.6 summarizes the performance without each type of dynamic feature or their combinations, when we apply cascaded chunking model. From the results shown in 4.6, we can conclude that any type of dynamic features are effective to improve the dependency and sentence accuracy.

4.4. Training data vs Accuracy

Figure 4.1 and 4.2 show the relationship between the size of the training data and the parsing accuracy. This figures also show the accuracy of with and without the dynamic features.

We can achieve accuracy of 86.52% with probabilistic model, and accuracy of 86.66% with cascaded chunking model for test data even with small training data (1172 sentences). This is due to a good characteristic of SVMs to cope with the data sparseness problem. Furthermore, it achieves almost 100% accuracy for the training data, showing that the training data are completely separated by appropriate combination of features. Generally, selecting those specific features of the training data tends to cause overfitting, and accuracy for test data may fall. However, the SVMs method achieve a high accuracy not only on the training data but also on the test data. We claim that this

Dimension of Polynomial Kernel	Dependency Accuracy		Estimated VC dimension	Estimated LOO Bound
	Test	Training		
1	67.09%	90.73%	14446	0.243
2	86.10%	99.77%	187878	0.194
3	86.52%	99.97%	215511	0.232
4	86.36%	99.99%	394709	0.278
5	86.00%	99.99%	885006	0.334

Table 4.7. Dimension vs Accuracy (1172 sentences, probabilistic model, $k = 3$)

Dimension of Polynomial Kernel	Dependency Accuracy		Estimated VC dimension	Estimated LOO Bound
	Test	Training		
1	84.05%	97.68%	55500	0.301
2	86.42%	99.99%	53387	0.372
3	86.65%	99.99%	78657	0.463
4	86.38%	99.99%	158024	0.553
5	85.95%	99.99%	366320	0.641

Table 4.8. Dimension vs Accuracy (1172 sentences, cascaded chunking model)

is due to the high generalization ability of SVMs. In addition, observing at the learning curve, further improvement will be possible if we increase the size of the training data.

4.5. Kernel Function vs Accuracy

Table 4.7 and 4.8 show the relationship between the dimension of the kernel function and the parsing accuracy. In addition, in order to investigate how VC-Bound and Leave-One-Out bound (LOO bound) can predict *true* error rate, these tables also show the error-rate estimated with these theoretical bound.

As a result, the case of $d = 3$ gives the best accuracy with both of two models. In addition, the accuracies with the case of $d = 1$ are significantly worse than the accuracies of other cases. With probabilistic model, the separating hyperplane seems not to be built in the case of $d = 1$, taking into account the accuracies of 67.09%

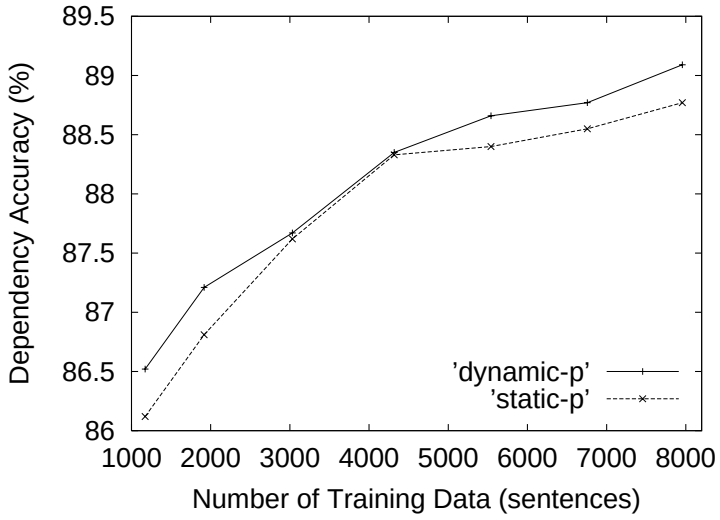


Figure 4.1. Training Data vs Accuracy (probabilistic model)

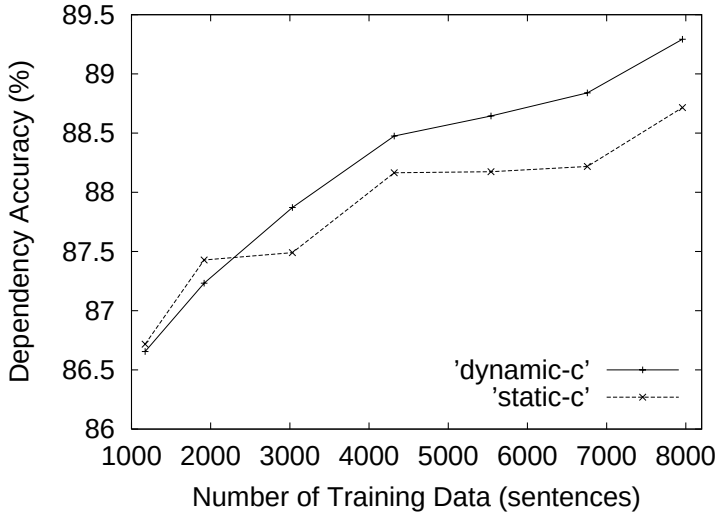


Figure 4.2. Training Data vs Accuracy (cascaded chunking model)

for test data and 90.73% for training data. This result supports our intuition that we need a combination of at least two features. Namely, it will be hard to confirm a dependency relation with only the features of the modifier or the modfiee. It is natural that a dependency relation is decided by at least the information from both of two chunks.

Ignoring the case of $d = 1$, error rates estimated by VC dimension and LOO bound show minimum in the case of $d = 2$. This means these theoretical bounds predict that best kernel dimension may be $d = 2$. Actually they cannot predict the optimal choice of kernel dimension ($d = 3$), however, we think this prediction seems to be reasonable and valuable, considering the differences of accuracies between the case of $d = 2$ and $d = 3$ are not significant, and size of training data used in this experiments is so small (1172 sentences) that the theoretical bounds cannot show their prediction abilities.

4.6. Beam width vs Accuracy

In the probabilistic model, we are allowed to optimize one free parameter — beam width for parsing. Sekine [20] gives an interesting report about the relationship between the beam width and the parsing accuracy. Generally, high parsing accuracy is expected when a large beam width is employed in the dependency structure analysis. However, the result is against our intuition. They report that a beam width between 3 and 10 gives the best parsing accuracy, and parsing accuracy falls down with a width larger than 10. This result suggests that Japanese dependency structures may consist of a series of local optimization processes.

We evaluate the relationship between the beam width and the parsing accuracy. Table 4.9 shows their relationships under the condition $d = 3$, along with the changes of the beam width from $k = 1$ to 15. The best parsing accuracy is achieved at $k = 5$ and the best sentence accuracy is achieved at $k = 5$ and $k = 7$.

We have to consider how we should set the beam width that gives the best parsing accuracy. We believe that the beam width that gives the best parsing accuracy is related not only with the length of the sentence, but also with the lexical entries and parts-of-speech that comprise the chunks.

Beam width	Dependency accuracy	Sentence accuracy
1	88.66%	45.76%
3	88.74%	45.20%
5	**88.77%**	**45.36%**
7	88.76%	**45.36%**
10	88.67%	45.28%
15	88.65%	45.28%

Table 4.9. Beam width vs Accuracy (6756 sentences, $d = 3$)

Probabilistic model	Dependency Accuracy	89.09% (10034/11263)
	Sentence Accuracy	46.17% (572/1239)
	Training Time	about 2 weeks
	Parsing Time	2.1 sec./sentence
Cascaded Chunking model	Dependency Accuracy	89.29% (10057/11263)
	Sentence Accuracy	47.53% (589/1239)
	Training Time	about 10 hours
	Parsing Time	0.5 sec./sentence

Table 4.10. Probabilistic model vs Cascaded Chunking model

4.7. Probabilistic model vs Cascaded Chunking model

Table 4.10 summarizes the best accuracies achieved with the probabilistic model and the cascaded chnking model.

Focusing on the difference of training and test time, we can conclude that the cascaded chunking model is significantly more efficient than the probabilistic model. Especially, the training time can be reduced from 2 weeks to 10 hours [1].

In addition, the parsing accuracy with the cascaded chunking model is slightly better than the accuracy with the probabilistic model. In the probabilistic model, the probabilities of dependencies are required to determine dependency relations. For this purpose, we carry out the learning by giving two segments which have dependency

[1] We carried out the training on AlphaSever 8400, and the testing on Linux (PentiumIII 1GHz)

relation in the training data as positive examples, and two segments which have no dependency relation as negative examples. Then, we calculate *pseudo* probabilities using some kinds of heuristics. Intuitively, we may conclude that higher accuracy will be achieved with probabilistic model, since we take all the candidates of dependency relation as training data, and the number of training data becomes larger. However, our proposed cascaded chunking model shows higher accuracy.

In the probabilistic framework, all candidates of two segments are selected as training data. Thus, if a segment appears to right-hand side of correct modifiee and has a similar function word with correct modifiee, this segment becomes an exception of negative example. In addition, if a segment is not modified by a modifier because of cross dependency constraints but has a similar function word with correct modifiee, this segment also becomes an exception. Actually, we cannot ignore these exceptions, since almost all segments will depend on immediately right-hand side segments. By selecting all candidates of dependency relation as training data without careful considerations, we have committed a number of exceptions needlessly. In order to avoid these needless commitments of exceptions, we have to distinguish three types of non-dependency relation:

1. Not modifying because the correct modifiee appears to left.
2. Not modifying because the correct modifiee appears to right.
3. Not modifying because of cross dependency constraints.

Taking a powerful heuristics for dependency structure analysis: "A segment modifies a nearer segment if possible", it will be most important that we give first priority to estimate whether the current segment modifies immediately right hand side segment or not. Our proposed cascaded chunking model are designed along with this heuristics indeed. In addition, we can avoid over-committing above mentioned exceptional examples since the training data are extracted as if we carry out parsing. Although the number of extracted training data becomes small, these extracted data appear to be more effective and valuable for dependency structure analysis.

From this result, we can conclude that our proposed cascaded chunking model is better than the probabilistic model from the viewpoints of efficiency as well as parsing accuracy.

4.8. Comparison with Related Work

Uchimoto and Sekine [23, 22, 20] report that using Kyoto University Corpus for their training and testing, they achieve around 87.93% accuracy by building statistical model based on Maximum Entropy framework. Kanyama also selects the ME framework for their parsing experiments with EDR corpus [9] .

In our experiments, we used exactly the same training and test data that Uchimoto used in order to make a fair comparison. In addition, we used almost same *static* features sets for our experiments. Using same training and test data, we can achieve accuracy of 89.29%. We think our model outperforms Uchimoto's model as far as the accuracies are compared.

Although Uchimoto suggests that the importance of considering combination of features, in ME framework we must expand these combination by introducing new feature set. Uchimoto heuristically selects *"effective"* combination of features. However, such a manual selection does not always cover all relevant combinations that are important in the determination of dependency relation. We think the week point of ME, that it has no potential to take the combinations of features automatically, is a one of main the factor that brings us a difference of parsing accuracy.

Although Decision Tree has a potential to take the combination of features, it is easy to fall into overffiting by itself. To avoid overfitting, Decision Tree is usually used as an *weak learner* for Boosting. In the recent statistical learning theories, SVMs and Boosting are mentioned as *Large Margin Classifiers*, and known to have a similar strategy with which one tries to maximize the *margin* between two classes. Combining Boosting technique with Decision Tree, there are possibilities that we can achieve higher accuracy. However, Haruno reported that the accuracy of dependency relation with Decision Tree fell down according to casting lexical entries with lower frequencies as feature sets[6]. We think that Decision Tree requires a careful feature selection (e.g. What kinds of lexical entries should be used for features) for achieving higher accuracy compared with SVMs.

On the other hand, SVMs has an ability to consider the combinations of given features set without increasing the computational overhead. In addition, SVMs have a high generalization ability even with these high dimensional expanded by some Kernel function.

Chapter 5

Future Work

Most of all statistical approaches for dependency structure analysis, including our models, only take the local series of segments for their detection and do not care many more components or features in a wider range of given sentence. The main theme for statistical approaches seems to be how we can employ the global contexts as features for machine learning algorithms. In this chapter, we describe the specific issues which may allow us to take the global contexts into account.

5.1. Coordinate Structure Analysis

Coordinate structures frequently appear in Japanese long sentences and make analysis hard. Most all previous statistical approaches for dependency structure analysis have no mechanism to deal with coordinate structures. For example, the probabilistic model assumes that each dependency relation is mutually independent, however, the pre- and post-conjuncts in a coordinate structures cannot be identified by this independence assumption. We think that a specific analysis for coordinate structures should be necessary for achieving higher accuracy.

Theoretically, our proposed cascaded chunkig model has an ability to detect coordinate structures simultaneously. Namely, by introducing a distinct tag set (for example, **P** tag) exclusively for coordinate structures, we can straightforwardly extend our proposed method to detect not only dependency relations but also coordinate structures.

In order to investigate how accurately our proposed cascaded chunking model with this simple extension can detect coordinate structures, we compare our proposed model

with KNP, which is a rule-based parser and detects coordinate structures accurately by considering the similarity of pre- and post-conjuncts in a coordinate structure[15]. Specifically, we compare them from the following two viewpoints:

1. Accuracy of dependency relation
2. Accuracy of coordinate structure

In order to deal with different three tag sets (**D,P,O**), we have to extend SVMs to multi-class classifiers. It is known that there are mainly two approaches to extend from a binary classification task to that of K classes. The first approach is typical called "*one class vs others.*" The idea is to build K classifiers that separate one class from all others. The second approach 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 majority voting. In our experiments, we decided to construct pairwise classifiers for all the pairs of tag sets because of the following two reasons:

- Some experiments [11] reports that combination of pairwise classifier perform better than K classifier.
- The amount of training data for a pair is less than the amount of training data for separating one class with all others.

For the test data, we used only the sentences out of all test data (1,246 sentences from the articles on January 9th) in which KNP determines all segments (bunsetsu) correctly. Actually, total 1132 sentences are chosen for the test data. Table 5.1 shows the results. In this table, the coordinate accuracy means the percentage of segments correctly determined the coordinate tag and dependency relation out of all segments with **P** tag in Kyoto University Corpus. Dependency accuracy means the percentage of correct dependencies out of all dependencies relation.

The dependency accuracy is slightly worse than that of KNP. However, we think the accuracy achieved with our model seems to be reasonable and valuable, since the syntactic rules defined in KNP is tuned manually or heuristically using Kyoto University Corpus as its training data. Thus, the accuracy of KNP can be recognized as the accuracy for *closed* data sets.

The accuracy of coordinate structures is worse than that of KNP by more than 10%. This is because our proposed method only takes the local series of segments as feature

	KNP	cascaded chunking model
Dependency Accuracy	89.68%	89.41%
Coordinate Accuracy	76.32%	65.87%

Table 5.1. Comparison with KNP

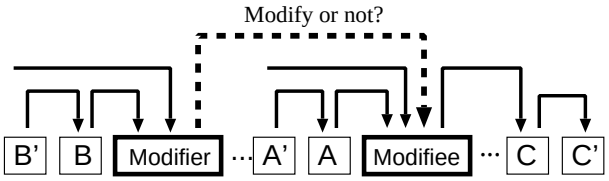


Figure 5.1. Recursive Dynamic Features

sets for actual training, and dose not care about the features covering wider range of the given sentence simultaneously. KNP detects a coordinate structures by calculating similarities between two arbitrary segments and employing dynamic programming method for detecting two most similar segments which can be reasonably considered as composing coordinate structures. However, the measure to calculate the similarities of two segments is optimized heuristically or manually, only considering quite small size of training data. We think such a manual optimization havs problems in coverage and consistency.

For future work, we like to improve the accuracy for coordinate structures, by taking whole information in given sentences.

5.2. Further investigation of dynamic features

In Chapter 3.4, we introduce three types of dynamic features (A,B,C), and present that all of these features are effective to improve the accuracy of dependency structure analysis. On the other hand, with the cascaded chunking framework, we can consider all dependency relations, which have narrower scope than that of current estimating two segments, as dynamic features. This implies that we are allowed to commit the dynamic features linking recursively to other dynamic features (Figure 5.1, A',B',C'). However, we may fall into overfitting if we introduce these recursively linking seg-

ments without careful consideration. In addition, it is not easy to represent these recursively linking segments as actual feature sets for individual machine learning algorithms. For future work, we like to investigate what kinds of dynamic features, including those recursively linking dependency relations, will improve the accuracy of dependency structure analysis.

Chapter 6

Conclusion

This thesis proposes Japanese dependency analysis based on Support Vector Machines. Through the experiments with Japanese bracketed corpus, the proposed method achieves a high accuracy even with a small training data and outperforms existing methods based on Maximum Entropy Models. The result shows that Japanese dependency analysis can be effectively performed by use of SVMs due to its good generalization and non-overfitting characteristics.

Acknowledgements

First of all, I would like to express my gratitude to Prof. Yuji Matsumoto, my supervisor. He introduced me to the world of statistical natural language processing during numerous extended discussion in our laboratory. I also would like to thank Prof. Kiyohiro Shikano, Associate Prof. Yasuharu Den and Dr. Takashi Miyata for helpful comments on this thesis.

Also, I would like to thank the members of machine learning work group in our laboratory : Hiroya Takamura, Hiroyasu Yamada, Hirotoshi Taira and Tetsuji Nakagawa. My significant interest to the machine learning is founded during the continuous discussion with them. From this excellent work group, I learned a lot about machine learning from the view points of theoretical analysis as well as real-world application.

Finally, I would like to thank all previous and current members of Prof. Matsumoto laboratory. They gave me a lot of useful and interesting knowledge about computer science.

References

[1] Eugene Charniak. A maximum-entropy-inspired parser. In *Processing of the NAACL 2000*, pages 132–139, 2000.

[2] Michael Collins. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the ACL '96*, pages 184–191, 1996.

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

[4] Y. Freund and Schapire. Experiments with a new Boosting algoritm. In *13th International Conference on Machine Learning*, 1996.

[5] Masakazu Fujio and Yuji Matsumoto. Japanese Dependency Structure Analysis based on Lexicalized Statistics. In *Proceedings of EMNLP '98*, pages 87–96, 1998.

[6] Masahiko Haruno, Satoshi Shirai, and Yoshifumi Ooyama. A Japanese Dependency Parser Based on a Decison tree. *Transactions of IPSJ*, 39(12):3117, 1998.

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

[8] Thorsten Joachims. Transductive Inference for Text Classification using Support Vector Machines. In *International Conference on Machine Learning (ICML)*, 1999.

[9] Hiroshi Kanayama, Kentaro Torisawa, Yutaka Mitsuishi, and Jun'ichi Tsujii. A Statistical Japanese Dependency Analysis Model with Choice Restricted to at Most Three Modification Candidates. *Natural Language Processing*, 7(5):71–91, 2000.

[10] T. Kasami. An efficient recognition and syntax analysis algorithm for context-free languages. Technical report, Air Force Cambridge Research Laboratory, Bedford, 1965.

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

[12] Taku Kudoh and Yuji Matsumoto. Chunking with Support Vector Machine. In *IPSJ SIG-NL 140*, pages 9–16, 2000.

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

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

[15] Sadao Kurohashi and Makoto Nagao. A Syntactic Analyisis Method of Long Japanese Sentences based on Coordinate Strucres' Detection. *Natural Language Processing*, 1(1):35–57, 1994.

[16] Sadao Kurohashi and Makoto Nagao. Kyoto University text corpus project. In *Proceedings of the ANLP, Japan*, pages 115–118, 1997.

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

[18] Adwait Ratnaparkhi. A Liner Observed Time Statistical Parser Based on Maximum Entropy Models. In *Proceedings of EMNLP '97*, 1997.

[19] Abney S. Parsing By Chunking. In *Principle-Based Parsing*. Kluwer Academic Publishers, 1991.

[20] Satoshi Sekine, Kiyotaka Uchimoto, and Hitoshi Isahara. Backward Beam Search Algorithm for Dependency Analysis of Japanese. In *Proceedings of the COLING 2000*, pages 754–760, 2000.

[21] Hirotoshi Taira and Masahiko Haruno. Feature Selection in SVM Text Categorization. *Transactions of IPSJ*, 41(4):1113, 2000.

[22] Kiyotaka Uchimoto, Masaki Murata, Satoshi Sekine, and Hitoshi Isahara. Dependency Model Using Posterior Context. *Natural Language Processing*, 7(5):3–17, 2000.

[23] Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. Japanese Dependency Structure Analysis Based on Maximum Entropy Models. *Transactions of IPSJ*, 40(9):3397–3407, 1998.

[24] Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. Japanese Dependency Structure Analysis Based on Maximum Entropy Models. In *Proceedings of the EACL*, pages 196–203, 1999.

[25] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[26] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.